

## **J Dictionary**



**Roger K.W. Hui**  
**Kenneth E. Iverson**

Copyright © 1991-2002 Jsoftware Inc. All Rights Reserved.  
Last updated: 2002-09-10  
[www.jsoftware.com](http://www.jsoftware.com)

# Table of Contents

1	<a href="#"><u>Introduction</u></a>
2	<a href="#"><u>Mnemonics</u></a>
3	<a href="#"><u>Ambivalence</u></a>
4	<a href="#"><u>Verbs and Adverbs</u></a>
5	<a href="#"><u>Punctuation</u></a>
6	<a href="#"><u>Forks</u></a>
7	<a href="#"><u>Programs</u></a>
8	<a href="#"><u>Bond Conjunction</u></a>
9	<a href="#"><u>Atop Conjunction</u></a>
10	<a href="#"><u>Vocabulary</u></a>
11	<a href="#"><u>Housekeeping</u></a>
12	<a href="#"><u>Power and Inverse</u></a>
13	<a href="#"><u>Reading and Writing</u></a>
14	<a href="#"><u>Format</u></a>
15	<a href="#"><u>Partitions</u></a>
16	<a href="#"><u>Defined Adverbs</u></a>
17	<a href="#"><u>Word Formation</u></a>
18	<a href="#"><u>Names and Displays</u></a>
19	<a href="#"><u>Explicit Definition</u></a>
20	<a href="#"><u>Tacit Equivalents</u></a>
21	<a href="#"><u>Rank</u></a>
22	<a href="#"><u>Gerund and Agenda</u></a>
23	<a href="#"><u>Recursion</u></a>
24	<a href="#"><u>Iteration</u></a>
25	<a href="#"><u>Trains</u></a>
26	<a href="#"><u>Permutations</u></a>
27	<a href="#"><u>Linear Functions</u></a>
28	<a href="#"><u>Obverse and Under</u></a>
29	<a href="#"><u>Identity Functions and Neutrals</u></a>
30	<a href="#"><u>Secondaries</u></a>
31	<a href="#"><u>Sample Topics</u></a>

32	<a href="#"><u>Spelling</u></a>
33	<a href="#"><u>Alphabet and Numbers</u></a>
34	<a href="#"><u>Grammar</u></a>
35	<a href="#"><u>Function Tables</u></a>
36	<a href="#"><u>Bordering a Table</u></a>
37	<a href="#"><u>Tables (Letter Frequency)</u></a>
38	<a href="#"><u>Tables</u></a>
39	<a href="#"><u>Classification</u></a>
40	<a href="#"><u>Disjoint Classification (Graphs)</u></a>
41	<a href="#"><u>Classification I</u></a>
42	<a href="#"><u>Classification II</u></a>
43	<a href="#"><u>Sorting</u></a>
44	<a href="#"><u>Compositions I</u></a>
45	<a href="#"><u>Compositions II</u></a>
46	<a href="#"><u>Junctions</u></a>
47	<a href="#"><u>Partitions I</u></a>
48	<a href="#"><u>Partitions II</u></a>
49	<a href="#"><u>Geometry</u></a>
50	<a href="#"><u>Symbolic Functions</u></a>
51	<a href="#"><u>Directed Graphs</u></a>
52	<a href="#"><u>Closure</u></a>
53	<a href="#"><u>Distance</u></a>
54	<a href="#"><u>Polynomials</u></a>
55	<a href="#"><u>Polynomials (Continued)</u></a>
56	<a href="#"><u>Polynomials in Terms of Roots</u></a>
57	<a href="#"><u>Polynomial Roots I</u></a>
58	<a href="#"><u>Polynomial Roots II</u></a>
59	<a href="#"><u>Polynomials: Stopes</u></a>
60	<a href="#"><u>Dictionary</u></a>
61	<a href="#"><u>I. Alphabet and Words</u></a>
62	<a href="#"><u>II. Grammar</u></a>
63	<a href="#"><u>A. Nouns</u></a>
64	<a href="#"><u>B. Verbs</u></a>
65	<a href="#"><u>C. Adverbs and Conjunctions</u></a>

66	<a href="#"><u>D. Comparatives</u></a>
67	<a href="#"><u>E. Parsing and Execution</u></a>
68	<a href="#"><u>F. Trains</u></a>
69	<a href="#"><u>G. Extended and Rational Arithmeti</u></a>
70	<a href="#"><u>H. Frets and Scripts</u></a>
71	<a href="#"><u>I. Locatives</u></a>
72	<a href="#"><u>J. Errors and Suspensions</u></a>
73	<a href="#"><u>III. Definitions</u></a>
74	<a href="#"><u>Vocabulary</u></a>
75	<a href="#"><u>= Self-Classify - Equal</u></a>
76	<a href="#"><u>=. Is (Local)</u></a>
77	<a href="#"><u>&lt; Box - Less Than</u></a>
78	<a href="#"><u>&lt;. Floor - Lesser Of (Min)</u></a>
79	<a href="#"><u>&lt;: Decrement - Less Or Equal</u></a>
80	<a href="#"><u>&gt; Open - Larger Than</u></a>
81	<a href="#"><u>&gt;. Ceiling - Larger of (Max)</u></a>
82	<a href="#"><u>&gt;: Increment - Larger Or Equal</u></a>
83	<a href="#"><u>_ Negative Sign / Infinity</u></a>
84	<a href="#"><u>_. Indeterminate</u></a>
85	<a href="#"><u>_: Infinity</u></a>
86	<a href="#"><u>+ Conjugate - Plus</u></a>
87	<a href="#"><u>+ . Real / Imaginary - GCD (Or)</u></a>
88	<a href="#"><u>+: Double • Not-Or</u></a>
89	<a href="#"><u>* Signum - Times</u></a>
90	<a href="#"><u>*. Length/Angle - LCM (And)</u></a>
91	<a href="#"><u>*: Square - Not-And</u></a>
92	<a href="#"><u>- Negate - Minus</u></a>
93	<a href="#"><u>-. Not - Less</u></a>
94	<a href="#"><u>-: Halve - Match</u></a>
95	<a href="#"><u>% Reciprocal - Divide</u></a>
96	<a href="#"><u>%. Matrix Inverse - Matrix Divide</u></a>
97	<a href="#"><u>%: Square Root - Root</u></a>
98	<a href="#"><u>^ Exponential - Power</u></a>
99	<a href="#"><u>^. Natural Log - Logarithm</u></a>

100     [^: Power](#)  
101     [^: Power v](#)  
102     [\\$ Shape Of - Shape](#)  
103     [\\$. Sparse](#)  
104     [\\$: Self-Reference](#)  
105     [~ Reflex - Passive](#)  
106     [EVOKE](#)  
107     [~. Nub -](#)  
108     [~: Nub Sieve - Not-Equal](#)  
109     [| Magnitude - Residue](#)  
110     [|. Reverse - Rotate \(Shift\)](#)  
111     [|: Transpose](#)  
112     [. Determinant • Dot Product](#)  
113     [.. Even](#)  
114     [: Explicit /](#)  
115     [Monad-Dyad](#)  
116     [:. Obverse](#)  
117     [:: Adverse](#)  
118     [, Ravel - Append](#)  
119     [., Ravel Items - Stitch](#)  
120     [,: Itemize - Laminate](#)  
121     [; Raze - Link](#)  
122     [;. Cut](#)  
123     [;; Word Formation -](#)  
124     [# Tally - Copy](#)  
125     [#. Base 2 - Base](#)  
126     [#: Antibase 2 - Antibase](#)  
127     [! Factorial - Out Of](#)  
128     [!. Fit \(Customize\)](#)  
129     [!: Foreign](#)  
130     [/ Insert - Table](#)  
131     [/. Oblique - Key](#)  
132     [/: Grade Up - Sort](#)  
133     [\ Prefix - Infix](#)

134 [\. Suffix - Outfix](#)  
135 [\: Grade Down - Sort](#)  
136 [\[ Same - Left](#)  
137 [\[: Cap](#)  
138 [{ Catalogue - From](#)  
139 [{. Head - Take](#)  
140  
141 [{:: Map - Fetch](#)  
142 [} Item Amend - Amend](#)  
143 [} Item Amend • Amend](#)  
144 [}. Behead - Drop](#)  
145 [}: Curtail -](#)  
146 [" Rank](#)  
147 [" Rank](#)  
148 [" Rank](#)  
149 [". Do - Numbers](#)  
150 [": Default Format - Format](#)  
151 [` Tie \(Gerund\)](#)  
152 [`: Evoke Gerund](#)  
153 [@ Atop](#)  
154 [@. Agenda](#)  
155 [@: At](#)  
156 [& Bond /](#)  
157 [Compose](#)  
158 [&.](#)  
159 [&.:](#)  
160 [&: Appose](#)  
161 [? Roll - Deal](#)  
162 [a. Alphabet](#)  
163 [A. Anagram Index - Anagram](#)  
164 [b. Boolean /](#)  
165 [Basic](#)  
166 [c. Characteristic Values](#)  
167 [C. Cycle-Direct - Permute](#)

168 [d. Derivative](#)  
169 [D. Derivative](#)  
170 [D: Secant Slope](#)  
171 [e. Raze In - Member \(In\)](#)  
172 [E. - Member of Interval](#)  
173 [f. Fix](#)  
174 [H. Hypergeometric](#)  
175 [i. Integers - Index Of](#)  
176 [i: Integers - Index Of Last](#)  
177 [j. Imaginary - Complex](#)  
178 [L. Level Of](#)  
179 [L: Level At](#)  
180 [m. n. Explicit Noun Args](#)  
181 [NB. Comment](#)  
182 [o. Pi Times - Circle Function](#)  
183 [p. Polynomial](#)  
184 [p.. Poly. Deriv. - Poly. Integral](#)  
185 [p: Primes -](#)  
186 [q: Prime Factors - Prime Exponents](#)  
187 [r. Angle - Polar](#)  
188 [s: Symbol](#)  
189 [S: Spread](#)  
190 [t. Taylor Coefficient](#)  
191 [t. Taylor Coefficient](#)  
192 [t: Weighted Taylor](#)  
193 [T. Taylor Approximation](#)  
194 [u. v. Explicit Verb Args](#)  
195 [u: Unicode](#)  
196 [x. y. Explicit Arguments](#)  
197 [x: Extended Precision](#)  
198 [\\_9: to 9: Constant Functions](#)  
199 [Constants](#)  
200 [Control structures](#)  
201 [assert.](#)

202     [break.](#)  
203     [continue.](#)  
204     [for.](#)  
205     [goto\\_name.](#)  
206     [if.](#)  
207     [return.](#)  
208     [select.](#)  
209     [throw.](#)  
210     [try.](#)  
211     [while.](#)  
212     [References](#)  
213     [Acknowledgments](#)  
214     [Foreign conjunction](#)  
215     [0!: Scripts](#)  
216     [1!: Files](#)  
217     [2!: Host](#)  
218     [3!: Conversions](#)  
219     [4!: Names](#)  
220     [5!: Representation](#)  
221     [6!: Time](#)  
222     [7!: Space](#)  
223     [9!: Global Parameters](#)  
224     [11!: Window Driver](#)  
225     [13!: Debug](#)  
226     [15!: Dynamic Link Library](#)  
227     [18!: Locales](#)  
228     [128!: Miscellaneous](#)  
229     [Special Code](#)  
230     [System Limits](#)  
231     [Index](#)



# Introduction

**J** is a general-purpose programming language available on a wide variety of computers. Although it has a simple structure and is readily learned by anyone familiar with mathematical notions and notation, its distinctive features may make it difficult for anyone familiar with more conventional programming languages.

This introduction is designed to present **J** in a manner that makes it easily accessible to programmers, by emphasizing those aspects that distinguish it from other languages. These include:

1. A mnemonic one- or two-character spelling for primitives.
2. No order-of-execution hierarchy among functions.
3. The systematic use of *ambivalent* functions that, like the minus sign in arithmetic, can denote one function when used with two arguments (*subtraction* in the case of  $-$ ), and another when used with one argument (*negation* in the case of  $-$ ).
4. The adoption of terms from English grammar that better fit the grammar of **J** than do the terms commonly used in mathematics and in programming languages. Thus, a function such as addition is also called a *verb* (because it performs an action), and an entity that modifies a verb (not available in most programming languages) is accordingly called an *adverb*.
5. The systematic use of adverbs and conjunctions to modify verbs, so as to provide a rich set of operations based upon a rather small set of verbs. For example,  $+/a$  denotes the sum over a list  $a$ ,  $*/a$  denotes the product over  $a$ ,  $a */ b$  is the multiplication table of  $a$  and  $b$ .
6. The treatment of vectors, matrices, and other arrays as single entities.

7. The use of *functional* or *tacit* programming that requires no explicit mention of the arguments of a function (program) being defined, and the use of assignment to assign names to functions (as in `sum=:+/` and `mean=:sum % #`).

The following sections are records of actual **J** sessions, accompanied by commentary that should be read only after studying the corresponding session (and perhaps experimenting with variations on the computer). The sections should be studied with the **J** system and dictionary at hand, and the exercises should be attempted. The reckless reader may go directly to the sample topics.

# 1. Mnemonics

The left side of the page shows an actual computer session with the result of each sentence shown at the left margin. First cover the comments at the right, and then attempt to state in English the meaning of each primitive so as to make clear the relations between related symbols. For example, "< is *less than*" and "<. is *lesser of* (that is, minimum)". Then uncover the comments and compare with your own.

0	7<5 Less than	A zero is interpreted as <i>false</i> .
5	7< .5	Lesser of
1	7>5	Greater than A one is <i>true</i> ( <i>à la</i> George Boole)
7	7> .5	Greater of
1000	10^3	Power ( <i>à la</i> Augustus De Morgan)
3	10^.1000	Logarithm
0	7=5	Equals
5	b=: 5 7<.b	Is ( <i>assignment</i> or <i>copula</i> )
	Min=: <. power=: ^ gt=: >	Min is <. power is ^ gt is >
1000	10 power (5 Min 3)	

## Exercises

- 1.1 Enter the following sentences on the computer, observe the results, give suitable names to any new primitives (such as `*` and `+.`  and `*.` ), and comment on their behaviour.

```
a=:0 1 2 3
b=:3 2 1 0
a+b
a*b
a-b
a%b
a^b
a^.b
a<b
a>b
(a<b)+(a>b)
(a<b)+. (a>b)
```

Compare your comments with the following:

- a) Negative `3` is denoted by `_3`. The underbar `_` is part of the representation of a negative number in the same sense that the decimal point is part of the representation of one-half when written in the form `0.5`, and the negative sign `_` must not be confused with the symbol used to denote subtraction (i.e., `-`).
- b) Division (`%`) by zero yields infinity, denoted by the underbar alone.
- c) Log of `2` to the base `1` is infinite, and log of `0` to the base `3` is negative infinity (`__`).
- d) Since the relation `5<7` is true, and the result of `5<7` is `1`, it may be said that true and false are represented by the ordinary integers `1` and `0`. George Boole used this same convention, together with the symbol `+` to represent the *boolean* function *or*. We use the distinct representation `+.`  to avoid conflict with the analogous (but different) *addition* (denoted by `+`).

- 1.2 Following the example `Min=: <.` , invent, assign, and use names for each of the primitives encountered thus far.

## 2. Ambivalence

Cover the comments on the right and provide your own.

```
7-5
2
-5
_5
```

The function in the sentence `7-5` applies to two arguments to perform subtraction, but in the sentence `-5` it applies to a single argument to perform negation.

Adopting from chemistry the term valence, we say that the symbol `-` is *ambivalent*, its effective binding power being determined by context.

```
7%5
1.4
```

The ambivalence of `-` is familiar in arithmetic; it is here extended to other functions.

```
%5
0.2
```

```
3^2
9
```

*Exponential* (that is, `2.71828^2`)

```
^2
7.38906
```

```
a=: i. 5
a
0 1 2 3 4
```

The function *integer* or *integer list*

*List* or *vector*

```
a i. 3 1
3 1
```

The function *index* or *index of*

```
b=: 'Canada'
b i. 'da'
4 1
```

Enclosing quotes denote literal characters

```
$ a
5
```

*Shape* function

```
3 4 $ a
0 1 2 3
4 0 1 2
```

*Reshape* function  
*Table* or *matrix*

```
3 4 0 1
```

```
3 4 $ b
```

```
Cana
```

```
daCa
```

```
nada
```

```
%a
```

```
_ 1 0.5 0.333333 0.25
```

Functions apply to lists

The symbol `_` alone denotes *infinity*

## Exercises

- 2.1 Enter the following sentences (and perhaps related sentences using different arguments), observe the results, and state what the two cases (*monadic* and *dyadic*) of each function do:

```
a=: 3 1 4 1 5 9
b=: 'Canada'
#a
1 0 1 0 1 3 # a
1 0 1 0 1 3 # b
/: a
/: b
a /: a
a /: b
b /: a
b /: b
c=: 'can' 't'
c
#c
c /: c
```

2.2 Make a summary table of the functions used thus far. Then compare it with the following table (in which a bullet separates the monadic case from the dyadic, as in Negate • Subtract):

		•	• •
+	• Add	• Or	
–	Negate • Subtract		
*	• Times	• And	
%	Reciprocal • Divide		
^	Exponential • Power	• Log	
<	• Less Than	• Lessor Of	
>	• Greater Than	• Greater Of	
=	• Equals		Is (Copula)
i		Integers • Index Of	
\$	Shape • Reshape		
/			Grade • Sort
#	Number Of Items • Replicate		

2.3 Try to fill some of the gaps in the table of Exercise 2.2 by experimenting on the computer with appropriate expressions. For example, enter `^.10` and `^. 2.71828` to determine the missing (monadic) case of `^.` and enter `%: 4` and `%: -4` and `+%: -4` to determine the case of `%` followed by a colon.

However, do not waste time on matters (such as, perhaps, complex numbers or the *boxed* results produced by the monad `<`) that are still beyond your grasp; it may be better to return to them after working through later sections. Note that the effects of certain functions become evident only when applied to arguments other than positive integers: try `<.1 2 3 4` and `<.3.4 5.2 3.6` to determine the effect of the monad `<.`

- 2.4 If `b =: 3.4 5.2 3.6`, then `<.b` yields the argument `b` rounded *down* to the nearest integer. Write and test a sentence that rounds the argument `b` to the *nearest* integer.

Answer: `<.(b+0.5)` or `<.b+0.5` or `<.b+1r2`

- 2.5 Enter `2 4 3 $ i. 5` to see an example of a *rank 3 array* or *report* (for two years of four quarters of three months each).
- 2.6 Enter `?9` repeatedly and state what the function `?` does. Then enter `t=: ?3 5 $ 9` to make a table for use in further experiments.

Answer: `?` is a (pseudo-) random number generator; `?n` produces an element from the population `i.n`



### 3. Verbs and Adverbs

In the sentence %a of Section 2, the % "acts upon" a to produce a result, and %a is therefore analogous to the notion in English of a *verb* acting upon a noun or pronoun. We will hereafter adopt the term *verb* instead of (or in addition to) the mathematical term *function* used thus far.

The sentence +/ 1 2 3 4 is equivalent to 1+2+3+4; the adverb / applies to *its* verb argument + to produce a new verb whose argument is 1 2 3 4, and which is defined by inserting the verb + between the items of its argument. Other arguments of the insert adverb are treated similarly:

```
*/b=:2 7 1 8 2 8
1792
```

```
<./b
1
```

```
>./b
8
```

The verb resulting from the application of an adverb may (like a primitive verb) have both monadic and dyadic cases; due to its two uses, the adverb / is called either *insert* or *table*. In the present instance of / the dyadic case produces a *table*. For example:

```
2 3 5 +/ 0 1 2 3
2 3 4 5
3 4 5 6
5 6 7 8
```

The verbs over=:({.;}.)@":@, and by=: ' '&i@,.@[,.] can be entered as utilities (for use rather than for immediate study), and can clarify the interpretation of *function tables* such as the addition table produced above. For example:

```
a=: 2 3 5
b=: 0 1 2 3

a by b over a +/ b
```

```

+--+-----+
|  |0 1 2 3|
+--+-----+
|2|2 3 4 5|
|3|3 4 5 6|
|5|5 6 7 8|
+--+-----+
      b by b over b </ b
+--+-----+
|  |0 1 2 3|
+--+-----+
|0|0 1 1 1|
|1|0 0 1 1|
|2|0 0 0 1|
|3|0 0 0 0|
+--+-----+

```

## Exercises

- 3.1 Enter `d=: i.5` and the sentences `st=: d-/d` and `pt=: d^/d` to produce function tables for subtraction and power.
- 3.2 Make tables for further functions from previous sections, including the relations `<` and `=` and `>` and *lesser-of* and *greater-of*.
- 3.3 Apply the verbs `|.` and `|:` to various tables, and try to state what they do.
- 3.4 The *transpose* function `|:` changes the subtraction table, but appears to have no effect on the multiplication table. State the property of those functions whose tables remain unchanged when transposed.

Answer: They are commutative

- 3.5 Enter `d by d over d! /d` and state the definition of the dyad `!`.

Answer: `!` is the binomial coefficient or *outof* function; `3!5` is the number of ways that three things can be chosen from five.

## 4. Punctuation

English employs various symbols to *punctuate* a sentence, to indicate the order in which its phrases are to be interpreted. Thus:

The teacher said he was stupid.

The teacher, said he, was stupid.

Math also employs various devices (primarily parentheses) to specify order of interpretation or, as it is usually called, *order of execution*. It also employs a set of rules for an unparenthesized phrase, including a hierarchy amongst functions. For example, *power* is executed before *times*, which is executed before *addition*.

**J** uses parentheses for punctuation, together with the following rules for unparenthesized phrases:

The right argument of a verb is the value of the entire phrase to its right.

Adverbs are applied first. Thus, the phrase  $a */ b$  is equivalent to  $a ( */ ) b$ , not to  $a ( */ b )$ .

For example:

$a = : 5$

$b = : 3$

$(a * a) + (b * b)$

34

$a * a + b * b$

70

$a * (a + (b * b))$

70

$(a + b) * (a - b)$

16

$a ( + * - ) b$

16

The last sentence above includes the *isolated* phrase  $+ * -$  which has thus far not been assigned a meaning. It is called a *trident* or *fork*, and is equivalent to the sentence that precedes it.

A fork also has a monadic meaning, as illustrated for the *mean* below:

$c = : 2 \ 3 \ 4 \ 5 \ 6$   
 $(+ / \% \#) \ c$   
 argument  
 4

The verb  $\#$  yields the number of items in its  
 $(+ / c) \% (\# c)$   
 4

## Exercises

4.1 In math, the expression  $3x^4 + 4x^3 + 5x^2$  is called a *polynomial*. Enter:

$x = : 2$   
 $(3 * x^4) + (4 * x^3) + (5 * x^2)$

to evaluate the polynomial for the case where  $x$  is 2.

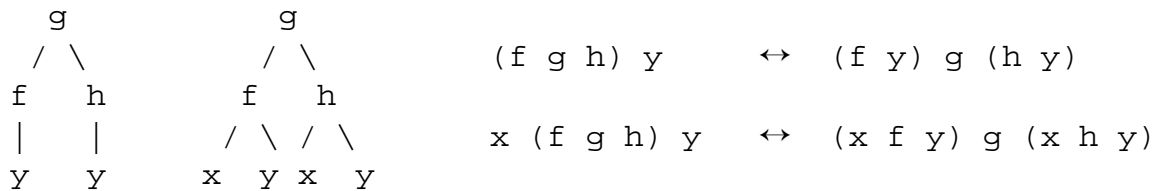
4.2 Note that the hierarchy among functions used in math is such that no parentheses are necessary in writing a polynomial. Write an equivalent sentence using no parentheses.

Answer:  $+ / 3 \ 4 \ 5 \ * \ x \ ^ \ 4 \ 3 \ 2$

or (first assigning names to the *coefficients*  $3 \ 4 \ 5$  and the *exponents*  $4 \ 3 \ 2$ ), as:  $+ / c * x ^ e$

## 5. Forks

As illustrated above, an isolated sequence of three verbs is called a *fork*; its monadic and dyadic cases are defined by:



The diagrams above provide visual images of the fork.

Before reading the notes at the right (and by using the facts that  $\%:$  denotes the *root* function and  $]$  denotes the *identity* function), try to state in English the significance of each of the following sentences:

a=: 8 7 6 5 4 3	
b=: 4 5 6 7 8 9	
2 %: b	Square root of b
2 2.23607 2.44949 2.64575 2.82843 3	

3 %: b	Cube root of b
1.5874 1.70998 1.81712 1.91293 2 2.08008	

(+ / % #) b	Arithmetic mean or average
6.5	

(# %: * /) b	Geometric mean
6.26521	

([ ] - (+ / % #)) b	Centre on mean (two forks)
_2.5 _1.5 _0.5 0.5 1.5 2.5	

([ ] - + / % #) b	Two forks (fewer parentheses)
_2.5 _1.5 _0.5 0.5 1.5 2.5	

a (+ * -) b	Dyadic case of fork
-------------	---------------------

48 24 0 \_24 \_48 \_72

(a^2)-(b^2)  
48 24 0 \_24 \_48 \_72

a (< +. =) b                      Less than or equal  
0 0 1 1 1 1

a<b  
0 0 0 1 1 1

a=b  
0 0 1 0 0 0

a (<: = < +. =) b                      A tautology (<: is *less than or equal*)  
1 1 1 1 1 1

2 ([: ^ -) 0 1 2                      Cap yields monadic case  
7.38906 2.71828 1

evens=: [: +: i.  
evens 7                      +: is *double*  
0 2 4 6 8 10 12

odds=: [: >: evens                      >: is *increment*  
odds 7  
1 3 5 7 9 11 13

## Exercises

- 5.1 Enter 5#3 and similar expressions to determine the definition of the dyad # and then state the meaning of the following sentence:

(# # >./) b=: 2 7 1 8 2

Answer: #b repetitions of the maximum over b

5.2 Cover the comments on the right, write your own interpretation of each sentence, and then compare your statements with those on the right:

$(+ / \% \#) b$

$(\# \# + / \% \#) b$

$+ / (\# \# + / \% \#) b$

$(+ / b) = + / (\# \# + / \% \#) b$

$(* / b) = * / (\# \# \# \% : * /) b$

Mean of  $b$

$(n = \#b)$  repetitions of mean

Sum of  $n$  means

Tautology

The product over  $b$  is the product over  $n$  repetitions of the geometric mean of  $b$

## 6. Programs

A *program* handed out at a musical evening describes the sequence of musical pieces to be performed. As suggested by its roots *gram* and *pro*, a program is something written in advance of the events it prescribes.

Similarly, the fork `+ / % #` of the preceding section is a program that prescribes the computation of the mean of its argument when it is applied, as in the sentence `(+ / % #) b`. However, we would not normally call the procedure a program until we assign a name to it, as illustrated below:

```
mean=: + / % #
mean 2 3 4 5 6
4

(mean=: # %: */ ) 2 3 4 5 6
3.72792
```

Since the program `mean` is a new *verb*, we also refer to a sentence such as `mean=: + / % #` as *verb definition* (or *definition*), and to the resulting verb as a *defined* verb or function.

Defined verbs can be used in the definition of further verbs in a manner sometimes referred to as *structured programming*. For example:

```
MEAN=: sum % #
sum=: + /
MEAN 2 3 4 5 6
4
```

Entry of a verb alone (without an argument) displays its definition, and the foreign conjunction `(! :)` can be used to specify the form of the display: boxed, tree, linear, or parens. (The session can also be configured to specify the form of verb display, under the menu item Edit|Configure...|View.) Thus:

```
mean
+ / % #

9! : 3 (2 4 5)
```



```

mean
+-----+---+
|+-+--+|%|#| | |
| | + | / | | |
|+-+--+| | |
+-----+---+
+- / --- +
--+- %
+- #
+ / % #

```

## Exercises

6.1 Enter `AT=: i. +/ i.` and use expressions such as `AT 5` to determine the behaviour of the program `AT`.

6.2 Define and use similar function tables for other dyadic functions.

6.3 Define the programs:

```

tab=: +/
ft=: i. tab i.
test1=: ft = AT

```

Then apply `test1` to various integer arguments to test the proposition that `ft` is equivalent to `AT` of Exercise 6.1, and enter `ft` and `AT` alone to display their definitions.

6.4 Define `aft=: ft f.` and use `test2=: aft = ft` to test their equivalence. Then display their definitions and state the effect of the adverb `f.`.

Answer: The adverb `f.` *fixes* the verb to which it applies, replacing each name used by its definition.

6.5 Redefine `tab` of Exercise 6.3 by entering `tab=: */` and observe the effects on `ft` and its *fixed* alternative `aft`.

6.6 Define `mean=: +/ % #` and state its behaviour when applied to a *table*, as in `mean t=: (i. !/ i.) 5`.

Answer: The result is the average *of*, not *over*, the rows of a table argument.

6.7 Write an expression for the mean of the *columns* of `t`.

Answer: `mean |: t` or `mean"1 t`

## 7. Bond Conjunction

A dyad such as `^` can be used to provide a family of monadic functions:

```

]b=: i.7
0 1 2 3 4 5 6

b^2                      Squares
0 1 4 9 16 25 36

b^3                      Cubes
0 1 8 27 64 125 216

b^0.5                    Square roots
0 1 1.41421 1.73205 2 2.23607 2.44949

```

The *bond* conjunction `&` can be used to bind an argument to a dyad in order to produce a corresponding defined verb. For example:

```

square=: ^&2              Square (power and 2)
square b
0 1 4 9 16 25 36

(sqrt=: ^&0.5) b          Square root function
0 1 1.41421 1.73205 2 2.23607 2.44949

```

A left argument can be similarly bound:

```

Log=: 10&^.              Base-10 logarithm
Log 2 4 6 8 10 100 1000
0.30103 0.60206 0.778151 0.90309 1 2 3

```

Such defined verbs can of course be used in forks. For example:

```

in29=: 2&< *. <&9         Interval test
in29 0 1 2 5 8 13 21
0 0 0 1 1 0 0

IN29=: in29 # ]           Interval selection
IN29 0 1 2 5 8 13 21
5 8

```

```

LOE=: <+. =
5 LOE 3 4 5 6 7
0 0 1 1 1

integertest=: <. = ]      The monad <. is floor
integertest 0 0.5 1 1.5 2 2.5 3
1 0 1 0 1 0 1

int=: integertest
int (i.13)%3
1 0 0 1 0 0 1 0 0 1 0 0 1

```

## Exercises

- 7.1 The verb `#` is used dyadically in the program `IN29`. Enter expressions such as `(j=: 3 0 4 0 1) # i.5` to determine the behaviour of `#`, and state the result of `#j#i.5`. (Also try `1j1#i.5`.)

Answer: `+/j`

- 7.2 Cover the answers on the right and apply the following programs to lists to determine (and state in English) the purpose of each:

<code>test1=: &gt;&amp;10 *. &lt;&amp;100</code>	Test if in 10 to 100
<code>int=: ] = &lt;.</code>	Test if integer
<code>test2=: int *. test1</code>	Test if integer and in 10 to 100
<code>test3=: int +. test1</code>	Test if integer or in 10 to 100
<code>sel=: test2 # ]</code>	Select integers in 10 to 100

- 7.3 Cover the program definitions on the left of the preceding exercise, and make new programs for the stated effects.
- 7.4 Review the use of the *fix* adverb in Exercises 6.4-5, and experiment with its use on the programs of Exercise 7.2.

## 8. Atop Conjunction

The conjunction @ applies to two verbs to produce a verb that is equivalent to applying the first *atop* the second. For example:

```
TriplePowersOf2=: (3&*)@(2&^)  
TriplePowersOf2 0 1 2 3 4  
3 6 12 24 48
```

```
CubeOfDiff=: (^&3)@-  
3 4 5 6 CubeOfDiff 6 5 4 3  
_27 _1 1 27
```

```
f=: ^@-
```

The rightmost function is first applied dyadically if possible; the second is applied monadically.

```
5 f 3  
7.38906
```

```
f 3  
0.0497871
```

```
g=: -@^  
5 g 3  
_125
```

```
g 3  
_20.0855
```

A conjunction, like an adverb, is executed before verbs; the *left* argument of either is the entire verb phrase that precedes it. Consequently, some (but not all) of the parentheses in the foregoing definitions can be omitted. For example:

```
COD=: ^&3@-  
3 4 5 6 COD 6 5 4 3  
_27 _1 1 27
```

```
TPO2=: 3&*(2&^)  
  
TPO2 0 1 2 3 4  
3 6 12 24 48
```

```

    tpo2=: 3&*@2&^
|domain error
|    tpo2=:      3&*@2&^

```

An error because the conjunction @ is defined only for a verb right argument

## Exercises

8.1 Cover the comments on the right, and state the effects of the programs. Then cover the programs and rewrite them from the English statements:

```

mc=: (+/%#)@| :
f=: +/ @ (^&2)
g=: %:@f
h=: {&' *'@(</)
k=: i. h i.
map=: {&' +-* %#$'
MAP=: map@ (6&<.)
add=: MAP@ (i.+/i.)

```

Means of columns of table  
Sum of squares of list  
Geometric length of list  
Map of comparison (dyad)  
Map (monad)  
7-character map  
Extended domain of map  
Addition table map

## 9. Vocabulary

Memorizing lists of words is a tedious and ineffectual way to learn a language, and better techniques should be employed:

- A) Conversation with a native speaker who allows you to do most of the talking.
- B) Reading material of interest in its own right.
- C) Learning how to use dictionaries and grammars so as to become independent of teachers.
- D) Attempting to *write* on any topic of interest in itself.
- E) Paying attention to the *structure* of words so that known words will provide clues to the unknown. For example, *program* (already analyzed) is related to *tele* (far off) *gram*, which is in turn related to *telephone*. Even tiny words may possess informative structure: *atom* means not cuttable, from *a* (not) and *tom* (as in *tome* and *microtome*).

In the case of **J**:

- A) The computer provides for precise and general conversation.
- B) Texts such as [Fractals, Visualization and J p212](#)[7], [Exploring Math p212](#)[8], and [Concrete Math Companion p212](#) [14] use the language in a variety of topics.
- C) The appended dictionary of **J** provides a complete and concise dictionary and grammar.
- D) [J Phrases p212](#) [9] provides guidance in writing programs, and almost any topic provides problems of a wide range of difficulty.
- E) Words possess considerable structure, as in  $+:$  and  $-:$  and  $*:$  and  $\%:$  for *double*, *halve*, *square*, and *square root*. Moreover, a beginner can assign and use mnemonic names appropriate to any native language, as in `sqrt=:%` and `entier=:<.` (French name) and `sin=:1&o.` and `SIN=:1&o. @ (%&180@o.)` (for sine in degrees).

We will hereafter introduce and use new primitives with little or no discussion, assuming that the reader will experiment with them on the computer, consult the

dictionary to determine their meanings, or perhaps infer their meanings from their structure. For example, the appearance of the word `o.` suggests a circle; it was used dyadically above to define the sine (one of the *circular* functions), and monadically for the function *pi times*, that is, the circumference of a *circle* when applied to its diameter.

For precise oral communication it may be best to use the names (or abbreviations) of the symbols themselves, as in:

<	Left a (angle)	/	Slash	&	Amp (ersand)	%	Per (cent)
[	Left b (racket)	\	Back (slash)	@	At	;	Semi (colon)
{	Left c (urly bracket)		Stile	^	Caret	~	Tilde
(	Left p (arenthesis)	_	(Under) Bar	`	Grave	*	Star

## Exercises

- 9.1 Experiment with a revised version of the program `MAP` of Exercise 7.1, using the *remainder* or *residue* dyad `(|)` instead of the *minimum* `(<.)`, as in `M=:map@(6&|)` and compare its results with those of `MAP`.
- 9.2 Experiment with the programs `sin` and `SIN` defined in this section.
- 9.3 Write programs using various new primitives found in the vocabulary at the end of the book.
- 9.4 Update the table of notation prepared in Exercise 2.2.



## 10. Housekeeping

In an extended session it may be difficult to remember the names assigned to verbs and nouns; the *foreign* conjunction `!:` (detailed in [Appendix A p214](#)) provides facilities for displaying and erasing them. For example:

```

b=: 3* a=: i. 6
sum=: +/
tri=: sum\ a
names=: 4!:1

names 0
+--+-----+
|a|b|tri|
+--+-----+

names 3
+-----+-----+
|names|sum|
+-----+-----+

erase=: 4!:55
erase <'tri'
1

names 0 3
+--+-----+-----+-----+
|a|b|erase|names|sum|
+--+-----+-----+-----+

erase names 0
1 1

names 0 3
+-----+-----+-----+
|erase|names|sum|
+-----+-----+-----+

```

The Windows drop-down menus can be used to save, retrieve, and print sessions. They can also be used to open *script windows*, in which any number of sentences

may be entered and edited, and from which they can be *executed* so as to appear in the normal execution window.

The script and execution windows can also be displayed side-by-side, so that the effect of executing scripts can be directly observed.

These matters are treated in detail in the help files, as are the housekeeping facilities provided by the script file *profile.ijs*. The menus may be used to execute this file explicitly so as to make its facilities available. If, however, its name is included in the *command line* (as described in the user manual) it is executed automatically at the beginning of a session.

## Exercises

10.1 Enter and experiment with the programs defined in this section.

10.2 Type in the sentence `+ / 2 3 5 * i . 3` and press the Enter key to execute it.

The following exercises describe facilities available only under Windows and Macintosh.

10.3 Use the Up-arrow cursor key to move the cursor back up the line entered in Exercise 10.2, and then:

Press Enter to bring the line down to the input area.

Use the Left-arrow key to move the cursor back to the `*` symbol, and the backspace key to erase it.

Enter `-` to replace the multiplication by subtraction, and press the Enter key to execute the revised sentence.

10.4 Use cursor keys to move the cursor to the immediate left of the `i` in the sentence executed in Exercise 10.3. Then hold down the Control key and press the F1 key to display Dictionary definition of the primitive `i`.

10.5 Press the Escape key to close the display invoked in Exercise 10.4. Then move the cursor to the left so that it is separated from the line by one or more spaces, and again perform Ctrl F1 to display the individually boxed words in the sentence.



# 11. Power and Inverse

The power conjunction ( $\wedge$ ) is analogous to the power function ( $^$ ). For example:

```

]a=: 10^ b=: i.5
1 10 100 1000 10000

b
0 1 2 3 4

%:a
1 3.16228 10 31.6228 100

%: %: a
1 1.77828 3.16228 5.62341 10

%: ^: 2 a
1 1.77828 3.16228 5.62341 10

%: ^: 3 a
1 1.33352 1.77828 2.37137 3.16228

%: ^: b a
1      10      100      1000      10000
1 3.16228      10 31.6228      100
1 1.77828 3.16228 5.62341      10
1 1.33352 1.77828 2.37137 3.16228
1 1.15478 1.33352 1.53993 1.77828

(cos=: 2&o.) ^: b d=:1
1 0.540302 0.857553 0.65429 0.79348

] y=: cos ^: _ d
0.739085

y=cos y
1

```

Successive applications of `cos` appear to be converging to a limiting value; the infinite power (`cos ^: _`) yields this limit.

A right argument of `_1` produces the *inverse* function. Thus:

```
%: ^: _1 b
0 1 4 9 16
```

```
*: b
0 1 4 9 16
```

```
%: ^: (-b) b
0 1      2      3      4
0 1      4      9      16
0 1     16     81     256
0 1    256    6561    65536
0 1 65536 4.30467e7 4.29497e9
```

## Exercises

- 11.1 The square function `*:` is the inverse of the square root function `%:` and `%: ^: _1` is therefore equivalent to `*:`. Look for, and experiment with, other inverse pairs among the primitive functions in the Vocabulary.

## 12. Reading and Writing

Translating to and from a known language provides useful beginning exercises in learning a new one. The following provides examples.

**Cover** the **right** side of the page and make a serious attempt to translate the sentences on the **left** to English; that is, state succinctly in English what the verb defined by each sentence does. Use any available aids, including the dictionary and experimentation on the computer:

f1=: <:	Decrement (monad); Less or equal
f2=: f1&9	Less or equal 9
f3=: f2 *. >:&2	Interval test 2 to 9 (inclusive)
f4=: f3 *. <. = ]	In 2 to 9 and integer
f5=: f3 +. <. = ]	In 2 to 9 or integer
g1=: %&1.8	Divide by 1.8
g2=: g1^:_1	Multiply by 1.8
g3=: -&32	Subtract 32
g4=: g3^:_1	Add 32
g5=: g1@g3	Celsius from Fahrenheit
g6=: g5^:_1	Fahrenheit from Celsius
h1=: >./	Maximum over list (monad)
h2=: h1-<./	Spread. Try h2 b with the parabola: b=: (-&2 * -&3) -:i.12
h3=: h1@]-i.@[*h2@]%<:@[	Grid. Try 10 h3 b
h4=: h3 <:/ ]	Barchart. Try 10 h4 b
h5=: {&' *' @ h4	Barchart. Try 10 h5 b

After entering the foregoing definitions, enter each verb name alone to display its

definition, and learn to interpret the resulting displays. To see several forms of display, first enter `9! : 3 ( 2 4 5 ) .`

**Cover** the **left** side of the page, and translate the English definitions on the **right** back into **J**.

## Exercises

- 12.1 These exercises are grouped by topic and organized like the section, with programs that are first to be read and then to be rewritten. However, a reader already familiar with a given topic might begin by writing.

### A. Properties of numbers

<code>pn=: &gt;:@i.</code>	Positive numbers (e.g. <code>pn 9</code> )
<code>rt=: pn   / pn</code>	Remainder table
<code>dt=: 0&amp;=@rt</code>	Divisibility table
<code>nd=: +/@dt</code>	Number of divisors
<code>prt=: 2&amp;=@nd</code>	Prime test
<code>prsel=: prt # pn</code>	Prime select
<code>N=: &gt;:@pn</code>	Numbers greater than 1
<code>rtt=: ,@(N */ N)</code>	Ravelled times table
<code>aprt=: -.@(N e. rtt)</code>	Alternative test and selection (primes do not occur in the times table for <code>N</code> )
<code>apsel=: aprt # N</code>	
<code>pfac=: q:</code>	Prime factors
<code>first=: p:@i.</code>	First primes

### B. Coordinate Geometry

Do experiments on the vector (or *point*) `p=: 3 4` and the triangle represented by the table `tri=: 3 2$ 3 4 6 5 7 2`

`L=: %:@:(+ / )@:*:"1`

Length of vector (See *rank* in the dictionary or in [Section 20 p21](#) of the Introduction)

`LR=: L"1`

Length of rows in table

`disp=: ] - 1&|.`

Displacement between rows

`LS=: LR@disp`

Lengths of sides of figure

`sp=: -:@(+ / )@LS`

Semiperimeter (try `sp tri`)

`H=: %:@(* / )@(sp, sp-LS)`

Heron's formula for area

`det=: -/ . *`

Determinant (See dictionary)

`SA=: det@(.&0.5)`

Signed area. Try `SA@|.`

`sa=: det@(|, .%@!@<:@#)`

General signed volume; try on the tetrahedron

`tet=: ?4 3$9`

as well as on the triangle `tri.`



## 13. Format

A numeric table such as:

```

]t=:(i.4 5)%3
      0 0.333333 0.666667      1 1.33333
1.66667      2 2.33333 2.66667      3
3.33333 3.66667      4 4.33333 4.66667
      5 5.33333 5.66667      6 6.33333

```

can be rendered more readable by *formatting* it to appear with a specified width for each column, and with a specified number of digits following the decimal point.

For example:

```

]f=: 6j2 " : t
0.00 0.33 0.67 1.00 1.33
1.67 2.00 2.33 2.67 3.00
3.33 3.67 4.00 4.33 4.67
5.00 5.33 5.67 6.00 6.33

```

The real part of the left argument of the format function specifies the column width, and the imaginary part specifies the number of digits to follow the decimal point.

Although the formatted table *looks* much like the original table `t`, it is a table of *characters*, not of numbers. For example:

```

$t
4 5

$f
4 30

+ / t
10 11.3333 12.6667 14 15.3333

+ / f
|domain error
|      + / f

```

However, the verb *do* or *execute* (". .) applied to such a character table yields a corresponding numeric table:

```

". . f
  0 0.33 0.67      1 1.33
1.67      2 2.33 2.67      3
3.33 3.67      4 4.33 4.67
  5 5.33 5.67      6 6.33

+/" . f
10 11.33 12.67 14 15.33

```

## Exercises

13.1 Using the programs defined in Section 12, experiment with the following expressions:

```

5j2 ": d=: %: i.12
5j2 ":",.d
fc=: 5j2&":@,.
fc d
20 (fc@h3 ,. h5) d
20 (fc@h3 ,. '|'&,.@h5) d
plot=: fc@h3,.'|'&,.@h5
20 plot d

```

## 14. Partitions

The function `sum=: +/` applies to an entire list argument; to compute *partial sums* or *subtotals*, it is necessary to apply it to each prefix of the argument. For example:

```
sum=: +/
a=: 1 2 3 4 5 6
(sum a) ; (sum\ a)
+---+-----+
|21|1 3 6 10 15 21|
+---+-----+
```

The symbol `\` denotes the *prefix* adverb, which applies its argument (in this case `sum`) to each prefix of the eventual argument. The adverb `\.` applies similarly to suffixes:

```
sum\. a
21 20 18 15 11 6
```

The monad `<` simply *boxes* its arguments, and the verbs `<\` and `<\.` therefore show the effects of partitions with great clarity. For example:

```
<1 2 3
+-----+
|1 2 3|
+-----+
```

```
(<1),(<1 2),(<1 2 3)
+-+---+-----+
|1|1 2|1 2 3|
+-+---+-----+
```

```
<\ a
+-+---+-----+-----+-----+-----+
|1|1 2|1 2 3|1 2 3 4|1 2 3 4 5|1 2 3 4 5 6|
+-+---+-----+-----+-----+-----+
```

```
<\. a
+-----+-----+-----+-----+-----+
|1|1 2|1 2 3|1 2 3 4|1 2 3 4 5|1 2 3 4 5 6|
+-+---+-----+-----+-----+-----+
```

```
| 1 2 3 4 5 6 | 2 3 4 5 6 | 3 4 5 6 | 4 5 6 | 5 6 | 6 |
+-----+-----+-----+-----+-----+-----+
```

The oblique adverb `/.` partitions a *table* along diagonal lines. Thus:

```
</. t=: 1 2 1 */ 1 3 3 1
+---+---+---+---+---+---+
| 1 | 3 2 | 3 6 1 | 1 6 3 | 2 3 | 1 |
+---+---+---+---+---+---+

t ; (sum/. t) ; (10 #. sum/. t) ; (121*1331)
+-----+-----+-----+-----+
| 1 3 3 1 | 1 5 10 10 5 1 | 161051 | 161051 |
| 2 6 6 2 |                  |          |          |
| 1 3 3 1 |                  |          |          |
+-----+-----+-----+-----+
```

## Exercises

- 14.1 Define programs analogous to `sum=: +/\` for progressive products, progressive maxima, and progressive minima.
- 14.1 Treat the following programs and comments like those of Section 12, that is, as exercises in reading and writing. Experiment with expressions such as `c pol x` and `c pp d` and `(c pp d) pol x` with `c=: 1 3 3 1` and `d=: 1 2 1` and `x=: i.5`. See the dictionary or [Section 20 p21](#) for the use of rank:

```
pol=: +/ @ ([*]^i. @ # @ [) "1 0    Polynomial
pp=: +//. @ (*)                      Polynomial product
pp11=: 1 1 & pp                      Polynomial product by 1 1
pp11 d
pp11^:5 (1)
ps=: +/ @ , :                        Polynomial sum
```

## 15. Defined Adverbs

Names may be assigned to adverbs, as they are to nouns and verbs:

```
a=:1 2 3 4 5
prefix=: \
< prefix 'abcdefg'
+--+--+--+--+--+--+--+--+--+--+
|a|ab|abc|abcd|abcde|abcdef|abcdefg|
+--+--+--+--+--+--+--+--+--+--+
```

```
+ / prefix a
1 3 6 10 15
```

Moreover, new adverbs result from a string of adverbs (such as `/\`) and from a conjunction together with one of its arguments, as well as from other trains listed in the dictionary ([Section II F p68](#)). Such adverbs can be *defined* by assigning names. Thus:

```
IP=: /\                               Insert Prefix
+ IP a
1 3 6 10 15
```

```
with3=: &3
% with3 a
0.333333 0.666667 1 1.33333 1.66667
```

```
^ with3 a
1 8 27 64 125
```

```
I=: ^: _1                             Inverse adverb
*: I a
1 1.41421 1.73205 2 2.23607
```

```
+ IP I 1 3 6 10 15
1 2 3 4 5
```

```
ten=: 10&
^. ten 5 10 20 100
0.69897 1 1.30103 2
```

```
#. ten 3 6 5
365
```

```
from=: -~
into=: %~
10 into 17 18 19
1.7 1.8 1.9
```

```
10 from 17 18 19
7 8 9
```

```
i=: "_1
{. i i. 3 4
0 4 8
```

Apply to items

## Exercises

15.1 Experiment with, and explain the behaviour of, the adverbs `pow=: ^&` and `log=: &^.`

15.2 State the significance of the following expressions, and test your conclusions by entering them:

<code>+/~ i=: i. 6</code>	Addition table
<code>ft=: /~</code>	Function table adverb
<code>+ ft i</code>	Addition table
<code>! ft i</code>	Binomial coefficients
<code>inv=: ^:_1</code>	Inverse adverb
<code>sub3=: 3&amp;+ inv</code>	Subtract-three function
<code>sub3 i</code>	

## 16. Word Formation

The interpretation of a written English sentence begins with word formation. The process is based on spaces to separate the sentence into units, but is complicated by matters such as apostrophes and punctuation marks: *was* and *Brown* and *Ross'* are each single units, but *however*, is not (since the comma is a separate unit).

The following lists of characters represent sentences in **J**, and can be executed by applying the *do* or *execute* function " . :

```
m=: '3 %: y. '
d=: 'x. %: y. '
x.=: 4
y.=: 27 4096

" . m
3 16

do=: " .
do d
2.27951 8
```

The word formation rules of **J** are prescribed in [Part I p61](#) of the dictionary. Moreover, the word-formation function ;: can be applied to the string representing a sentence to produce a boxed list of its words:

```
;: m
+---+---+
| 3 | %: | y. |
+---+---+

words=: ;:
words d
+---+---+
| x. | %: | y. |
+---+---+
```

The rhematic rules of **J** apply reasonably well to English phrases:

```
words p=: 'Nobly, nobly, Cape St. Vincent'
```

```

+-----+-----+-----+-----+
|Nobly|,|nobly|,|Cape|St.|Vincent|
+-----+-----+-----+-----+

```

```

>words p
Nobly
'
nobly
'
Cape
St.
Vincent

```

## Exercises

- 16.1 Choose sentences such as `pp=:+//.@(*/)` from earlier exercises, enclose them in quotes, and observe the effects of word-formation (`:` `:`) on them.
- 16.2 The following facility is available under Windows and Macintosh: Move the cursor to the left of a line so that it is separated from the line by one or more spaces, and press Ctrl F1 to display the individually boxed words in the sentence.



## 17. Names and Displays

In addition to the normal names used thus far, there are three further classes:

- 1)  $\$$ : is used for self-reference, allowing a verb to be defined recursively without necessarily assigning a name to it, as illustrated in [Section 22 p23](#).
- 2) The names  $x.$  and  $y.$  are used in explicit definition, discussed in [Section 18 p19](#). They denote the arguments used in explicit definition.
- 3) A name (such as  $ab\_cd\_$ ) that has two underbars of which one is the final character, is a *locative*. Names used in a *locale*  $F$  can be referred to in another locale  $G$  by using the suffix  $F$  in a locative name of the form  $pqr\_F\_$ , thus avoiding conflict with otherwise identical names in the locale  $G$ . See [Section I p71](#) of Part II for further details.

The form of the display invoked by entering a name alone is established by  $9! : 3$ , as described in [Appendix A p223](#). For example:

```
mean=: +/ % #
```

```
9! : 3 (4)
```

Tree form

```
mean
+- / --- +
--+- %
+- #
```

```
9! : 3 (5)
```

Linear form

```
mean
+ / % #
```

Multiple displays are also possible:

```
9! : 3 (5 4 2)
```

```
mean
```

```

+ / % #
  +- / --- +
--+- %
  +- #
+-----+--+
| +-+-+ | % | # | | |
| | + | / | | |
| +-+-+ | | |
+-----+--+

```

## Exercises

17.1 Experiment with the use of locatives.

## 18. Explicit Definition

The sentences in the example:

```
m=: '3 %: y.'
d=: 'x. %: y.'
```

that were executed in the discussion of word formation, can be used with the *explicit definition* conjunction `:` to produce a verb:

```
script=: 0 : 0
3 %: y.
:
x. %: y.
)

roots=: 3 : script
roots 27 4096
3 16

4 roots 27 4096
2.27951 8
```

Adverbs and conjunctions may also be defined explicitly. For example:

```
table=: 1 : 0
[ by ] over x./          Verbs by and over from Section 3 p4
)

2 3 5 ^ table 0 1 2 3
+-+-----+
| |0 1 2 3|
+-+-----+
|2|1 2 4 8|
|3|1 3 9 27|
|5|1 5 25 125|
+-+-----+
```

Control structures may also be used. For example:

```

f=: 3 : 0
if. y.<0
  do. *:y.
  else. %:y.
end. y.=. y.-1
)

f"0 (_4 4)
16 2

factorial=: 3 : 0
a=. 1
while. y.>1
  do. a=.: a*y.
end. a
)

factorial"0 i. 6
1 1 2 6 24 120

```

## Exercises

- 18.1 Experiment with and display the functions `roots=: 3 : script` and `13 : script` (which are equivalent).
- 18.2 See the discussion of control structures in the dictionary, and use them in defining further verbs.
- 18.3 Experiment with expressions such as `! d b=: i.7`, after defining the adverb `d` :
- ```

d=: 1 : 0
+:@x.
)

```
- 18.4 Using the program `pol` from Exercise 14.2, perform the following experiments and comment on their results:

```

g=: 11 7 5 3 2 & pol
e=: 11 0 5 0 2 & pol
o=: 0 7 0 3 0 & pol
(g = e + o) b=: i.6
(e = e@-) b
(o = -@o@-) b

```

Answer: The function `g` is the sum of the functions `e` and `o`. Moreover, `e` is an even function (whose graph is reflected in the vertical axis), and `o` is an odd function (reflected in the origin).

18.5 Review [Section H p70](#) of Part II and use scripts to make further explicit definitions.

18.6 Enter the following explicit definition of the adverb `even` and perform the suggested experiments with it, using the functions defined in the preceding exercise:

```
even=: 2 : 0
-: @(x.f. + x.f.@-)
)
ge=: g even
(e = ge) b
(e = e even) b
```

18.7 Define an adverb `odd` and use it in the following experiments:

```
exp=: ^
sinh=: 5&o.
cosh=: 6&o.
(sinh = exp odd) b
(sinh = exp .: -) b
(cosh = exp even) b
(exp = exp even + exp odd) b
```

The primitive odd adverb .: -

18.8 The following experiments involve complex numbers, and should perhaps be ignored by anyone unfamiliar with them:

```
sin=: 1&o.
cos=: 2&o.
(cos = ^@j. even) b
(j.@sin = ^@j. odd) b
```

## 19. Tacit Equivalents

Verbs may be defined either explicitly or tacitly. In the case of a one-sentence explicit definition, of either a monadic or dyadic case, the corresponding tacit definition may be obtained by using the adverb `13 :` as illustrated below. First enter `9! : 3 ] 2 5` so as to obtain both the boxed and linear displays of verbs:

```
s=: 0 : 0
(+/y.) % (#y.)
)
```

```
mean=: 3 : s
MEAN=: 13 : s
```

| mean                   | MEAN         |
|------------------------|--------------|
| +--+-----+             | +-----+--+   |
| 3   :   (+/y.) % (#y.) | +--+   %   # |
| +--+-----+             | +   /        |
| 3 : ' (+/y.) % (#y.) ' | +--+         |
|                        | +-----+--+   |
|                        | + / % #      |

The explicit form of definition is likely to be more familiar to computer programmers than the tacit form. Translations provided by the adverb `13 :` may therefore be helpful in learning tacit programming.

### Exercises

- 19.1 Use the display of the tacit definition of `MEAN` to define an equivalent function called `M`.

Answer: `M=: +/ % #`

## 20. Rank

The shape (\$), tally (#), and rank (#@\$), of a noun are illustrated by the noun `report`, which may be construed as a report covering two years of four quarters of three months each:

```

      ]report=: i. 2 4 3
0   1   2
3   4   5
6   7   8
9  10  11

12  13  14
15  16  17
18  19  20
21  22  23

```

```

      ($ ; # ; #@$) report           Shape, Number of items, Rank
+-----+---+
| 2 4 3 | 2 | 3 |
+-----+---+

```

The last  $k$  axes determine a  $k$ -cell of a noun; the 0-cells of `report` are the atoms (such as 4 and 14), the 1-cells are the three-element quarterly reports, and the two-cells (or *major cells* or *items*) are the two four-by-three yearly reports.

The rank conjunction `"` is used in the phrase `f" k` to apply a function `f` to each of the  $k$ -cells of its argument. For example:

```

      ,report
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

      ,"2 report
0  1  2  3  4  5  6  7  8  9 10 11
12 13 14 15 16 17 18 19 20 21 22 23

      <@i. s=: 2 5
+-----+
| 0 1 2 3 4 |
| 5 6 7 8 9 |

```

```

+-----+
  <@i."0 s
+-----+
| 0 1|0 1 2 3 4|
+-----+

```

Both the left and right ranks of a dyad may be specified. Thus:

```

      10 11 12 (,"0 1 ; , "1 1 ; , "1) 0 1 2
+-----+-----+-----+
10 0 1 2	10 11 12 0 1 2	10 11 12 0 1 2
11 0 1 2		
12 0 1 2		
+-----+-----+-----+

```

The *basic characteristics* adverb `b.` is very useful in analyzing functions (or expressions that define them) with respect to their ranks. For example:

```

      (# b. 0) ; (+/\ b. 0) ; (+/\ % #) b. 0
+-----+-----+-----+
|_ 1 _|_ 0 _|_ _ _|
+-----+-----+-----+

```

## Exercises

20.1 Observe the results of the following uses of the monads produced by the rank conjunction, and comment on them:

```

a=: i. 3 4 5
<"0 a
<"1 a
<"2 a
<"3 a
< a
<"_1 a
<"_2 a
mean=: +/ % #
mean a
mean"1 a
mean"2 a

```



Answer:  $\langle k$  applies  $\langle$  to each cell of rank  $k$ , with  $\langle(\#\$a)$   $a$  being equivalent to  $\langle a$ . Moreover, a negative value of  $k$  specifies a *complementary* rank that is effectively  $|k|$  less than the rank of the argument  $a$ .

- 20.2 Use the results of the following experiments to state the relation between the conjunctions  $@$  (*Atop*) and  $@:$  (*At*), and compare your conclusions with the dictionary definition:

```
(g=: <"2) a=: i. 3 4 5
|. @: g a
|. @ g a
|: @: (<"1) a
|: @ (<"1) a
```

Answer: The rank of the function  $|. @: g$  is itself infinite and  $|. @:$  therefore applies to the entire list result of  $g a$ , consequently reversing it. On the other hand, the function  $f @ g$  *inherits* the rank of  $g$ , and  $|. @$  therefore applies individually to the atoms produced by  $g$ , producing no effect.

- 20.3 Use the results of the following experiments to comment on the use of the rank conjunction in dyads:

```
b=: 'ABC'
c=: 3 5 $ 'abcdefghijklmno'
c
b,c
b , "0 1 c
b , "1 1 c
b , "1 c
```

## 21. Gerund and Agenda

In English, a *gerund* is a noun that carries the force of a verb, as does the noun *cooking* in the phrase *the art of cooking*; and *agenda* is a list of items for action. The *tie* conjunction ``` applies to two verbs to form a gerund, from which elements can be chosen for execution. Thus, if the agenda conjunction `@.` is applied to a gerund, its verb right argument provides the results that choose the elements. For example:

```

g=: +`^
a=: <
2 a 3
1

2 g@.a 3
8

3 g@.a 2
5

+:`-:~*:`%: @. (4&|@<.)"0 i. 10
0 0.5 4 1.73205 8 2.5 36 2.64575 16 4.5

```

The verb produced by `g@.a` is often called a *case* or *case statement*, since it selects one of the "cases" of the gerund for execution.

The *insert* adverb `/` applies to a gerund in a manner analogous to its application to a verb. For example:

```

c=: 3 [ x=: 4 [ power=: _1
g/ c,x,power
3.25

3+x^_1
3.25

```

The elements of the gerund are repeated as required. For example:

```

+`*/1,x,3,x,3,x,1

```

The last sentence above corresponds to Horner's efficient evaluation of the polynomial with coefficients  $1 \ 3 \ 3 \ 1$  and argument  $x$ .

### Exercises

21.1 Define a function `f` such that `(x=: 4) f c=: 1 3 3 1` yields the result used as the argument to `+`*/` in Horner's method.

Answer: `f=: } .@, @, .`

## 22. Recursion

The factorial function  $!$  is commonly defined by the statement that factorial  $n$  is  $n$  times factorial  $n-1$ , and by the further statement that factorial  $0$  is  $1$ . Such a definition is called *recursive*, because the function being defined recurs in its definition.

A case statement can be used to make a recursive definition, the case that employs the function under definition being chosen repeatedly until the terminating case is encountered. For example:

```
factorial=: 1: `( ) * factorial @<: ) @. *
factorial "0 i.6
1 1 2 6 24 120
```

Note that  $1:$  denotes the constant function whose result is  $1$ .

In the sentence `(sum=: +/) i.5` the verb defined by the phrase `+/` is assigned a name before being used, but in the sentence `+/ i.5` it is used anonymously. In the definition of `factorial` above, it was essential to assign a name to make it possible to refer to it within the definition. However, the word `$:` provides *self-reference* that permits anonymous recursive definition. For example:

```
1: `( ) * $: @<: ) @. * "0 i. 6
1 1 2 6 24 120
```

In the Tower of Hanoi puzzle, a set of  $n$  discs (each of a different size) is to be moved from post A to post B using a third post C and under the restriction that a larger disc is never to be placed on a smaller. The following is a recursive definition of the process:

```
h=: b` (p, .q, .r) @. c
c=: 1: < [
b=: 2&, @ [ $ ]
p=: <: @ [ h 1: A. ]
q=: 1: h ]
r=: <: @ [ h 5: A. ]
```

```

3 h x=: 'ABC'
AABACCA
BCCBABB

```

```

0 1 2 3 4 <@h"0 1 x
++-+---+-----+-----+
| | A | AAC | AABACCA | AACABBAACCBACAAC |
| | B | CBB | BCCBABB | CBBCACCBBAABCBB |
++-+---+-----+-----+

```

## Exercises

22.1 Use the following as exercises in reading and writing:

```

f=:1: `( +//. @ ( , : ~ ) @ ( $ : @ < : ) ) @ . *
<@f"0 i.6
g=:1: `( ( [ , +/@( _2&{ . ) ) @ $ : @ < : ) @ . *

```

Binomial Coeffs  
Boxed binomials  
Fibonacci

## 23. Iteration

The repetition of a process, or of a sequence of similar processes, is called *iteration*. Much iteration is implicit, as in  $a/b$  and  $a^*/b$ , and  $a*b$ ; explicit iteration is provided by the power conjunction ( $^:$ ), by agenda ( $@.$ ) with self-reference, and by control structures:

```
(cos=: 2&o.) ^: (i.6) b=: 1
1 0.540302 0.857553 0.65429 0.79348 0.701369

]y=: cos^:_ b
0.739085

y=:cos y
1
```

The example `cos^:_` illustrates the fact that infinite iteration is meaningful (that is, terminates) if the process being applied converges.

*Controlled* iteration of a process  $p$  is provided by  $p^:q$ , where the result of  $q$  determines the number of applications of  $p$  performed. The forms  $q:@p^:q$  and  $p^:q^:_$  are commonly useful.

If  $f$  is a continuous function, and if  $f\ i$  and  $f\ j$  differ in sign, then there must be a *root*  $r$  between  $i$  and  $j$  such that  $f\ r$  is zero; the list  $b=:i,j$  is said to *bracket* a root. A narrower bracket is provided by the mean of  $b$  together with that element of  $b$  whose result on applying  $f$  differs in sign from its result. Thus:

```
f=: %: - 4:                                A sample function

root=: 3 : 0
m=. +/ % #
while. ~:/y.
do.
  if. ~:/ * f ({.,m) y.
    do. y.=. ({.,m) y.
  else. y.=. ({:,m) y.
  end.
end. m y.
```

```
)
    b=: 1 32
    root b
16

    f b,root b
_3 1.65685 1.77636e_15
```

## Exercises

- 23.1 Use the function `root` to find the roots of various functions, such as `f=: 6&-@!`
- 23.2 Experiment with the function `fn=: +/\` (which produces the figurate numbers when applied repeatedly to `i.n`), and explain the behaviour of the function `fn^:(?@3:)`

## 24. Trains

The train of nouns in the English phrase *Ontario museum Egyptian collection* represents a single noun. Similarly, the fork discussed in [Section 5 p6](#) and its exercises permit the use of arbitrarily long trains of verbs to produce a verb.

[Section 15 p16](#) introduced the use of trains of adverbs, and of conjunctions together with nouns or verbs, to represent adverbs. *Conjunctions* may also be produced by trains of adverbs and conjunctions in a manner analogous to forks.

For example, the case diagrammed on the right below can be used as follows:

```

cj=: \@ \

      < cj (+/) a=: i.3 3
+-----+-----+-----+
0 1 2	0 1 2	0 1 2
	3 5 7	3 5 7
		9 12 15
+-----+-----+-----+

      ( < \ ) @ ( + / \ ) a
+-----+-----+-----+
0 1 2	0 1 2	0 1 2
	3 5 7	3 5 7
		9 12 15
+-----+-----+-----+

      c
      / \
    a1 a2
      |  |
      x  y

      ( * / ) cj (+/) a
      0 1 2
      0 5 14
      0 60 210

```

The explicit form of defining conjunctions treated in the exercises of [Section 18 p19](#) can be used to produce an equivalent conjunction `CJ` as shown below.

```

s=: 0 : 0
(x.\)@(y.\)
)

CJ=: 2 : s
      ( * / ) CJ (+/) a
0 1 2
0 5 14

```



0 60 210

## 25. Permutations

Anagrams are familiar examples of the important notion of *permutations*:

```
w=: 'STOP'
3 2 0 1 { w
POST
```

```
2 3 1 0 { w
OPTS
```

```
3 0 2 1 { w
PSOT
```

The left arguments of `{` above are themselves permutations of the list `1.4`; examples of *permutation vectors*, used to represent permutation functions in the form `p&{`.

If `p` is a permutation vector, the phrase `p&C.` also represents the permutation `p&{`. However, other cases of the *cycle* function `C.` are distinct from the *from* function `{`. In particular, `C. p` yields the *cycle representation* of the permutation `p`. For example:

```
]c=: C. p=: 2 4 0 1 3
+---+-----+
| 2 0 | 4 3 1 |
+---+-----+
```

```
c C. 'ABCDE'
CEABD
```

```
C. c
2 4 0 1 3
```

Each of the boxed elements of a cycle specify a list of positions that cycle among themselves; in the example above, the element from position 3 moves to position 4, element 1 moves to 3, and element 4 to 1.

A permutation can be identified by its index in the table of all  $n!$  permutations of order  $n$  listed in increasing order. This index is the *anagram index* of the permutation; the corresponding permutation is effected by the function `A.` as illustrated below:

```
1 A. 'ABCDE'
ABCED
```

```
(i.!3) A. i.3
0 1 2
0 2 1
1 0 2
1 2 0
2 0 1
2 1 0
```

```
A. 0 1 2 4 3
1
```

```
(i.!3) A. 'ABC'
ABC
ACB
BAC
BCA
CAB
CBA
```

## Exercises

- 25.1 Use the following as exercises in reading and writing (try on `a=: 'abcdef'` and `b=: i. 6` and `c=: i. 6 6`):

|                                 |                                                       |
|---------------------------------|-------------------------------------------------------|
| <code>f=: 1&amp;A.</code>       | Interchange last two items                            |
| <code>g=: 3&amp;A.</code>       | Rotate last three items                               |
| <code>h=: 5&amp;A.</code>       | Reverse last three items                              |
| <code>i=: &lt;:@!@[ A. ]</code> | <code>k i a</code> reverses last <code>k</code> items |

- 25.2 Experiment with the following expressions and others like them to determine the rules for using abbreviated arguments to `C.` and compare your conclusions with the dictionary definitions:

```
2 1 4 C. b=:i.6
(<2 1 4) C. b
(3 1;5 0) C. b
```

- 25.3 Make a program `ac` that produces a table of the cycle representations of all permutations of the order of its argument, as in `ac 3`.

Answer: `ac=: C.@(i.@! A. i.)`

## 26. Linear Functions

A function  $f$  is said to be *linear* if  $f(x+y)$  equals  $(f\ x)+(f\ y)$  for all arguments  $x$  and  $y$ . For example:

```
f=: 3&|. @: +: @: |.
]x=: i.# y=: 2 3 5 7 11
0 1 2 3 4
```

```
x+y
2 4 7 10 15
```

```
f x+y
8 4 30 20 14
```

```
(f x),:(f y)
2 0 8 6 4
6 4 22 14 10
```

```
(f x)+(f y)
8 4 30 20 14
```

A linear function can be defined equivalently as follows:  $f$  is linear if  $f@:+$  and  $+&f$  are equivalent. For example:

```
x f@:+ y x +&f y
8 4 30 20 14 8 4 30 20 14
```

If  $f$  is a linear function, then  $f\ y$  can be expressed as the *matrix product*  $mp\&M$   $y$ , where

```
mp=: +/ . *
M=: f I=: = i.#y          I is an identity matrix

mp&M y
6 4 22 14 10
f y
6 4 22 14 10
```

Conversely, if  $m$  is any square matrix of order  $\#y$ , then  $m\&mp$  is a linear function on  $y$ , and if  $m$  is invertible, then  $(\%.m)\&mp$  is its inverse:

```
x=: 1 2 3 [ y=: 2 3 5
]m=: ? 3 3$9
5 7 3
```

```
7 2 3
4 4 2
```

```
]n=: %.m
_1.33333 _0.333333 2.5
_0.333333 _0.333333 1
3.33333 1.33333 _6.5
```

```
g=: mp&m
h=: mp&n
```

```
x g@:+ y
82 63 40
```

```
x +&g y
82 63 40
```

```
g h y
2 3 5
```

## Exercises

- 26.1 For each of the following functions, determine the matrix  $M$  such that  $M$  ( $mp=: +/ . *$ )  $N$  is equivalent to the result of the function applied to the matrix  $N$ , and test it for the case  $N=: i. 6 6$

```
| .
-
+:
(4&*-2&*@| .)
2&A.
```

## 27. Obverse and Under

The result of  $f^{\wedge}:_1$  is called the *obverse* of the function  $f$ ; if  $f=:g:.h$ , this obverse is  $h$ , and it is otherwise an inverse of  $f$ . Inverses are provided for over 25 primitives (including the case of the square root illustrated in [Section 11 p12](#)), as well as many bonded dyads such as  $-&3$  and  $10\&^.$  and  $2\&o..$

Moreover,  $u@v^:_1$  is given by  $(v^:_1)@(u^:_1)$ . For example:

```
fFc=: (32&+ )@(*&1.8)
]b=:fFc _40 0 100
_40 32 212
```

```
cFf=: fFc^:_1
cFf b
_40 0 100
```

The result of the phrase  $f \& . g$  is the verb  $(g^:_1)@(f \& g)$ . The function  $g$  can be viewed as *preparation* (which is done before and undone after) for the application of the "main" function  $f$ . For example:

```
b=: 0 0 1 0 1 0 1 1 0 0 0
sup=: </\
sup b
0 0 1 0 0 0 0 0 0 0 0
```

Suppress ones *after* the *first*

```
|. sup |. b
0 0 0 0 0 0 0 1 0 0 0
```

Suppress ones *before* the *last*

```
sup&.|. b
0 0 0 0 0 0 0 1 0 0 0
```

```
3 +&.^. 4
12
```

Multiply by applying the exponential  
to the sum of logarithms

```
(^.3)+(^.4)
2.48491
```

```
^ (^ .3)+(^.4)
12
```

```

]c=: 1 2 3;4 5;6 7 8
+-----+-----+-----+
| 1 2 3|4 5|6 7 8|
+-----+-----+-----+

```

```

|. &. > c
+-----+-----+-----+
| 3 2 1|5 4|8 7 6|
+-----+-----+-----+

```

Open, reverse, and then box

## Exercises

27.1 Use the following as exercises in reading and writing. Try using arguments such as `a=: 2 3 5 7` and `b=: 1 2 3 4` and `c=: <@i."0 i. 3 4`:

```

f=: +&.^
g=: +&.(10&^. )
h=: *&.^
i=: |.&.>
j=: +/&.>
k=: +/&>

```

Multiplication by addition of natural logs  
 Multiplication using base-10 logs  
 Addition from multiplication  
 Reverse each box  
 Sum each box  
 Sum each box and leave open

## 28. Identity Functions and Neutral

The monads  $0\&+$  and  $1\&*$  are *identity* functions, and  $0$  and  $1$  are said to be *identity elements* or *neutrals* of the dyads  $+$  and  $*$  respectively. Insertion on an empty list yields the neutral of the dyad inserted. For example:

|     |         |     |        |     |                |
|-----|---------|-----|--------|-----|----------------|
| $0$ | $+/i.0$ | $0$ | $+/''$ | $0$ | $+/0\{. 2 3 5$ |
| $1$ | $*/i.0$ | $1$ | $*/''$ | $1$ | $*/0\{. 2 3 5$ |

These results are useful in partitioning lists; they ensure that certain obvious relations continue to hold even when one of the partitions is empty. For example:

```
+/ a=: 2 3 5 7 11
28
```

```
(+/4{.a)+(+/4}.a)
28
```

```
(+/0{.a)+(+/0}.a)
28
```

```
*/a
2310
```

```
(*/4{.a)*(*/4}.a)
2310
```

```
(*/0{.a)*(*/0}.a)
2310
```

The identity functions and other basic characteristics of functions (such as *rank*) are given by the adverb  $b.$ , as illustrated below:

```
^ b. _1      Inverse
^.
```

```
^ b. 0      Ranks
_ 0 0
```



`^ b. 1`  
`$&1@ ( } . @$ )`

Identity function

## Exercises

28.1

Predict and test the results of the following expressions:

```
* / ' '
< . / ' '
> . / ' '
> . / 0 4 4 $ 0
+ / . * / 0 4 4 $ 0
1 2 3 4 +& . ^ . / 5 6 7 8
```

28.2

Experiment with the dyad `{@;` and give the term used to describe it in mathematics.

Answer: Cartesian product

28.3

Test the assertion that the monads `(%:@~. +/ . * =)` and `%:` are equivalent, and state the utility of the former when applied to a list such as `1 4 1 4 2` that has repeated elements.

Answer: The function `%:` (which could be a function costly to execute) is applied only to the distinct elements the argument (as selected by the *nub* function `~.`).

28.4

Comment on the following experiments before reading the comments on the right:

|                 |                     |
|-----------------|---------------------|
| a=: 2 3 5 [ b=: |                     |
| 1 2 4           |                     |
| a (f=: *: @+) b | Square of sum       |
| a (g=: +&*: +   | Sum of              |
| +: @*) b        | squares plus        |
|                 | double              |
|                 | product             |
| a (f=g) b       | Expression of       |
|                 | the identity of     |
|                 | the functions       |
| a (f=:g) b      | f and g in a        |
|                 | <i>tautology</i>    |
|                 | (whose result       |
|                 | is                  |
| taut=: f=:g     | <i>always</i> true; |
|                 | that is, 1).        |

28.5

A phrase such as  $f=:g$  may be a tautology for the dyadic case only, for the monadic case only, or for both. Use the following tautologies as reading and writing exercises, including statements of applicability (Dyad only, etc.):

t1=: >: -: > +. =

(Dyad only) The primitive  $>:$  is identical to greater than *or* equal

t2=: <. -: -@>.&-

(Both) Lesser-of is neg on greater-of on neg; Floor is neg on ceiling on neg

t3=: <. -: >.&.-

Same as t2 but uses *under*

t4=: \*: @>: -: \*: + +: + 1:

(Monad) Square of  $a+1$  is square of  $a$  plus twice  $a$  plus 1

t5=: \*: @>: -: #.&1 2 1"0

Same as t4 using polynomial

t6=: ^&3 @>: -: #.&1 3 3 1"0

Like t5 for cube

bc=: i. @>: ! ]

Binomial coefficients

t7=: (>: @)^([) -: ([ #. bc @ ([) "0

Like t6 with  $k$  &  $t7$  for  $k$ th power

s=: 1&o.

Sine

c=: 2&o.

Cosine

t8=: s@+-: (s@[\*c@])+(c@[\*s@])

(Dyad) Addition and Subtraction

t9=: s@--: (s@[\*c@])-(c@[\*s@])

Formulas for sine

det=: -/ . \*

Determinant

perm=: +/ . \*

Permanent

sct=: 1 2&o."0@ ( , "0)

Sine and cosine tables

t10=: s@- -: det@sct

Same as t9 but using the determinant of the sin and cos table

t11=: s@+ -: perm@sct

Like t8 using the permanent

S=: 5&o.

Hyperbolic sine

C=: 6&o.

Hyperbolic cosine

SCT=: 5 6&o."0@ ( , "0)

Sinh and Cosh table

t12=: S@+ -: perm@SCT

Addition theorem for sinh

SINH=: ^ . -: -

Odd part of exponential

COSH=: ^ . . -: -

Even part of exponential

t13=: SINH -: S

Sinh is odd part of exponential

t14=: COSH -: C

Cosh is the even part of exponential

sine=: ^&.j. . -: -

Sine is the odd part of exponential

t15=: sine -: s

under multiplication by  $0j1$

28.6 Comment on the following expressions before reading the comments on the right:

```
g=: + > >.
```

```
5 g 2
```

```
5 g _2 _1 0 1 2
```

```
f=: *.&(0<)
```

```
theorem=: f <: g
```

```
5 theorem _2 _1 0 1 2
```

Test if sum exceeds maximum

True for positive arguments

but not true in general

Test if both arguments exceed 0

The truth value of the result of `f` does not exceed that of `g`. This may also be stated as "if `f` (is true) then `g` (is true)" or as "`f` implies `g`"

## 29. Secondaries

It is convenient to supplement the *primitives* or *primaries* provided in a language by *secondaries* whose names belong to an easily recognized class. The following examples use names beginning with a capital letter:

|        |                                        |                                     |
|--------|----------------------------------------|-------------------------------------|
| Ad=:   | [ 0: } -@>: @\$@ ] { . ]               | Append diagonal scalar              |
| Ai=:   | >: @i .                                | Augmented integers                  |
| Area=: | [ : Det ] , . % @ ! @ # " 1            | Area (Vol) try Area tet=: 0, =i.3   |
| Bc=:   | i . ! / i .                            | Binomial coefficients               |
| Bca=:  | % . @Bc                                | Binomial coefficients (alternating) |
| By=:   | ' ' & i @ , . @ [ , . ]                | By (format; see Ta)                 |
| Cpa=:  | ] % . i . @ # @ ] ^ / Ei @ [           | Coeffs of poly approx               |
| CPA=:  | ( @ ) ] % . i . @ # @ ] ^ / Ei @ [     | Coeffs of poly approx (adverb)      |
| Det=:  | - / . *                                | Determinant                         |
| Dpc=:  | 1: } . ] * i . @ #                     | Differentiate poly coeffs           |
| Dl=:   | ( " 0 ) ( D . 1 )                      | Derivative (scalar, first)          |
| Ei=:   | i . @ ( + * + 0 & = )                  | Extended integers                   |
| Epc=:  | Bc @ # X ]                             | Expand poly coeffs                  |
| Ipc=:  | 0: , ] % Ai @ #                        | Integrate poly coeffs               |
| Inv=:  | ^: _1                                  | Inverse                             |
| Id=:   | = @ i .                                | Identity matrix                     |
| Mat=:  | -: /: ~                                | Monotone ascending test             |
| Mdt=:  | -: \: ~                                | Monotone descending test            |
| Mrg=:  | + & \$ { . , @ (   : @ , : )           | Merge                               |
| Over=: | ( { . ; } . ) @ " : @ ,                | Over (format; see Ta)               |
| Pad=:  | 2 : ' x . % . ] ^ / Ei @ ( y . " _ ) ' | Polynomial approx of degree         |
| Pp=:   | + / / . @ ( * / )                      | Polynomial coeffs product           |
| Si=:   | ( Ei @ +: -   ) : ( - / i . )          | Symmetric and subsiding int         |

|        |                           |                              |
|--------|---------------------------|------------------------------|
| Span=: | 2 : 'y."_ x.\ ]'          | Span of apply of left arg    |
| S1=:   | :@ @(^!._1/~%.^/~)@i.     | Stirling numbers (1st kind)  |
| S2=:   | :@ (^/~%.^!._1/~)@i.      | Stirling numbers (2nd kind)  |
| Ta=:   | 1 : '[By]Over x./'        | Table adverb                 |
| Thr=:  | ] * 0.1&^@[ <:  @]        | Threshold for non-zero       |
| Tile=: | \$@]{.[\$~\$@]+2: 1:+\$@] | Tile (try 0 1 Tile i. 2 3 4) |
| X=:    | +/. *                     | Times (matrix product)       |
| XA=:   | -/. *                     | Times (alternating)          |

## Exercises

29.1 Enter the definitions of the secondaries (or at least those used in these exercises), and then enter the following expressions:

```
(Ai 2 3);(Ai 2 _3);(Ei 2 3);(Ei 2 _3)
(i.;i.@-;Ai;Ai@-;Ei;Ei@-) 4
(Si 4);(7 4 Si 4)
+Ta~@i. 4
(S1;S2) 7
(];X/;%/;%./;(%./%{.)) y=: (Bc ,: Bca) 5
(0 1&Cb;1 _1&Cb) i. 2 3 4
```

29.2 Perform further experiments with the secondaries.

## Sample Topics

This part provides examples of the use of **J** in various topics; it is designed to be used in conjunction with the dictionary and at the keyboard of a **J** system. It is also designed to be used *inductively*, as follows:

- Read one or two sentences and their results (which begin at the left margin), and attempt to state clearly in English what each sentence does.
- Enter similar sentences to test the validity of your statements.
- Consult the dictionary to confirm your understanding of the meaning of primitives such as `i .` (used with one argument or two). Use the Vocabulary at the end of the book as an index to pages in the dictionary. In the Windows system, highlight a word (e.g. `/:`) and hit ctrl-F1 to display the dictionary entry for that word.
- Enter *parts* of a complex sentence, such as `i . 26` and `j+ / i . 26` in the case of `a . { ~j+ / i . 26` used in the topic [Alphabet and Numbers p33](#).

# 1. Spelling

```

    phr=: 'index=: a.i.' 'aA' ''
    ;:phr
+-----+---+---+---+---+
|index|=: |a.| |i.| 'aA' |
+-----+---+---+---+---+

$ ;:phr
5

> ;:phr
index
=:
a.
i.
'aA'

(do=: ".) phr
97 65

do 'abc =: 3 1 4 2'
3 1 4 2

abc
3 1 4 2

```



## 2. Alphabet and Numbers

```
(a. {~ j +/ i.26) ; (j +/ i.6) ; (j=: a. i. 'aA') ; ($ a.)
+-----+-----+-----+-----+
|abcdefghijklnopqrstuvwxy|97 98 99 100 101 102|97 65|256|
|ABCDEFGHIJKLMNopqrstuvwxy|65 66 67 68 69 70|    |    |
+-----+-----+-----+-----+
```

```
1 2 3{ t=: 8 32$a.          The major alphabet
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNopqrstuvwxy[ \ ] ^ _
`abcdefghijklnopqrstuvwxy{|}~•
```

The table `t` arranges the alphabet in eight rows, but its complete display would look odd because of the effects of various control characters such as the carriage return. A boxed display `<t` is more readable because spaces are substituted for them.

```
i. 2 5                      Table of integers
0 1 2 3 4
5 6 7 8 9
```

```
r=: 0j1 _1 0j1 1          Square roots of plus and minus 1
+ r                          (Complex) conjugates
0j1 _1 0j1 1
```

```
r * +r
1 1 1 1
```

```
r */ r                      Multiplication table of roots of unity
_1 0j1 1 0j1
0j1 1 0j1 _1
1 0j1 _1 0j1
0j1 _1 0j1 1
```

```
! 45x                      "x" denotes extended precision
119622220865480194561963161495657715064383733760000000000
```

### 3. Grammar

```

    fahrenheit =: 50
    (fahrenheit - 32) * 5%9
10

    prices =: 3 1 4 2
    orders =: 2 0 2 1
    orders * prices
6 0 8 2

    +/ orders * prices
16

    +/ \ 1 2 3 4 5
1 3 6 10 15

    2 3 * / 1 2 3 4 5
2 4 6 8 10
3 6 9 12 15

    (cube=: ^&3) i. 9
0 1 8 27 64 125 216 343 512

```

#### Parts Of Speech

|               |                |
|---------------|----------------|
| 50 fahrenheit | Nouns/Pronouns |
| + - * % cube  | Verbs/Proverbs |
| / \           | Adverbs        |
| &             | Conjunction    |
| =:            | Copula         |
| ( )           | Punctuation    |

## 4. Function Tables

Just as the behaviour of *addition* is made clear by addition tables in elementary school, so the behaviour of other verbs (or *functions*) can be made clear by function tables.

The next few pages show how to make function tables, and how to use the *utility* functions `over` and `by` to border them with their arguments to make them easier to interpret.

Study the tables shown, and make tables for other functions (such as `<`, `<.` and `%`) suggested by the Vocabulary.

| ( + / ~ ; * / ~ ) 0 1 2 |   |   |   |   |   |  | Addition and Times tables |  |
|-------------------------|---|---|---|---|---|--|---------------------------|--|
| +-----+-----+           |   |   |   |   |   |  |                           |  |
| 0                       | 1 | 2 | 0 | 0 | 0 |  |                           |  |
| 1                       | 2 | 3 | 0 | 1 | 2 |  |                           |  |
| 2                       | 3 | 4 | 0 | 2 | 4 |  |                           |  |
| +-----+-----+           |   |   |   |   |   |  |                           |  |

| ^/ ~ i. 4 |   |   |    |  | Power table |  |
|-----------|---|---|----|--|-------------|--|
| 1         | 0 | 0 | 0  |  |             |  |
| 1         | 1 | 1 | 1  |  |             |  |
| 1         | 2 | 4 | 8  |  |             |  |
| 1         | 3 | 9 | 27 |  |             |  |

| + . / ~ 0 1 |   |  | Or table |  |
|-------------|---|--|----------|--|
| 0           | 1 |  |          |  |
| 1           | 1 |  |          |  |

## 5. Bordering a Table

```
over=: ({. ; }. )@":@,
by=: ' ' & ; @, .@[ , .]
```

*Utility* functions, intended for use  
rather than for immediate study

```
primes=: 2 3 5
i=: 0 1 2 3 4
```

```
primes by i over primes */ i
```

```
+---+-----+
| |0 1  2  3  4|
+---+-----+
2	0 2  4  6  8
3	0 3  6  9 12
5	0 5 10 15 20
+---+-----+
```

```
tba=: 1 : '[ by ] over x./'
```

Table adverb

```
primes * tba i
```

```
+---+-----+
| |0 1  2  3  4|
+---+-----+
2	0 2  4  6  8
3	0 3  6  9 12
5	0 5 10 15 20
+---+-----+
```

```
7 11 ^ tba i
```

```
+---+-----+
| |0 1  2  3  4|
+---+-----+
| 7|1  7  49  343  2401|
|11|1 11 121 1331 14641|
+---+-----+
```

## 6. Tables (Letter Frequency)

```

text=: ' i sing of olaf glad and big'
alph=: ' abcdefghijklmnopqrstuvwxyz'
10{.alph=/text
1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0
0 1 0 0
0 0
0 1 0 0 0 1 0 0 0
0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0
0 1 0 0 1 0 1 0

```

```

'01'{~10{.alph=/text
1010000100100001000010001000
00000000000000100001001000000
0000000000000000000000000100
0000000000000000000000000000
00000000000000000000100010000
0000000000000000000000000000
0000000001000010000000000000
00000010000000000100000000001
0000000000000000000000000000
0100100000000000000000000010

```

```

]LF=: 2 13 $ +/"1 alph =/ text
7 3 1 0 2 0 2 3 0 3 0 0 2
0 2 2 0 0 0 1 0 0 0 0 0 0

```

Letter frequency table

```

+ / + / LF
28

```

```

$text
28

```

## 7. Tables

```
div=: 0=rem=: i | /i=: i.7
(, .i) ; rem ; div
```

Divisibility and remainder tables

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 2 | 3 | 4 | 5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

```
(i#~2=+/div);(2=+/div);(+div)
```

Primes, test, # of divisors

| 2 | 3 | 5 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 7 | 1 | 2 | 2 | 3 | 2 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 3 | 5 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 7 | 1 | 2 | 2 | 3 | 2 | 4 |

```
(=/~ ; </~ ; <:/~; ~:/~) i. 5
```

| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

```
t=: (/ ~) (@i.)
```

Table adverb

```
(=t;<t;<:t;~:t) 5
```

| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

```
(^t;!t;-t) 5
```

| +-----+ |   |    |    |     | +-----+ |   |   |   |   | +-----+ |    |    |    |    | +-----+ |  |  |  |  |
|---------|---|----|----|-----|---------|---|---|---|---|---------|----|----|----|----|---------|--|--|--|--|
| 1       | 0 | 0  | 0  | 0   | 1       | 1 | 1 | 1 | 1 | 0       | _1 | _2 | _3 | _4 |         |  |  |  |  |
| 1       | 1 | 1  | 1  | 1   | 0       | 1 | 2 | 3 | 4 | 1       | 0  | _1 | _2 | _3 |         |  |  |  |  |
| 1       | 2 | 4  | 8  | 16  | 0       | 0 | 1 | 3 | 6 | 2       | 1  | 0  | _1 | _2 |         |  |  |  |  |
| 1       | 3 | 9  | 27 | 81  | 0       | 0 | 0 | 1 | 4 | 3       | 2  | 1  | 0  | _1 |         |  |  |  |  |
| 1       | 4 | 16 | 64 | 256 | 0       | 0 | 0 | 0 | 1 | 4       | 3  | 2  | 1  | 0  |         |  |  |  |  |
| +-----+ |   |    |    |     | +-----+ |   |   |   |   | +-----+ |    |    |    |    | +-----+ |  |  |  |  |

## 8. Classification

Classification is a familiar notion. For example, the classification of letters of the alphabet as *vowel*, *consonant*, *sibilant*, or *plosive*; of colors as *primary* and *secondary*; and of numbers as *odd*, *even*, *prime*, and *complex*.

It is also very important; it provides the basis for many significant notions, such as graphs, barcharts, and sets.

A classification may be *complete*, (each object falling into at least one class), and it may be *disjoint* (each object falling into at most one class). A *graph* is a *disjoint* classification corresponding to the non-disjoint classification used to produce a barchart.

```
x=: 1 2 3 4 5 6 7
]y=: (x-3) * (x-5)
```

8 3 0 \_1 0 3 8

Parabola (roots at 3 and 5)

```
range=: >./ - i.@spread
spread=: 1: + >./ - <./
spread y
10
```

```
range y
8 7 6 5 4 3 2 1 0 _1
```

```
((range <:/ ]) ; {&' *'@(range<:/)) y
```

Barcharts of y

```
+-----+-----+
1 0 0 0 0 0 1	*      *
1 0 0 0 0 0 1	*      *
1 0 0 0 0 0 1	*      *
1 0 0 0 0 0 1	*      *
1 0 0 0 0 0 1	*      *
1 1 0 0 0 1 1	**     **
1 1 0 0 0 1 1	**     **
1 1 0 0 0 1 1	**     **
1 1 1 0 1 1 1	***    ***
1 1 1 1 1 1 1	*****
+-----+-----+
```



## 9. Disjoint Classification (Graphs)

If only the final element of a boolean list `b` is non-zero, then (and only then) will the result of `</b` be non-zero. Consequently, `</\` applied to `b` suppresses all ones after the first, and the result therefore represents a disjoint classification. For example:

```
cct=: #:@i.@(2: ^ #)
b=: |: cct 2 3 5 7
b
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

</b
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

</\b
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

y=: (x-3) * (x-5) [ x=: 1 2 3 4 5 6 7
range=: >./ - i.@spread
spread=: 1: + >./ - <./
bc=: (range <:/]) y
bc;(</\bc);({&'.'*' </\bc)
```

Barchart and graphs

```
+-----+-----+-----+
1 0 0 0 0 0 1	1 0 0 0 0 0 1	*.....*
1 0 0 0 0 0 1	0 0 0 0 0 0 0	.....
1 0 0 0 0 0 1	0 0 0 0 0 0 0	.....
1 0 0 0 0 0 1	0 0 0 0 0 0 0	.....
1 0 0 0 0 0 1	0 0 0 0 0 0 0	.....
1 1 0 0 0 1 1	0 1 0 0 0 1 0	.*...*
1 1 0 0 0 1 1	0 0 0 0 0 0 0	.....
1 1 0 0 0 1 1	0 0 0 0 0 0 0	.....
1 1 1 0 1 1 1	0 0 1 0 1 0 0	..*.*..
1 1 1 1 1 1 1	0 0 0 1 0 0 0	...*...
```

+-----+-----+-----+

## 10. Classification (with Selection and Inner Product)

The complete classification table can be used in a variety of ways, including use with various inner products. For example:

```
cct=: #:i.(2: ^ #)
m=: 2 3 5 ,: 4 2 1
n=: |: cct 0{m
m ; n ; m +/ . * n
```

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |    |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| 2 | 3 | 5 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 5 | 3 | 8 | 2 | 7 | 5 | 10 |
| 4 | 2 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7  |
|   |   |   | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |   |   |   |   |   |   |   |    |

The pattern of the inner product can be seen more clearly in the following display: the element in a given row and column of the matrix product  $p$  in the lower right box corresponds to the row of the left argument and the column of the right argument. Thus:

```
(' ' ; n) ,: (m ; p=: m +/ . * n)
```

|  |   |   |   |   |   |   |   |   |
|--|---|---|---|---|---|---|---|---|
|  | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|  | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
|  | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

|   |   |   |   |   |   |   |   |   |   |    |
|---|---|---|---|---|---|---|---|---|---|----|
| 2 | 3 | 5 | 0 | 5 | 3 | 8 | 2 | 7 | 5 | 10 |
| 4 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7  |

```
(+/r*c) ; (r*c) ; (r=: 0{m) ; (c=: 3{"1 n) ; (<0 3){p +-----+
+-----+-----+--+ |8|0 3 5|2 3 5|0 1 1|8| +-----+-----+-----+
Just as the ordinary matrix product yields sums over products, the inner product
*/ . ^ yields products over powers. Hence, m */ . ^ n produces products
over all possible subsets of the rows of m:
```

```
m */ . ^ n
```

|   |   |   |    |   |    |   |    |
|---|---|---|----|---|----|---|----|
| 1 | 5 | 3 | 15 | 2 | 10 | 6 | 30 |
| 1 | 1 | 2 | 2  | 4 | 4  | 8 | 8  |

Also see the use of the *key* adverb (/ .) for classification.

## 11. Classification (Sets and Propositions)

The list `-.+./t` appended to any classification table `t` will yield a complete classification table, and the function defined below therefore completes a classification table. The function `tab` ensures that a scalar or vector argument is treated as a one-rowed table.

```
c=: complete=: (] , (+./ { . , :)@:-.:@:(+./))@:tab
tab=: , :^:(0:>.2:-#@$(
```

```
    c 0 0 1, :0 1 0
0 0 1
0 1 0
1 0 0
```

```
    c 1 0 1, :0 1 0
1 0 1
0 1 0
```

```
    (c 1 0 1);(c c 1 0 1);(c 0);(c 1)
+-----+-----+---+
|1 0 1|1 0 1|0|1|
|0 1 0|0 1 0|1| |
+-----+-----+---+
```

A function that yields a single boolean list is called a *proposition*; its result is a one-way classification called a *set*. The classification can, of course, be completed by the complementary set. For example:

```
p1=: 2<: *. <5          Set defined by interval
p1a=: (2:<:] ) *. ( ]<5:) Alternative definition
p2=: = <.              Set of integers
a=: 2 %~ i. 11
(] ,p1,p1a,p2,(p1+.p2),:(p1*.p2)) a
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
0 0 0 0 1 1 1 1 1 1 0
0 0 0 0 1 1 1 1 1 1 0
1 0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 1 1 1 1 1 1
0 0 0 0 1 0 1 0 1 0 0
```

```

    list=: 1 : 0
x. # ]
)

```

Adverb to list elements of set

```

    ((p1 list);(p2 list);((p1*.p2)list)) a
+-----+-----+-----+
|2 2.5 3 3.5 4 4.5|0 1 2 3 4 5|2 3 4|
+-----+-----+-----+

```

## 12. Sorting

The sort `x /: y` re-orders `x` according to the grade of `y`, that is, `/:y`:

```
x=: 2 7 1 8 [ y=: 1 7 3 2
  (/:y);((/:y){x);(x/:y);(x/:x)
+-----+-----+-----+-----+
|0 3 2 1|2 8 1 7|2 8 1 7|1 2 7 8|
+-----+-----+-----+-----+
```

The grade and sort of literal characters is based upon the ordering of the underlying alphabet `a. .` For example, if the name "text" is used for the present sentence (up to and including the colon), then:

```
tdw=: >dwds=: ~. wds=: ;: text
($tdw),($dwds),($wds),($text)
21 9 21 25 103
```

```
]alph=: a. {~ ,(i. 26) +/ (a.i.'aA')
aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ
```

```
tdw; (tdw /: tdw);(tdw/: alph i. tdw)
+-----+-----+-----+
For	"	and	
example	(	colon	
,	)	example	
if	,	for	
the	For	For	
name	and	if	
"	colon	including	
text	example	is	
is	for	name	
used	if	present	
for	including	sentence	
present	is	text	
sentence	name	then:	
(		present	the
up	sentence	to	
to	text	up	
and	the	used	
```

|           |         |         |   |
|-----------|---------|---------|---|
| including | then:   | ,       |   |
| colon     | to      | "       |   |
| )         | up      | (       |   |
| then:     | used    | )       |   |
| +-----+   | +-----+ | +-----+ | + |

The middle column is the alphabetized table of distinct words, but (because the cases are not interleaved in the alphabet a.), the words "for" and "For" are widely separated; they are brought together in the last column by indexing the table by a suitable alphabet.



## 13. Compositions (Based on Conjunctions)

In math, the symbol  $\circ$  is commonly used to produce a function defined as the *composition* of two functions:  $f \circ g \ y$  is defined as  $f (g \ y)$ . Normally, such composed functions are only defined to apply to a single scalar argument.

**J** provides compositions effected by five distinct conjunctions, as well as compositions effected by isolated sequences of verbs: hooks and forks, and longer trains formed from them. The five conjunctions are  $\&$ ,  $\&.$ ,  $\&:$ ,  $@$  and  $@:$ , the conjunctions  $@$  and  $@:$  being related in the same manner as  $\&$  and  $\&:$ .

The conjunction  $\&$  is closest to the composition  $\circ$  used in math, being identical to it when used for two scalar (rank zero) functions to produce a function to be applied to a single scalar argument. However, it is also extended in two directions:

1. Applied to one verb and one noun it produces a monadic function illustrated by the cases  $10\&^.$  (Base ten logarithm) and  $\&^3$  (Cube).
2. Applied to two verbs it produces (in addition to the monadic case used in math) a dyadic case defined by:  $x \ f\&g \ y \leftrightarrow (g \ x) \ f \ (g \ y)$ . For example,  $x \ \%!\ y$  is the quotient of the factorials of  $x$  and  $y$ .

The conjunction  $\&.$  applies only to verbs, and  $f\&.g$  is equivalent to  $f\&g$  except that the inverse of  $g$  is applied to the final result. For example:

$$\begin{array}{cc} 3 \ +\&^{\cdot} \ . \ 4 & 3 \ +\&.\&^{\cdot} \ . \ 4 \\ 2.48491 & 12 \end{array}$$

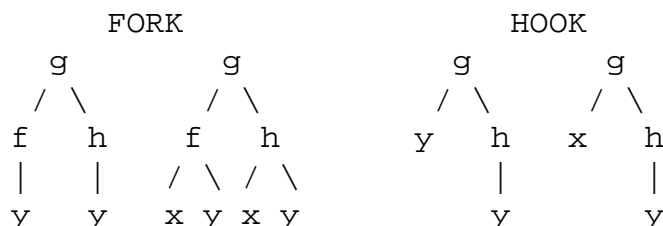
For scalar arguments the functions  $f\&:g$  and  $f\&g$  are equivalent, but for more general arguments,  $g$  applies to each cell as dictated by its ranks. In the case of  $f\&g$ , the function  $f$  then applies to each result produced; in the case of  $f\&:g$  it applies to the overall result of all of the cells. For example:

$$\begin{array}{c} ([ \ ; \ \% \ . \ ; \ | \ : \ \% \ . \ ; \ | \ : \ \% \ : \ \% \ . \ ) \ i. \ 2 \ 2 \ 2 \\ +---+-----+-----+-----+ \\ | 0 \ 1 | \_1.5 \ 0.5 | \_1.5 \ 1 | \_1.5 \ \_3.5 | \\ | 2 \ 3 | \quad 1 \quad 0 | 0.5 \ 0 | \quad 1 \quad 3 | \end{array}$$



## 14. Compositions (Based on Hooks and Forks)

A *verb* is produced by trains of three or two verbs, as defined by the following diagrams:



For example,  $5(+*- )3 \leftrightarrow (5+3)*(5-3)$ .

The foregoing definition is from [Section II F p68](#) of the dictionary. The following examples concern functions used in statistics:

|                                        |                        |
|----------------------------------------|------------------------|
| mean=: +/%#                            | Mean                   |
| norm=: ] - +/ % #                      | Centered on mean       |
| sqcm=: 2: ^~ ] - +/ % #                | Sqr of centred on mean |
| var =: [: mean 2: ^~ ] - mean=: +/%#   | Variance               |
| stdv=: 2:%[:mean 2: ^~ ] - mean=: +/%# | Standard Deviation     |

```

, . & . > @ ( ] ; mean ; norm ; sqcm ; var ; stdv ) y=: 2 3 4 5
+--+---+-----+-----+-----+-----+
2	3.5	_1.5	2.25	1.25	1.11803
3		_0.5	0.25		
4		0.5	0.25		
5		1.5	2.25		
+--+---+-----+-----+-----+

```

## 15. Junctions

Four functions are commonly used to join arguments: `;`, `,`, `.` and `,:`. We will illustrate them for the cases of vector and matrix arguments:

```
a=: 'pqr' [ b=: 'PQR'
m=: 3 3$ 'abcdefghi' [ n=: 3 3$ 'ABCDEFGHI'
```

```
a (i i , i ,. i ,:) b
+-----+-----+-----+
+---+---+	pqrPQR	pP	pqr			
	pqr	PQR			qQ	PQR
+---+---+		rR				
+-----+-----+-----+
```

```
m (i i , i ,. i ,:) n
+-----+-----+-----+
+---+---+	abc	abcABC	abc			
	abc	ABC		def	defDEF	def
	def	DEF		ghi	ghiGHI	ghi
	ghi	GHI		ABC		
+---+---+	DEF		ABC			
	GHI		DEF			
			GHI			
+-----+-----+-----+
```

```
a (i i , i ,. i ,:) n
+-----+-----+-----+
+---+---+	pqr	pABC	pqr			
	pqr	ABC		ABC	qDEF	
		DEF		DEF	rGHI	
		GHI		GHI		
+---+---+			ABC			
				DEF		
				GHI		
+-----+-----+-----+
```

```
m (i i , i ,. i ,:) b
+-----+-----+-----+
|+---+---+|abc|abcP|abc|
```

[illegible]

## 16. Partitions (Adverbs)

Used monadically, the results of the adverbs `\` and `\.` and `/.` provide *prefix*, *suffix*, and *oblique* partitions. They are commonly applied to arithmetic functions such as summation (`+/`), product over (`*/`), and continued fractions (`((+%) /)`); we will also illustrate them for `box (<)`, which shows their structure more clearly:

```
a=: 2 3 5 7 11 [ t=: 1 2 1 */ 1 3 3 1

, .&.>((+/\a) ; (+/\.a) ; ((+%) / \a) ; (+//.t);t)

+---+---+-----+---+-----+
2	28	2	1	1 3 3 1
5	26	2.33333	5	2 6 6 2
10	23	2.3125	10	1 3 3 1
17	18	2.31304	10	
28	11	2.31304	5	
			1	
+---+---+-----+---+-----+
```

```
<\a

+---+---+-----+-----+-----+
|2|2 3|2 3 5|2 3 5 7|2 3 5 7 11|
+---+---+-----+-----+-----+
```

```
<\.a

+-----+-----+-----+-----+---+
|2 3 5 7 11|3 5 7 11|5 7 11|7 11|11|
+-----+-----+-----+-----+---+
```

```
</.t

+---+---+-----+---+---+
|1|3 2|3 6 1|1 6 3|2 3|1|
+---+---+-----+---+---+
```

Used dyadically, they provide *infix*, *outfix*, and *key classification*. For example:

```
3 <\ a

+-----+-----+-----+
|2 3 5|3 5 7|5 7 11|
+-----+-----+-----+
```

```

      _3 <\ a
+-----+-----+
| 2 3 5 | 7 11 |
+-----+-----+

```

```

      2<\.a
+-----+-----+-----+-----+
| 5 7 11 | 2 7 11 | 2 3 11 | 2 3 5 |
+-----+-----+-----+-----+

```

```

      1 2 3 1 3 *//. a
14 3 55

```

## 17. Partitions (Based on Cut Conjunction)

The left argument of the cut conjunction is a function which is applied to various types of partitions specified by the numeric right argument. We will illustrate its behaviour for the fixed function *box*, using the adverb *< i . :*

```
t=: 'When eras die/their legacies/are left to/strange police/'

i: t
+-----+-----+-----+-----+-----+-----+-----+-----+
|When|eras|die|/|their|legacies|/|are|left|to|/|strange|police|/|
+-----+-----+-----+-----+-----+-----+-----+-----+

cut=: < i .

_2 cut i: t
+-----+-----+-----+-----+-----+-----+-----+-----+
|+-----+-----+-----+-----+-----+-----+-----+-----+
||When|eras|die|/|their|legacies|/|are|left|to|/|strange|police|/|
|+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+

2 cut i: t
+-----+-----+-----+-----+-----+-----+-----+-----+
|+-----+-----+-----+-----+-----+-----+-----+-----+
||When|eras|die|/|their|legacies|/|are|left|to|/|strange|police|/|
|+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+

(i. 3 2 2);(i. 12 12)
+-----+-----+-----+-----+-----+-----+-----+-----+
0 1	0 1 2 3 4 5 6 7 8 9 10 11
2 3	12 13 14 15 16 17 18 19 20 21 22 23
	24 25 26 27 28 29 30 31 32 33 34 35
4 5	36 37 38 39 40 41 42 43 44 45 46 47
6 7	48 49 50 51 52 53 54 55 56 57 58 59
	60 61 62 63 64 65 66 67 68 69 70 71
8 9	72 73 74 75 76 77 78 79 80 81 82 83
10 11	84 85 86 87 88 89 90 91 92 93 94 95
	96 97 98 99 100 101 102 103 104 105 106 107
	108 109 110 111 112 113 114 115 116 117 118 119
	120 121 122 123 124 125 126 127 128 129 130 131
	132 133 134 135 136 137 138 139 140 141 142 143
+-----+-----+-----+-----+-----+-----+-----+-----+

(i. 3 2 2) 0 cut i. 12 12
```



|       |    |    |     |     |     |     |     |     |     |     |     |     |
|-------|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 2 3 |    |    | 53  | 54  | 55  | 56  | 57  | 58  | 59  | 105 | 106 | 107 |
| 13    | 14 | 15 | 65  | 66  | 67  | 68  | 69  | 70  | 71  | 117 | 118 | 119 |
|       |    |    | 77  | 78  | 79  | 80  | 81  | 82  | 83  | 129 | 130 | 131 |
|       |    |    | 89  | 90  | 91  | 92  | 93  | 94  | 95  | 141 | 142 | 143 |
|       |    |    | 101 | 102 | 103 | 104 | 105 | 106 | 107 |     |     |     |
|       |    |    | 113 | 114 | 115 | 116 | 117 | 118 | 119 |     |     |     |

## 18. Geometry

We will illustrate the topic of coordinate geometry by defining functions for polygons in two dimensions that give the displacements between adjacent vertices, the length of sides, the semiperimeter, and the area based on Heron's formula.

We also present a more general definition for area that not only gives the signed area (positive if the vertices appear in counter-clockwise order), but also applies to polyhedra in higher dimensions (in which case the name *area* might better be replaced by *volume*). This definition is based on the use of the determinant applied to a square matrix obtained by bordering a table of vertices `t` by a final row of the values `%!#t`. Thus:

```
(length=: +/&.*:) 5 12
13

disp=: ] - 1&|. "1
sides=: length@disp
semiper=: -:@(+/@)@sides
HERON=: %:@(*/@)@(semiper - 0: , sides)
area=: -/ . * @ (] , %@!@#)

t=: 0 0 4 ,: 3 4 7
(];(1&|. "1);disp;sides;semiper;HERON;area) t
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 4|0 4 0| 0 _4 4|0 4 4|0.5 3.5 4|2.29129 0j0.5 0|_2|
|3 4 7|4 7 3|_1 _3 4|1 3 4|          |          |          |
+-----+-----+-----+-----+-----+-----+-----+

tet1=:6 0 3 0,3 6 5 8,:7 4 0 5
tet2=: 0,.=i.3 3
tet1;(area tet1);tet2;(area tet2)
+-----+-----+-----+-----+
6 0 3 0	11.5	0 1 0 0	_0.166667
3 6 5 8		0 0 1 0	
7 4 0 5		0 0 0 1	
+-----+-----+-----+-----+
```

## 19. Symbolic Functions

For any function, a corresponding *symbolic* function can be defined to display the expression rather than evaluate it. For example:

```

minus=: [ , '-' "_ , ]
'a' minus 'b'
a-b

list=: 'abcd'
table=: 4 4$'ABCDEFGHJKLMNOP'
minus/list
a-b-c-d

(minus/\list);('01'minus"0/list);(minus//.table);table
+-----+---+-----+-----+
a	0-a	A	ABCD
a-b	0-b	B-E	EFGH
a-b-c	0-c	C-F-I	IJKL
a-b-c-d	0-d	D-G-J-M	MNOP
		H-K-N	
	1-a	L-O	
	1-b	P	
	1-c		
	1-d		
+-----+---+-----+-----+

(, .list)=: 4 3 2 1
(" . minus/\list) ,: (-/\4 3 2 1)
4 1 3 2
4 1 3 2

3 (minus/\ ; minus/\.) 'abcdefg'
+-----+-----+
a-b-c	d-e-f-g
b-c-d	a-e-f-g
c-d-e	a-b-f-g
d-e-f	a-b-c-g
e-f-g	a-b-c-d
+-----+-----+

```

## 20. Directed Graphs

A *directed graph* is a collection of *nodes* with *connections* or *arcs* specified between certain pairs of nodes. It can be used to specify matters such as the precedences in a set of processes (stuffing of envelopes must precede sealing), or the structure of a *tree*.

The connections can be specified by a boolean *connection matrix* instead of by arcs, and the connection matrix can be determined from the list of arcs.

The connection matrix is convenient for determining various properties of the graph, such as the *in-degrees* (number of arcs entering a node), the *out-degrees*, immediate descendants, and the *closure*, or connection to every node reachable through some path. For example:

```

from=: 3 7 2 5 5 7 1 5 5 5 2 6 1 2 3 7 7 4 7 2 7 4
to=: 5 6 0 2 6 2 7 6 0 7 3 3 2 1 7 0 4 2 3 0 0 3

$ arcs=: from,.to
22 2

|: arcs { nodes=: 'ABCDEFGH'           Transposed for display
DHCFFHBFFFCGBCDHHEHCHE
FGACGCHGAHDDCBHAECDAAD

CM=: #. e.~ [: i. [ , [               Connection matrix from arcs
]cm=: (>:>./,arcs) CM arcs
0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 1
1 1 0 1 0 0 0 0
0 0 0 0 0 1 0 1
0 0 1 1 0 0 0 0
1 0 1 0 0 0 1 1
0 0 0 1 0 0 0 0
1 0 1 1 1 0 1 0

(+ /cm);(+ /"1 cm); (+ /+ /cm);(#arcs);(#~.arcs)
+-----+-----+-----+
|3 1 4 4 1 1 2 3|0 2 3 2 2 4 1 5|19|22|19|

```

```
+-----+-----+--+--+--+
```

The foregoing results are the in, out, and total degrees; followed by the number of arcs, and the number of distinct arcs. A boolean vector `b` may be used to represent the nodes, and the inner product `b +./ . *. cm` gives the same representation of the nodes reachable from them. The immediate family (which includes the original points themselves) is therefore given by the function `imfam` :

```
imfam=: [ +. +./ . *.
        (b=: 1 0 0 0 0 0 0 1) imfam cm
1 0 1 1 1 0 1 1
```

## 21. Closure

Just as `b imfam cm` produces the immediate family of `b`, so does the phrase `cm imfam cm` produce the immediate families of each of the rows of `cm`. We will, however, use a new sparser connection matrix that will be more instructive, and will use powers of `imfam` to produce families of further generations, including an infinite power to give the *closure* of the connection matrix; that is, the connection matrix for all points reachable by a path of any length:

```
cm=: (i. =/ <:@i.) 8
<"2 cm imfam^:0 1 2 _ cm
```

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |   |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |   |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |   |   |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |   |   |

The closure of `cm` can therefore be expressed as `cm imfam^:_ cm`, and a monadic closure function can be defined as follows:

```
(closure=: imfam^:_ ~) cm
0 1 1 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 0 1 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 0 0 1 1 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
```

The complete definition of the closure function may now be displayed as follows:

```
closure f.
([ +. +./ .*.)^:_~
```

## 22. Distance

The "street" distance between two points will be defined as the sum of the magnitudes of the difference of their coordinates along each axis. Thus:

```
d=: +/@:|@:-"1
p=: 3 5 1 [ q=: 7 4 0
p d q
```

6

```
table=: #: i. 2^3
(]; d/~) table
```

Table and distances between each pair of points in it

```
+-----+-----+
0 0 0	0 1 1 2 1 2 2 3
0 0 1	1 0 2 1 2 1 3 2
0 1 0	1 2 0 1 2 3 1 2
0 1 1	2 1 1 0 3 2 2 1
1 0 0	1 2 2 3 0 1 1 2
1 0 1	2 1 3 2 1 0 2 1
1 1 0	2 3 1 2 1 2 0 1
1 1 1	3 2 2 1 2 1 1 0
+-----+-----+
```

```
g=: [ * [ = d/~@]
(];(d/~);(1&g);(2&g)) table
```

```
+-----+-----+-----+-----+
0 0 0	0 1 1 2 1 2 2 3	0 1 1 0 1 0 0 0	0 0 0 2 0 2 2 0
0 0 1	1 0 2 1 2 1 3 2	1 0 0 1 0 1 0 0	0 0 2 0 2 0 0 2
0 1 0	1 2 0 1 2 3 1 2	1 0 0 1 0 0 1 0	0 2 0 0 2 0 0 2
0 1 1	2 1 1 0 3 2 2 1	0 1 1 0 0 0 0 1	2 0 0 0 0 2 2 0
1 0 0	1 2 2 3 0 1 1 2	1 0 0 0 0 1 1 0	0 2 2 0 0 0 0 2
1 0 1	2 1 3 2 1 0 2 1	0 1 0 0 1 0 0 1	2 0 0 2 0 0 2 0
1 1 0	2 3 1 2 1 2 0 1	0 0 1 0 1 0 0 1	2 0 0 2 0 2 0 0
1 1 1	3 2 2 1 2 1 1 0	0 0 0 1 0 1 1 0	0 2 2 0 2 0 0 0
+-----+-----+-----+-----+
```

```
{&' .+*' &.> (];(d/~);(1&g);(2&g);(3&g)) table
```

```
+-----+-----+-----+-----+
|      | .+.++*| .. .   | + ++   | *      |
| .|. .+.+. *+|. . .   | + + +   | *      |
```

|  |   |  |   |   |   |   |   |   |   |   |   |  |   |   |   |   |  |  |   |  |
|--|---|--|---|---|---|---|---|---|---|---|---|--|---|---|---|---|--|--|---|--|
|  | . |  | + | . | + | * | + |   | . | . | . |  | + | + | + |   |  |  | * |  |
|  | . |  | + | . | . | * | + | + |   | . | . |  | + |   | + | + |  |  | * |  |
|  | . |  | + | + | * | . | + |   | . | . | . |  | + | + |   | + |  |  | * |  |
|  | . |  | + | . | * | + | . | + |   | . | . |  | + | + | + |   |  |  | * |  |
|  | . |  | + | * | . | + | + | . |   | . | . |  | + | + | + |   |  |  | * |  |
|  | . |  | * | + | + | . | + | . |   | . | . |  | + | + | + |   |  |  | * |  |
|  | . |  | * | + | + | . | + | . |   | . | . |  | + | + | + |   |  |  | * |  |
|  | . |  | . | . | . | . | . | . |   | . | . |  | + | + | + |   |  |  | * |  |



## 23. Polynomials

The monadic function  $M =: 3: * ] ^ 2:$  is a multiple of an integral power of its argument, and is called a *monomial*; and a sum of monomials such as  $SM =: (3: * ] ^ 2:) + (2.5" _ * ] ^ 4:) + (_5" _ * ] ^ 0:)$  is a *polynomial*.

Any polynomial can be expressed in the *standard* form  $c \& p$ , where  $c$  is a suitable list of *coefficients*, and where  $p =: +/@([ * ] ^ i. @ \# @ [ ) " 1 0$ . For example:

```
SM=: (3:*]^2:)+(2.5"_*]^4:)+(_5"_*]^0:)
p=: +/@([ * ] ^ i. @ # @ [ ) " 1 0
c=: _5 0 3 0 2.5
x=: _2 _1 0 1 2
(SM x), (c p x), : (c&p x)
47 0.5 _5 0.5 47
47 0.5 _5 0.5 47
47 0.5 _5 0.5 47
```

The primitive  $p.$  is equivalent to the function  $p$  defined above, and will be used hereafter. The polynomial  $c \& p.$  is very important for a number of reasons, including:

1. It applies to any numeric argument, real or complex (and the parameter  $c$  may also be complex).
2. It can be used to approximate a wide range of functions.
3. It is *closed* under a number of operations; that is, the sum, difference, product, the composition  $@$ , the derivative, and the integral of polynomials are themselves polynomials.
4. The coefficients of the results of each case listed in 3 are easily expressed. For example, if  $\#c$  equals  $\#d$ , then  $c \& p. + d \& p.$  is equal to  $(c+d) \& p.$ . More generally, it is equal to  $(+ / c, : d) \& p.$ . Thus:

```
ps=: + / @ , :      Polynomial sum
pd=: - / @ , :      Polynomial difference
```

pp=: +//. @ ( \* / )

D=: d.1

pD=: 1: } . ] \* i. @ #

pI=: 0: , ] % 1: + i. @ #

Polynomial product

Scalar (rank 0) first derivative

Polynomial derivative

Polynomial integral

## 24. Polynomials (Continued)

For example:

```

c=: 1 2 1 [ d=: 1 3 3 1
x=: 2 1 0 _1 _2
, .&. > ((c pp d); ((c pp d)&p. x); (c&p.*d&p.)x)
+---+---+---+
1	243	243
5	32	32
10	1	1
10	0	0
5	_1	_1
1		
+---+---+---+

, .&. > ((d&p.D x); (pD d); ((pD d)&p. x); (pI d); pD pI d)
+---+---+---+---+---+
27	3	27	0	1
12	6	12	1	3
3	3	3	1.5	3
0		0	1	1
3		3	0.25	
+---+---+---+---+---+

]e=: c(ps pp pd)d
0 _2 _9 _16 _14 _6 _1

e&p. x
_648 _48 0 0 0

((c&p.+d&p.)*(c&p.-d&p.)) x
_648 _48 0 0 0

f=: c&p.(+*-) d&p.
f x
_648 _48 0 0 0

]g=: pD c pp d
5 20 30 20 5

```

```
(g&p. ,: (c&p.*d&p.) D) x
405 80 5 0 5
405 80 5 0 5
```

## 25. Polynomials in Terms of Roots

The product  $\ast/y-r$  is called a *polynomial in terms of the roots*  $r$ , because it can also be expressed as a polynomial applied to the argument  $y$ , and because  $r$  is the list of *roots* or *zeros* of the resulting function. For example:

```

*/y-r [ y=: 7 [ r=: 2 3 5 [ x=: 7 6 5 4 3 2
40

pp=: +//. @ (*/ )
c=: pp/monomials=: (- ,. 1:) r
cfr=: [: pp/ - ,. 1:          Coefficients from roots
pir=: */@([ ]- [ ]) "1 0      Polynomial in terms of roots

, .&. > (r; monomials; c; (cfr r); (c&p. y); (r pir x))
+--+-----+-----+-----+-----+
2	_2 1	_30	_30	40	40	
3	_3 1	31	31		12	
5	_5 1	_10	_10		0	
			1	1		_2
						0
						0
+--+-----+-----+-----+-----+

```

Since the last (highest order) coefficient produced by `cfr` is necessarily 1, the function `pir` cannot produce a general polynomial, but it can if provided with a multiplier. We therefore re-define `cfr` and `pir` to apply to a boxed list of multiplier and roots as follows:

```

CFR=: (* cfr) &> /
PIR=: CFR@[ p. ]

CFR 3;r
_90 93 _30 3

(3;r) PIR x
120 36 0 _6 0 0

```

We now illustrate the use of a polynomial in approximation:

`]ce=: ^ t. i. 7`
First seven terms of Taylor series for  
 exponential

1 1 0.5 0.166667 0.0416667 0.00833333 0.00138889

`(^ - ce&p.) _1 _0.5 0 0.5 1`
Comparison with exponential  
 \_0.000176114 \_1.45834e\_6 0 1.65264e\_6 0.000226273

`pD ce`
The exponential function equals its own  
 derivative

1 1 0.5 0.166667 0.0416667 0.00833333

## 26. Polynomials: Roots from Coefficients (Newton's Method)

Because the polynomials  $(m;r)\&PIR$  and  $(c=:CFR\ (m;r))\&p.$  are identical, the parameters  $m;r$  and  $c$  are said to be different *representations* of the same function. Each representation has its own useful properties. For example, addition of polynomials is easy in the coefficient representation but difficult in the root representation; the identification of the zeros of the function is difficult in the coefficient representation but trivial in the root representation. It is therefore useful to have functions that transform each representation to the other.  $CFR$  serves for one direction; the inverse problem is approached by methods of successive approximation.

For any function  $f$ , the difference  $(f\ r)-(f\ a)$  for nearby points  $r$  and  $a$  is approximately equal to the difference  $r-a$  multiplied by the slope of the tangent to the graph of  $f$  at the point  $a, f\ a$ , that is, the derivative of  $f$  at  $a$ . Conversely, the difference  $r-a$  is approximated by  $((f\ r)-(f\ a))\%f\ D\ a$ , and  $r$  is approximated by  $a+((f\ r)-(f\ a))\%f\ D\ a$ .

If  $f$  is the polynomial  $c\&p.$  and  $r$  is one of its roots, then  $f\ r$  is zero, and if  $a$  is an approximation to  $r$ , the expression for  $r$  reduces to  $a-(f\ a)\%f\ D\ a$ . This may provide a better approximation to  $r$ , and is embodied in *Newton's method*, defined as an adverb, and illustrated as follows:

```
newton=: 1 : 0
] - x. % x.D
)

f=: (c=: 12 _10 2)\&p.

f a=: 2.4
_0.48

f newton a
1.2

f 2
```

0

```
f newton ^:0 1 2 3 4 _ a
2.4 1.2 1.75385 1.9594 1.99848 2
```

```
]a=: (^ - 4:) newton ^: 0 1 2 3 _ a=: 1
1 1.47152 1.38982 1.3863 1.38629
```

```
^ {: a
4
```

For the particular case of polynomials, we may define an adverb that applies to coefficients and uses the polynomial derivative `pD` instead of the general derivative `D`:

```
pD=: 1: }. ] * i.@#
NEWTON=: 1 : ' ] - x.&p. % (pD x.)&p. '

c NEWTON ^:0 1 2 3 4 _ a=: 2.4
2.4 1.2 1.75385 1.9594 1.99848 2
```



## 27. Polynomials: Roots from Coefficients (Kerner's Method)

Newton's method applies to one root at a time and requires a good starting approximation. Kerner's method is a generalization that gives all the roots, starting from a *list*  $a$  and dividing each element of the residual  $f(a)$  by the derivative with respect to the corresponding root. It applies only to a polynomial whose highest order coefficient is 1, and we first normalize the coefficients by dividing by the last, yielding a polynomial having the same roots. The method converges to complex roots only if at least one of the initial approximations is complex. We will use the Taylor series approximation to the exponential function, because the corresponding polynomial has complex roots:

```

]d=: ^ t. i.6
1 1 0.5 0.166667 0.0416667 0.00833333

]c=: (norm=: % {:) d
120 120 60 20 5 1

+. a=: (init=: r.@}.@i.@#) c |a
0.540302 0.841471 1 1 1 1 1
_0.416147 0.909297
_0.989992 0.14112
_0.653644 _0.756802
0.283662 _0.958924

deriv=: [: */ 0&=@{.}%(-/~ ,: 1:)

kerner=: 1 : 0
] - x.&p. % deriv@]
)

r=: c kerner ^:_ a
+.(/:|) r
magnitude
_2.18061 4.57601e_31
_1.6495 1.69393
_1.6495 _1.69393
0.239806 3.12834

```

Real and imaginary parts of roots sorted by magnitude

```
0.239806      _3.12834
```

```
>./|c p. r
1.04488e_13
```

These results may be compared with the use of the primitive `p.` on the unnormalized coefficients `d`. Thus:

```
p. 2 4 2
++-----+
|2|_1 _1|
++-----+

,.;}.p. d
0.2398064j3.12834
0.2398064j_3.12834
 _1.6495j1.69393
 _1.6495j_1.69393
 _2.18061
```

Newton's method may also be used for a complex root:

```
+. d NEWTON ^:0 1 2 3 _ a=: 1j1
      1      1
0.0166065  0.99639
_0.990523  0.992532
_1.95338   1.10685
 _1.6495    1.6939
```



to *Stirling* numbers:

```
(0 TO _1 i.5);(_1 TO 0 i.5)
+-----+-----+
1 0 0 0 0	1 0  0  0  0
0 1 1 1 1	0 1 _1  2 _6
0 0 1 3 7	0 0  1 _3 11
0 0 0 1 6	0 0  0  1 _6
0 0 0 0 1	0 0  0  0  1
+-----+-----+
```

The stoep polynomial is also provided by the variant `p.!s`.

# Dictionary

**J** is a dialect of APL, a formal imperative language. Because it is imperative, a sentence in **J** may also be called an *instruction*, and may be *executed* to produce a *result*. Because it is formal and unambiguous it can be executed mechanically by a computer, and is therefore called a *programming language*. Because it shares the analytic properties of mathematical notation, it is also called an analytic language.

APL originated in an attempt to provide consistent notation for the teaching and analysis of topics related to the application of computers, and developed through its use in a variety of topics, and its implementation in computer systems[\[1-5\]](#)  
[p212](#).

**J** is implemented in C (as detailed in [Hui \[6\] p212](#)), and is ported to a number of different host computer systems. The effect of the specific host is minimal, and communication with it is confined to the single *foreign conjunction* detailed in [Appendix A p214](#). See help files for other host facilities such as Windows.

The Introduction in this book provides guidance to beginners. [References \[7-9\]](#)  
[p212](#) use **J** in the exposition of various mathematical topics.

# I. Alphabet and Words

The alphabet is standard ASCII, comprising *digits*, *letters* (of the English alphabet), the *underline* (used in names and numbers), the (single) *quote*, and others (which include the space) to be referred to as *graphics*.

Numbers are denoted by digits, the underbar (for negative signs and for infinity and minus infinity - when used alone or in pairs), the period (used for decimal points and *necessarily* preceded by one or more digits), the letter *e* (as in  $2.4e3$  to signify 2400 in exponential form), and the letter *j* to separate the real and imaginary parts of a complex number, as in  $3e4j_0.56$ . Also see the discussion of [Constants p199](#).

A numeric *list* or *vector* is denoted by a list of numbers separated by spaces. A list of ASCII characters is denoted by the list enclosed in single quotes, a pair of adjacent single quotes signifying the quote itself: 'can't' is the five-character contraction of the six-character English word 'cannot'. The *ace* *a:* denotes the boxed empty list <\$0 .

*Names* (used for pronouns and other surrogates, and assigned referents by the copula, as in `prices=: 4.5 12`) begin with a letter and may continue with letters, underlines, and digits. A name that ends with an underline or that contains two consecutive underlines is a *locative*, as discussed in [Section II.I p71](#).

A *primitive* or *primary* may be denoted by a single graphic (such as *+* for *plus*) or by a graphic modified by one or more following *inflections* (a period or colon), as in *+.*  and *+:*  for *or* and *nor*. A primary may also be an inflected name, as in *e.*  and *o.*  for *membership* and *pi times*. A primary cannot be assigned a referent.

## II. Grammar

The following sentences illustrate the six parts of speech:

```
fahrenheit=: 50
(fahrenheit-32)*5%9
10
```

```
prices=: 3 1 4 2
orders=: 2 0 2 1
orders * prices
6 0 8 2
```

```
+ /orders*prices
16
```

```
+ / \ 1 2 3 4 5
1 3 6 10 15
```

```
bump=: 1&+
bump prices
4 2 5 3
```

### PARTS of SPEECH

|               |                |
|---------------|----------------|
| 50 fahrenheit | Nouns/Pronouns |
| + - * % bump  | Verbs/Proverbs |
| / \           | Adverbs        |
| &             | Conjunction    |
| ( )           | Punctuation    |
| =:            | Copula         |

Verbs act upon nouns to produce noun results; the nouns to which a particular verb applies are called its *arguments*. A verb may have two distinct (but usually related) meanings according to whether it is applied to one argument (to its right), or to two arguments (left and right). For example, `2%5` yields `0.4`, and `%5` yields `0.2`.

An adverb acts on a single noun or verb to its *left*. Thus, `+ /` is a derived verb

(which might be called *plus over*) that sums an argument list to which it is applied, and `*/` gives the product of a list. A conjunction applies to two arguments, either nouns or verbs.

Punctuation is provided by parentheses that specify the sequence of execution as in elementary algebra; other punctuation includes `if. do. end.` as discussed under [Explicit Definition \(: p114\)](#) and [Control Structures p200](#).

The word `=:` behaves like the copulas "is" and "are" in English, and is read as such, as in "area is 3 times 4" for `area=: 3*4`. The name `area` thus assigned is a *pronoun* and, as in English, it plays the role of a noun. Similar remarks apply to names assigned to verbs, adverbs, and conjunctions. Entry of a name alone displays its value. Errors are discussed in [Section II.J \(Errors and Suspension\) p72](#).



## A. Nouns

Nouns are classified in three independent ways: numeric or literal; open or boxed; arrays of various ranks. The atoms of any array must belong to a single class: numeric, literal, or boxed. Arrays of ranks 0, 1, and 2 are also called *atom*, *list*, and *table*, or, in math, *scalar*, *vector*, and *matrix*. Numbers and literals are represented as stated in [Part I p61](#).

**Arrays.** A single entity such as 2.3 or \_2.3j5 or 'A' or '+' is called an atom. The verb denoted by comma chains its arguments to form a list whose *shape* (given by the verb \$) is equal to the number of atoms combined. For example:

```
$ date=: 1,7,7,6          word=: 's','a','w'
4
|. date                  |. word
was                      6 7 7 1
```

The verb |. used above is called *reverse*. The phrase s\$b produces an array of shape s from the list b. For example:

```
(3,4) $ date,1,8,6,7,1,9,1,7
1 7 7 6
1 8 6 7
1 9 1 7

table=: 2 3$ word,'bat'
table
saw
bat

$table
2 3
```

The number of atoms in the shape of a noun is called its *rank*. Each position of the shape is called an *axis* of the array, and axes are referred to by indices 0, 1, 2, etc. For example, axis 0 of table has length 2 and axis 1 has length 3.

The last *k* axes of an array b determine *rank-k cells* or *k-cells* of b. The rest of the shape vector is called the frame of b relative to the cells of rank *k*; if \$c is 2 3 4 5, then c has the frame 2 3 relative to cells of rank 2, the frame 2 3 4 5 relative to 0-cells (atoms), and an empty frame relative to 4-cells. If:

```

] b=:2 3 4 $ 'abcdefghijklmnopqrstuvw'
abcd
efgh
ijkl

mnop
qrst
uvwx

```

then the list `abcd` is a 1-cell of `b`, and the letters are each 0-cells.

A cell of rank one less than the rank of `b` is called an *item* of `b`; an atom has one item, itself. For example, the verb *from* (denoted by `{`) selects items from its right argument, as in:

|                                 |                                 |                         |
|---------------------------------|---------------------------------|-------------------------|
| <pre> 0{b abcd efgh ijkl </pre> | <pre> 1{b mnop qrst uvwx </pre> | <pre> 0{0{b abcd </pre> |
| <pre> 2 1{0{b ijkl efgh </pre>  | <pre> 1{2{0{b j </pre>          | <pre> 0{3 3 </pre>      |

Moreover, the verb *grade* (denoted by `/:`) provides indices to `{` that bring items to "lexical" order. Thus:

|                                                |                        |                                          |
|------------------------------------------------|------------------------|------------------------------------------|
| <pre> g=: /: n=: 4 3\$3 1 4 2 7 9 3 2 0 </pre> |                        |                                          |
| <pre> n 3 1 4 2 7 9 3 2 0 3 1 4 </pre>         | <pre> g 1 0 3 2 </pre> | <pre> g{n 2 7 9 3 1 4 3 1 4 3 2 0 </pre> |

Negative numbers, as in `_2`-cell and `_1`-cell (an item), are also used to refer to cells whose *frames* are of the length indicated by the magnitude of the number. For example, the list `abcd` may be referred to either as a `_2`-cell or as a `1`-cell of `b`.

**Open and Boxed.** The nouns discussed thus far are called *open*, to distinguish them from *boxed* nouns produced by the verb *box* denoted by `<`. The result of *box* is an atom, and boxed nouns are displayed in boxes. *Box* allows one to treat any array (such as the list of letters that represent a word) as a single entity, or atom. Thus:

```

words=:(<'i'),(<'was'),(<'it') letters=: 'i was it' $words

```

```
$letters 3 8 |. words |. letters +---+---+--+ ti saw i |it|was|i| +--  
+---+--+ 2 3$words,|.words +---+---+---+ |i |was|it| +---+---+---+  
|it|was|i | +---+---+---+
```

## B. Verbs

**Monads and Dyads.** Verbs have two definitions, one for the *monadic* case (one argument), and one for the *dyadic* case (two arguments). The dyadic definition applies if the verb is preceded by a suitable left argument, that is, any noun that is not itself an argument of a conjunction; otherwise the monadic definition applies.

The monadic case of a verb is also called a *monad*, and we speak of the *monad* % used in the phrase %4 , and of the *dyad* % used in 3%4 . Either or both may have empty domains.

**Ranks of Verbs.** The notion of verb rank is closely related to that of noun rank: a verb of rank  $k$  applies to each of the  $k$ -cells of its argument. For example (using the array `b` from [Section A p63](#)):

```
,b
abcdefghijklmnopqrstuvwxyz
```

```
, "2 b
abcdefghijklmnopqrstuvwxyz
mnopqrstuvwxyz
```

```
, "_1 b
abcdefghijklmnopqrstuvwxyz
mnopqrstuvwxyz
```

Since the verb *ravel* (denoted by `,`) applies to its entire argument, its rank is said to be *unbounded*. The *rank* conjunction `"` used in the phrase `, "2` produces a related verb of rank 2 that ravel each of the 2-cells to produce a result of shape 2 by 12.

The shape of a result is the frame (relative to the cells to which the verb applies) catenated with the shape produced by applying the verb to the individual cells. Commonly these individual shapes agree, but if not, they are first brought to a common rank by introducing leading unit axes to any of lower rank, and are then brought to a common shape by *padding* with an appropriate *fill* element: space for a character array, 0 for a numeric array, and a boxed empty list for a boxed array. For example:

```
i."0 s=: 2 3 4
0 1 0 0
```

```
>'I'; 'was'; 'here'
I
```

|         |      |
|---------|------|
| 0 1 2 0 | was  |
| 0 1 2 3 | here |

The dyadic case of a verb has two ranks, governing the left and right arguments. For example:

```

p=: 'abc'
q=: 3 5$'wake read lamp '
p,"0 1 q
awake
bread
clamp

```

Finally, each verb has three intrinsic ranks: monadic, left, and right. The definition of any verb need specify only its behaviour on cells of the intrinsic ranks, and the extension to arguments of higher rank occurs systematically. The ranks of a verb merely place upper limits on the ranks of the cells to which it applies; its domain may include arguments of lower rank. For example, matrix inverse (%.) has monadic rank 2, but treats the case of a vector as a one-column matrix.

**Agreement.** In the phrase  $p \vee q$ , the arguments of  $\vee$  must *agree* in the sense that one frame must be a prefix of the other, as in  $p, "0 1 q$  above, and in the following examples:

|              |                  |
|--------------|------------------|
| $p, " 1 1 q$ | $3 4 5 * i. 3 4$ |
| abcwake      | 0 3 6 9          |
| abcread      | 16 20 24 28      |
| abclamp      | 40 45 50 55      |

```

(i.3 4)*3 4 5
0 3 6 9
16 20 24 28
40 45 50 55

```

If a frame contains 0, the verb is applied to a cell of fills. For example:

```

($ # "2 i. 1 0 3 4);($ 2 3 % "1 i. 0 2)
+---+---+
|1 0|0 2|
+---+---+

($ $ "2 i. 1 0 3 4);($ 2 3 %/"1 i. 0 4)
+-----+-----+
|1 0 2|0 2 4|

```

+-----+-----+

## C. Adverbs and Conjunctions

Unlike verbs, adverbs and conjunctions have fixed valence: an adverb is monadic (applying to a single argument to its *left*), and a conjunction is dyadic.

Conjunctions and adverbs apply to noun or verb arguments; a conjunction may produce as many as four distinct classes of results.

For example,  $u \& v$  produces a *composition* of the verbs  $u$  and  $v$ ; and  $^{\wedge} \& 2$  produces the *square* by bonding the power function with the right argument  $2$ ; and  $2 \& ^{\wedge}$  produces the function *2-to-the-power*. The conjunction  $\&$  may therefore be referred to by different names for the different cases, or it may be referred to by the single term *and* (or *with*), which roughly covers all cases.

## D. Comparatives

The comparison  $x=y$  is treated like the everyday use of equality (that is, with a reasonable relative tolerance), yielding 1 if the difference  $x-y$  falls relatively close to zero. Tolerant comparison also applies to other relations and to *floor* and *ceiling* ( $<.$  and  $>.$ ); a precise definition is given in Part III under *equal* ( $=$ ). An arbitrary tolerance  $t$  can be specified by using the *fit* conjunction ( $!.$ ), as in  $x = !.t y$ . The global tolerance can be queried and set using [9!:18 p223](#) and [9!:19 p223](#).



## E. Parsing and Execution

A sentence is evaluated by executing its phrases in a sequence determined by the parsing rules of the language. For example, in the sentence  $10\%3+2$ , the phrase  $3+2$  is evaluated first to obtain a result that is then used to divide 10. In summary:

1. Execution proceeds from right to left, except that when a right parenthesis is encountered, the segment enclosed by it and its matching left parenthesis is executed, and its result replaces the entire segment and its enclosing parentheses.
2. Adverbs and conjunctions are executed before verbs; the phrase  $, "2-a$  is equivalent to  $(, "2)-a$ , not to  $, "(2-a)$ . Moreover, the left argument of an adverb or conjunction is the entire verb phrase that precedes it. Thus, in the phrase  $+ / . * / b$ , the rightmost adverb  $/$  applies to the verb derived from the phrase  $+ / . *$ , not to the verb  $*$ .
3. A verb is applied dyadically if possible; that is, if preceded by a noun that is not itself the right argument of a conjunction.
4. Certain *trains* form verbs, adverbs, and conjunctions, as described in [§ p68](#)
- F.
5. To ensure that these summary parsing rules agree with the precise parsing rules prescribed below, it may be necessary to parenthesize an adverbial or conjunctival phrase that produces anything other than a noun or verb.

One important consequence of these rules is that in an unparenthesized expression the right argument of any verb is the result of the entire phrase to its right. The sentence  $3*p\%q^{|r-5}$  can therefore be *read* from left to right: the overall result is 3 times the result of the remaining phrase, which is the quotient of  $p$  and the part following the  $\%$ , and so on.

Parsing proceeds by moving successive elements (or their *values* except in the case of proverbs and names immediately to the left of a copula) from the tail end of a *queue* (initially the original sentence prefixed by a left marker **\$**) to the top of a *stack*, and eventually executing some eligible portion of the stack and replacing it

by the result of the execution. For example, if  $a = 1\ 2\ 3$ , then  $b = +/2*a$  would be parsed and executed as follows:

```

S b =: + / 2 * a
S b =: + / 2 *      1 2 3
S b =: + / 2      * 1 2 3
S b =: + /      2 * 1 2 3
S b =: +      / 2 * 1 2 3
S b =: +      / 2 4 6
S b =:      + / 2 4 6
S b      =: + / 2 4 6
S b      =: 12
S      b =: 12
S      12
S      S 12

```

The foregoing illustrates two points: 1) Execution of the phrase  $2 * 1\ 2\ 3$  is deferred until the next element (the  $/$ ) is transferred; had it been a conjunction, the  $2$  would have been *its* argument, and the monad  $*$  would have applied to  $1\ 2\ 3$ ; and 2) Whereas the *value* of the name  $a$  moves to the stack, the name  $b$  (because it precedes a copula) moves unchanged, and the pronoun  $b$  is assigned the value 12.

Parsing can be observed using the trace conjunction [13!:16 p225](#).

The executions in the stack are confined to *the first four elements* only, and eligibility for execution is determined only by the *class* of each element (noun, verb, etc., an unassigned name being treated as a verb), as prescribed in the following parse table. The classes of the first four elements of the stack are compared with the first four columns of the table, and the first row that agrees in all four columns is selected. The bold italic elements in the row are then subjected to the action shown in the final column, and are replaced by its result. If no row is satisfied, the next element is transferred from the queue.

|          |                       |                  |          |
|----------|-----------------------|------------------|----------|
| EDGE     | <b>VERB</b>           | <b>NOUN</b> ANY  | 0 Monad  |
| EDGE+AVN | VERB                  | <b>VERB NOUN</b> | 1 Monad  |
| EDGE+AVN | <b>NOUN</b>           | <b>VERB NOUN</b> | 2 Dyad   |
| EDGE+AVN | <b>VERB+NOUN</b> ADV  | ANY              | 3 Adverb |
| EDGE+AVN | <b>VERB+NOUN CONJ</b> | <b>VERB+NOUN</b> | 4 Conj   |

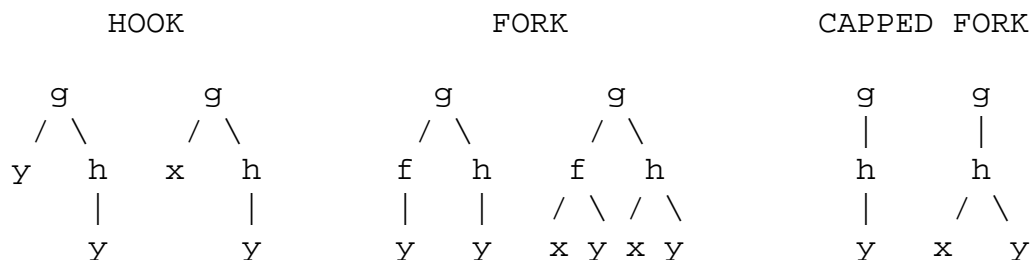
|                       |             |                  |           |
|-----------------------|-------------|------------------|-----------|
| EDGE+AVN              | <b>VERB</b> | <b>VERB VERB</b> | 5 Trident |
| EDGE                  | <b>CAVN</b> | <b>CAVN ANY</b>  | 6 Bident  |
| <b>NAME+NOUN</b> ASGN |             | <b>CAVN ANY</b>  | 7 Is      |
| <b>LPAR</b>           | <b>CAVN</b> | <b>RPAR ANY</b>  | 8 Paren   |

Legend: AVN denotes ADV+VERB+NOUN  
CAVN denotes CONJ+ADV+VERB+NOUN  
EDGE denotes MARK+ASGN+LPAR

## F. Trains

An isolated sequence, such as  $(+ * /)$ , which the "normal" parsing rules do not resolve to a single part of speech is called a *train*, and may be further resolved as described below.

A train of two or three verbs produces a verb and (by repeated resolution), a verb train of any length also produces a verb. For example, the trains  $+ -*\%$  and  $+ -*\%^$  are equivalent to  $+(-*\%)$  and  $+-( *\%^)$ . The production is defined by the following diagrams:



For example,  $5(+*- )3$  is  $(5+3)*(5-3)$ . If  $f$  is a cap ( $[ : )$ ) the capped branch simplifies the forks to  $g \ h \ y$  and  $g \ x \ h \ y$ . The ranks of the hook and fork are infinite.

A two-element train of a conjunction with a noun or a verb produces an adverb. For example,  $\&.>$  produces an adverb that might be called "each", and the adverb  $bc=:<$  might be called "box cells" because, for example,  $0 \ bc \ x$  would box the atoms of  $x$ .

Finally, a train of two adverbs produces an adverb, and (by implication) a train of any number of adverbs also produces an adverb. For example,  $/\backslash$  is the adverb "insert scan", and  $\sim/\sim$  is the "commuted table". For example:

```
is=: /\
+ is 1 2 3 4 5
1 3 6 10 15
```

```
ct=: ~/\~
- ct 1 2 3
0 1 2
```

|    |    |   |
|----|----|---|
| _1 | 0  | 1 |
| _2 | _1 | 0 |

## G. Extended and Rational Arithmetic

Extended precision integer constants can be entered as a sequence of decimal digits terminated by an `x`. The monad `x:` applies to integers and produces extended integers. For example, the 2-element vector `1234x 56x` (or `1234 56x`) is equivalent to `x: 1234 56`. Various primitives produce extended results if the argument(s) are extended. Some functions `f` produce floating point (inexact) results on some extended arguments because the result is not integral; however, `<.@f` and `>.@f` produce extended integer results when applied to extended integer arguments. Comparisons involving extended integers are exact. For example:

```
! 40
8.15915e47
```

```
!40x
8159152832478977343456112695961158942720000000000
```

```
*/ x: >: i.40
8159152832478977343456112695961158942720000000000
```

|                                   |                            |
|-----------------------------------|----------------------------|
| 0j_25 ": ! 2000x * i. 5 1         | Exponent format, 25 digits |
| 1.00000000000000000000000000e0    |                            |
| 3.3162750924506332411753934e5735  |                            |
| 1.8288019515140650133147432e12673 |                            |
| 2.6839997657267395961163166e20065 |                            |
| 5.1841810604808769398058198e27752 |                            |

```
] r=: <.@%: 2 * 10 ^ 56x
14142135623730950488016887242
```

[illegible]

Rational constants can be entered as the decimal digits of the numerator and denominator, separated by an `r` and preceded by an optional sign. Thus `3r4` is the rational number three-quarters and `_12r5` is negative 12 divided by 5.

Rational numbers are stored and displayed in a standard form, with the numerator and denominator relatively prime and the denominator positive. Thus:

```
1r2 _1r2 2r4 2r_4 _2r_4 0r9 5 _5
1r2 _1r2 1r2 _1r2 1r2 0 5 _5
```

Various primitive functions produce (exact) rational results if the argument(s) are rational; non-rational functions produce (inexact) floating point or complex results when applied to rationals, if the function only has a limited number of rational arguments that produce rational results. (For example, `%:y` is rational if the atoms of `y` are perfect squares; `^0r1` is floating point.) The quotient of two extended integers is an extended integer (if evenly divisible) or rational (if not). Comparisons involving two rationals are non-tolerant (exact). Functions or operators that require integer arguments (such as the left arguments of `{` `{.` `#` `$`) also accept rational arguments, if they are integers. Other dyadic functions (e.g. `+` `-` `*` `%` `,` `=` `<`) convert their arguments to the same type, according to the following table:

|   | B | I | X | Q | D | Z |                      |
|---|---|---|---|---|---|---|----------------------|
| B | B | I | X | Q | D | Z | B - boolean          |
| I | I | I | X | Q | D | Z | I - integer          |
| X | X | X | X | Q | D | Z | X - extended integer |
| Q | Q | Q | Q | Q | D | Z | Q - rational         |
| D | D | D | D | D | D | Z | D - floating point   |
| Z | Z | Z | Z | Z | Z | Z | Z - complex          |

For example, in the expression `2.5+1r2`, the `1r2` is converted to `0.5` before being added to `2.5`, resulting in a floating point `3`. And in the expression `2+1r2`, the `2` is converted to `2r1` before being added to `1r2`, resulting in `5r2`.

In particular, a comparison involving a rational and a floating point number is tolerant, because the rational argument is first converted into a floating point number.

The verb `x:` (q.v.) produces rational approximations to non-rational arguments.

```
2%3
0.666667

2%3x
2r3
```

(+%)/\10\$1 Floating point convergents to golden  
mean

1 2 1.5 1.66667 1.6 1.625 1.61538 1.61905 1.61765 1.61818

(+%)/\x: 10\$1 Rational versions of same  
1 2 3r2 5r3 8r5 13r8 21r13 34r21 55r34 89r55

|: 2 x: (+%)/\x: 10\$1  
1 2 3 5 8 13 21 34 55 89  
1 1 2 3 5 8 13 21 34 55

(+%) / 100\$1r1  
573147844013817084101r354224848179261915075

0j30 ": (+%)/100\$1r1 Display 30 decimal places  
1.618033988749894848204586834366

H=: % @: >: @: (+/~) @: i. @ x: Hilbert matrix of order n

] h=: H 6  
1 1r2 1r3 1r4 1r5 1r6  
1r2 1r3 1r4 1r5 1r6 1r7  
1r3 1r4 1r5 1r6 1r7 1r8  
1r4 1r5 1r6 1r7 1r8 1r9  
1r5 1r6 1r7 1r8 1r9 1r10  
1r6 1r7 1r8 1r9 1r10 1r11

-/ .\* h Determinant of h  
1r186313420339200000

~. q: % -/ .\* h Unique prime factors of reciprocal of  
det  
2 3 5 7 11

i.&.(p:^:\_1) 2\*6 Primes less than 2\*n  
2 3 5 7 11

^ 2r1 ^y is floating point or complex  
7.38906

%: 49r25 %: on a rational perfect square is  
rational  
7r5

%: 49r25 10r9  
1.4 1.05409



```
%: _2r1  
0j1.41421
```

```
1 = 1+10r1^_15  
0
```

Exact (rational) comparison

```
(1.5-0.5) = 1+10r1^_15  
1
```

Tolerant (floating point) comparison

```
0.5 = 1r2  
1
```



r), or verb (3 : r or 4 : r). For example:

f=: 3 : s  
f 9  
dyadic cases  
27

The colon in the script separates the monadic and

3 f 4  
arguments  
72

The x. and y. refer to the left and right

The phrases a=: 1 : 0 and c=: 2 : 0 and v=: 3 : 0 provide direct entry of adverbs, conjunctions, and verbs.

Files of scripts may define functions and other entities that can serve to supplement the primaries of **J**. They are commonly referred to as *secondary* or *tertiary* functions according to their relative generality.

# I. Locatives

Locative `abc_f_` refers to `abc` in locale `f`; indirect locative `abc__xy` refers to `abc` in the locale whose name is the current value of `xy`. For compatibility with previous versions, the non-standard `abc__` is accepted and is the same as `abc_base_`. Thus:

```

b=: 1
Rome=: 2
Rome_NewYork_=: 20
f_NewYork_=: 3 : '3*b=: Rome+y.'
f_NewYork_ 10
90

b,Rome
1 2

b_NewYork_
30

```

A name is global if it is not assigned by `=.` within Explicit Definition (`:`). Every global name is executed in the current locale. Initially, the current locale is `base`. A locative `f_abc_`, while it is executing, switches the current locale to `abc`. The verb `18!:4` also switches the current locale, and `18!:5` gives its name.

The name `f_abc_` is *executed in* locale `abc` in the sense that a global name referenced in `f` is sought therein and, if not found, is then sought in the locales in the path of `abc` (but is still executed in `abc`). The path of a locale is initially `,<,'z'`, except that locale `z` has an empty path initially, and may be changed using `18!:2`.

A locale is commonly populated by a script, by appropriate naming of the verb used to execute the script. For example, if the file `stats` contains the script:

```

mean=: sum % #
sum=: +/

```

Then:

```

ssx_z_=: 0!:10          Silent script execution

ssx_a_ <'stats'          Populate locale  a

mean=: 'in base locale'
mean_a_ 3 4 5
4

ssx_bc_ <'stats'          Populate locale  bc
sum_bc_ 3 4 5
12

```

The example also illustrates the use of locale *paths*, in this case the `z` locale: First, the utility `ssx` is defined in the `z` locale. In executing `ssx_a_`, `ssx` is not found in locale `a` and is therefore sought (and found) in locale `z`. Since `ssx_a_` is *executed* in locale `a`, the names in the stats script are *defined* in locale `a`, populating it thereby. Similarly for `ssx_bc_`.

See also [18!: p227](#) in Appendix A and the "Locales" and "Object Oriented Programming" labs distributed with the system.

## J. Errors and Suspension

Execution of a sentence *suspends* when an error occurs, and an error message and context information are then displayed. Four blanks indicate where parsing stopped. Suspension may occur in immediate execution, in the execution of a script file, or in the execution of a user-defined named verb, adverb, or conjunction, as illustrated by the following examples:

### Immediate execution

```

2+'a'
|domain error
|  2    +'a'

```

### Execution of a script file

```

t=: '2*3',(10{a.),'2+'a''',(10{a.),'2+3'
t
2*3
2+'a'
2+3

```

A script

```
t 1!:2 <'test'
```

Write script file

```
0! :011 <'test'
```

Execute file, continue on error, display (011)

```
2*3
6
```

```
2+'a'
2+3
5
```

```
0! :001 <'test'
```

Execute file, stop on error, display (001)

```
2*3
6
```

```
2+'a'
```

```
|domain error
|  2    +'a'
|[-2]
```

## User-Defined Verb

```
g=: 3 : ('1+y.' ; ':' ; '2+x.+y.')

3+g 'a'
|domain error: g
|  1    +y.

13!:0 (1)                                Enable suspension
3+g 'a'
|domain error: g
|  1    +y.
|g[0]
```

```
      y.                                Six-space indent indicates suspension
a

      y.=. 12                            Redefine local value of y.

      13!:4 ''                            Resume execution at the current line
16   Result using redefined value of y.
```

Sentences can be executed in the suspended environment, local values can be accessed, and execution can be resumed. Errors cause suspension *only* if suspension is enabled (by the phrase `13!:0`1). When suspension is in effect, the input prompt is six spaces.

Suspension and debugging facilities are controlled by the [13!: p225](#) family as described in Appendix A.

### III. Definitions

Each main entry in the body of the dictionary is headed by a line beginning with the informal name of the monadic case of the function, and ending with the informal name of the dyadic case. The line also contains the formal name of the function, consisting of a single graphic, or a graphic modified by one or more following inflections (a period or colon). In the case of a conjunction, the formal name is preceded by *m* or by *u* (denoting a noun or a verb argument) and is followed by *n* or *v*. An adverb has no symbol to its right.

The three ranks (in the order *monadic*, *left*, and *right*) are also indicated, using the symbol *\_* for an infinite (unbounded) rank, and with ranks dependent on the ranks of argument verbs shown as *mu*, *lv*, etc.

Examples are provided with each definition, and the more complex of them may use auxiliary functions as yet unfamiliar. These examples may be approached by ignoring all but the immediately relevant aspects of such auxiliaries, and by examining (and perhaps entering for execution) component phrases that can be used to build up the final result.

For example, in the discussion of the adverb */*, the sentences below display various uses of it for convenient comparison:

```

x=: 1 2 3 4 5 [ y=: 7 5 3
( , .x ) ; ( x + / y ) ; y ; ( x * / y ) ; ( + / y ) ; ( * / y )
+---+-----+-----+-----+-----+
1	8  6 4	7 5 3	7  5  3	15	105
2	9  7 5		14 10  6		
3	10  8 6		21 15  9		
4	11  9 7		28 20 12		
5	12 10 8		35 25 15		
+---+-----+-----+-----+-----+

```

Even if the auxiliary functions *;* and *, .* are unfamiliar, their relevant effects are probably evident; if not, they may be clarified by the following experiments:

```

x ; y          , . 7 8          $ , . 7 8
+-----+-----+          7          2 1

```



8

Although a name (such as *foreign* for !:) is suggested for each word, others can be used in addition to or instead of them. Thus, *joy* might be used for ! since the exclamation mark derives from an I placed above an o, an abbreviation of the Latin *io*. Similarly, *iota* might be used instead of *integers* and *index of* for i. .

## Vocabulary ( [Constants p199](#) [Controls p200](#) [Foreigns p214](#) )

|                                                                           |                                                                      |                                                                      |
|---------------------------------------------------------------------------|----------------------------------------------------------------------|----------------------------------------------------------------------|
| <a href="#">= <u>Self-Classify</u> • <u>Equal</u> p75</a>                 | <a href="#">= . <u>Is (Local)</u> p76</a>                            | <a href="#">=: <u>Is (Global)</u> p76</a>                            |
| <a href="#">&lt; <u>Box</u> • <u>Less Than</u> p77</a>                    | <a href="#">&lt; . <u>Floor</u> • <u>Lesser Of (Min)</u> p78</a>     | <a href="#">&lt; : <u>Decrement</u> • <u>Less Or Equal</u> p79</a>   |
| <a href="#">&gt; <u>Open</u> • <u>Larger Than</u> p80</a>                 | <a href="#">&gt; . <u>Ceiling</u> • <u>Larger of (Max)</u> p81</a>   | <a href="#">&gt; : <u>Increment</u> • <u>Larger Or Equal</u> p82</a> |
| <a href="#">___ <u>Negative Sign</u> / <u>Infinity</u> p83</a>            | <a href="#">___ . <u>Indeterminate</u> p84</a>                       | <a href="#">___ : <u>Infinity</u> p85</a>                            |
| <a href="#">+ <u>Conjugate</u> • <u>Plus</u> p86</a>                      | <a href="#">+ . <u>Real / Imaginary</u> • <u>GCD (Or)</u> p87</a>    | <a href="#">+ : <u>Double</u> • <u>Not-Or</u> p88</a>                |
| <a href="#">* <u>Signum</u> • <u>Times</u> p89</a>                        | <a href="#">* . <u>Length/Angle</u> • <u>LCM (And)</u> p90</a>       | <a href="#">* : <u>Square</u> • <u>Not-And</u> p91</a>               |
| <a href="#">- <u>Negate</u> • <u>Minus</u> p92</a>                        | <a href="#">- . <u>Not</u> • <u>Less</u> p93</a>                     | <a href="#">- : <u>Halve</u> • <u>Match</u> p94</a>                  |
| <a href="#">% <u>Reciprocal</u> • <u>Divide</u> p95</a>                   | <a href="#">% . <u>Matrix Inverse</u> • <u>Matrix Divide</u> p96</a> | <a href="#">% : <u>Square Root</u> • <u>Root</u> p97</a>             |
| <a href="#">^ <u>Exponential</u> • <u>Power</u> p98</a>                   | <a href="#">^ . <u>Natural Log</u> • <u>Logarithm</u> p99</a>        | <a href="#">^ : <u>Power</u> p100</a>                                |
| <a href="#">\$ <u>Shape Of</u> • <u>Shape</u> p102</a>                    | <a href="#">\$ . <u>Sparse</u> p103</a>                              | <a href="#">\$ : <u>Self-Reference</u> p104</a>                      |
| <a href="#">~ <u>Reflex</u> • <u>Passive</u> / p105 <u>EVOKE</u> p106</a> | <a href="#">~ . <u>Nub</u> • p107</a>                                | <a href="#">~ : <u>Nub Sieve</u> • <u>Not-Equal</u> p108</a>         |
| <a href="#">  <u>Magnitude</u> • <u>Residue</u> p109</a>                  | <a href="#">  . <u>Reverse</u> • <u>Rotate (Shift)</u> p110</a>      | <a href="#">  : <u>Transpose</u> p111</a>                            |
| <a href="#">. <u>Determinant</u> • <u>Dot Product</u> p112</a>            | <a href="#">. . <u>Even</u> p113</a>                                 | <a href="#">. : <u>Odd</u> p113</a>                                  |
| <a href="#">: <u>Explicit</u> / p114 <u>Monad-Dyad</u> p115</a>           | <a href="#">: . <u>Obverse</u> p116</a>                              | <a href="#">: : <u>Adverse</u> p117</a>                              |
| <a href="#">, <u>Ravel</u> • <u>Append</u> p118</a>                       | <a href="#">, . <u>Ravel Items</u> • <u>Stitch</u> p119</a>          | <a href="#">, : <u>Itemize</u> • <u>Laminate</u> p120</a>            |
| <a href="#">; <u>Raze</u> • <u>Link</u> p121</a>                          | <a href="#">; . <u>Cut</u> p122</a>                                  | <a href="#">; : <u>Word Formation</u> • p123</a>                     |
| <a href="#"># <u>Tally</u> • <u>Copy</u> p124</a>                         | <a href="#"># . <u>Base 2</u> • <u>Base</u> p125</a>                 | <a href="#"># : <u>Antibase 2</u> • <u>Antibase</u> p126</a>         |
| <a href="#">! <u>Factorial</u> • <u>Out Of</u> p127</a>                   | <a href="#">! . <u>Fit (Customize)</u> p128</a>                      | <a href="#">! : <u>Foreign</u> p129</a>                              |

[/ \*Insert\* • \*Table\* p130](#)

[\ \*Prefix\* • \*Infix\* p133](#)

[\[ \*Same\* • \*Left\* p136](#)

[\] \*Same\* • \*Right\* p136](#)

[{ \*Catalogue\* • \*From\* p138](#)

[} \*Item Amend\* • \*Amend\* p142](#)

[" \*Rank\* p146](#)

[` \*Tie \(Gerund\)\* p151](#)

[@ \*Atop\* p153](#)

[& \*Bond\* / \*p156 Compose\* p157](#)

[? \*Roll\* • \*Deal\* p161](#)

[a. \*Alphabet\* p162](#)

[b. \*Boolean\* / \*p164 Basic\* p165](#)

[d. \*Derivative\* p168](#)

[e. \*Raze In\* • \*Member \(In\)\* p171](#)

[H. \*Hypergeometric\* p174](#)

[j. \*Imaginary\* • \*Complex\* p177](#)

[m. n. \*Explicit Noun Args\* p180](#)

[p. \*Polynomial\* p183](#)

[q: \*Prime Factors\* • \*Prime Exponents\* p186](#)

[S: \*Spread\* p189](#)

[/ . \*Oblique\* • \*Key\* p131](#)

[\ . \*Suffix\* • \*Outfix\* p134](#)

[{ . \*Head\* • \*Take\* p139](#)

[} . \*Behead\* • \*Drop\* p144](#)

[". \*Do\* • \*Numbers\* p149](#)

[@ . \*Agenda\* p154](#)

[& . \*p158\* & . : \*p159 Under \(Dual\)\* p158](#)

[? . \*Roll\* • \*Deal \(fixed seed\)\* p161](#)

[a: \*Ace\* \(Boxed Empty\) p162](#)

[c. \*Characteristic Values\* p166](#)

[D. \*Derivative\* p169](#)

[E. • \*Member of Interval\* p172](#)

[i. \*Integers\* • \*Index Of\* p175](#)

[L. \*Level Of\* p178](#)

[NB. \*Comment\* p181](#)

[p. . \*Poly. Deriv.\* • \*Poly. Integral\* p184](#)

[r. \*Angle\* • \*Polar\* p187](#)

[t. \*Taylor Coefficient\* p190](#)

[/ : \*Grade Up\* • \*Sort\* p132](#)

[\ : \*Grade Down\* • \*Sort\* p135](#)

[\[ : \*Cap\* p137](#)

[p140 { : \*Tail\* • p140](#)

[} : \*Curtail\* • p145](#)

[" : \*Default Format\* • \*Format\* p150](#)

[` : \*Evoke Gerund\* p152](#)

[@ : \*At\* p155](#)

[& : \*Appose\* p160](#)

[A. \*Anagram Index\* • \*Anagram\* p163](#)

[C. \*Cycle-Direct\* • \*Permute\* p167](#)

[D: \*Secant Slope\* p170](#)

[f. \*Fix\* p173](#)

[i: \*Integers\* • \*Index Of Last\* p176](#)

[L: \*Level At\* p179](#)

[o. \*Pi Times\* • \*Circle Function\* p182](#)

[p: \*Primes\* • p185](#)

[s: \*Symbol\* p188](#)

[t: \*Weighted Taylor\* p192](#)

[T. \*\*Taylor Approximation\*\* p193](#)

[x. y. Explicit Arguments p196](#)

[u. v. Explicit Verb Args p194](#)

[x: Extended Precision p197](#)

[u: Unicode p195](#)

[\\_9: to \\_9: Constant Functions p198](#)

**Self-Classify**

= \_ 0 0

**Equal**

`=y` classifies the items of the nub of `y` (that is, `~.y`) according to equality with the items of `y`, producing a boolean table of shape `#~.y` by `#y`. For example:

```
y=: 3 3 $ 'abcdef'
y i (~.y) i (=y)
+---+---+---+
abc	abc	1 0 1
def	def	0 1 0
abc		
+---+---+---+
```

`x=y` is 1 if `x` is equal to `y`, and is otherwise 0.

The comparison is made with a *tolerance* `t`, normally 2 to the power `_44` but also controlled by the fit conjunction `!.`, as in `x=!..0 y`. Formally, `x=y` is 1 if the magnitude of `x-y` does not exceed `t` times the larger of the magnitudes of `x` and `y`.

Tolerance applies similarly to other verbs as indicated for each, notably to Match (`-:`), to Floor (`<.`), and to Signum (`*`), but not to Grade (`/:`).

Both the monadic and dyadic cases of the verb `=` apply to nouns of any rank, and to boxed as well as simple nouns. For example:

```
]a=: ;: 'Try and try and try again.'
+---+---+---+---+---+---+
|Try|and|try|and|try|again.|
+---+---+---+---+---+---+
```

```
~. a
+---+---+---+---+
|Try|and|try|again.|
+---+---+---+---+
```

```
=a
1 0 0 0 0 0
0 1 0 1 0 0
0 0 1 0 1 0
0 0 0 0 0 1
```

```
a = <'and' 0 1 0 1 0 0
```

Because of the limited precision of the computer, results which should agree (such as  $144 \cdot (13 \div 144)$  and  $13$ ) may not; the tolerant comparison allows such a comparison to show agreement (a result 1). More or less stringent comparisons may be made by using the conjunction  $!.$  to specify a tolerance  $\epsilon$ , as in the function  $eq =: =!.\epsilon$ .

## Copula                      = .   Local                      = :   Global

The copula is used to assign a referent to a name, as in `a=:3` and in `sum=:+/.`. The copula `=.` is *local* as discussed under Explicit Definition (`:`), and `=:` is *global*. Copulas may also be used *indirectly*, with the name or names specified as a character list or a boxed list; moreover, if the character list begins with ``` then gerund referents are evoked.

For example:

```
f=: 3 : 0
a=. +:y.
b=: *:a
10*b
)
```

```
a=: b=: 678
a,b
678 678
```

```
f 3
360
```

```
a,b
678 36
```

```
x=: 'abc';'c'
(x) =: 3 4 ; 5 6 7
abc
3 4
c
5 6 7
```

Note that the parentheses around the name `x` force it to be evaluated before the assignment specified by the copula is effected.

```
'alpha beta'=: i.2 4
alpha
0 1 2 3
```

```

    beta
4 5 6 7

    '`sum sqrt'=: +/ ` %:
    sum 3 1 4 2
10
    sqrt 2
1.41421

```



**Box**

&lt; \_ 0 0

**Less Than**

$\langle y$  is an *atomic encoding* of  $y$ , as discussed in [Section II A p63](#). The result has rank 0, and is *decoded* by  $\rangle$ .

$x \langle y$  is 1 if  $x$  is tolerantly less than  $y$ . See Equal (=) for a definition of tolerance.  $\langle !.t$  uses tolerance  $t$ .

Boxing is also effected by verbs such as Link (;) and Word Formation (;:):

```
(<'abc'),(<5 7),(<i.2 3)
+---+---+---+
|abc|5 7|0 1 2|
|   |   |3 4 5|
+---+---+---+
```

```
;; 'Now is the time'
+---+---+---+
|Now|is|the|time|
+---+---+---+
```

```
] a=: 2;3 5;7 11 13
+---+---+---+
|2|3 5|7 11 13|
+---+---+---+
```

```
>a
2 0 0
3 5 0
7 11 13
```

Cut (;.) with < has several uses (chosen by the right argument); the phrase <@v avoids the padding (and some domain errors) that may result from applying v alone:

```
<:_1 '/i sing/of olaf/'
+---+---+---+
|i sing|of olaf||
+---+---+---+
```

```

      i."(0) 2 3 4
0 1 0 0
0 1 2 0
0 1 2 3

```

```

      <@i."(0) 2 3 4
+---+-----+-----+
|0 1|0 1 2|0 1 2 3|
+---+-----+-----+

```

If `y` is a high-rank array, `<"_1 y` or `<"_2 y` often gives a more intelligible display than `y` itself. The display of a boxed array would normally be corrupted by control characters (such as carriage returns and linefeeds) occurring therein; in the display such characters are replaced by spaces. For example, try `< 8 32 $ a.`

**Floor**

&lt;. 0 0 0

**Lesser Of (Min)**

$\<.\ y$  gives the *floor* of  $y$ , that is, the largest integer less than or equal to  $y$ . Thus:

```
<. 4.6 4 _4 _4.6
4 4 _4 _5
```

The implied comparison with integers is tolerant, as discussed under Equal (=), and is controlled by  $\<.!.\ t$ . See below for complex arguments.

$x\<.\ y$  is the lesser of  $x$  and  $y$ . For example:

```
3 <. 4 _4
3 _4
```

```
<./7 8 5 9 2
2
```

```
<./\7 8 5 9 2
7 7 5 5 2
```

For a complex argument, the definition of  $\<.$  is modelled by:

```
floor=: j./@ (ip+(c2>c1),c1+:c2)
'`c1 c2 fp ip'=: (1:>+/@fp)`(>:/@fp)`(+.-ip)`(<.@+.)
```

As developed by [McDonnell \[10\] p212](#), this function has the following properties:

**Convexity:** If  $(\<.z1)=(\<.z2)$  and  $z3$  lies on the line between  $z1$  to  $z2$ , then  $(\<z3)=(\<z1)$ .

**Translatability:** If  $z4$  is a Gaussian integer, then  $(z4+\<.z5)=(\<.z4+z5)$ .

**Compatibility:**  $(\<.x\ j.0)=((\<.x)j.0)$  and  $(\<.0\ j.x)=(0\ j.(\<.x))$

The function  $\<.$  can be viewed as a tiling by rectangles of unit area, all arguments within a rectangle sharing the same floor. One rectangle has vertices at  $1j0$  and  $0j1$ , with the other side passing through the origin. Rectangles along successive diagonals are displaced by one-half the length.

The phrase  $j./@ip$  "floors" the individual parts of a complex argument.

Moreover, the floor  $\<.\ y$  is equivalent to  $\rightarrow.-y$ . In other words, it is the dual of *ceiling* with respect to (that is, *under*) arithmetic negation:  $\<.\ \leftrightarrow\ >.\&.-$  and  $\>.\ \leftrightarrow\ <.\&.-$ . Thus:

```

      (>.&.- ; <.) 4.6 4 _4 _4.6
+-----+-----+
| 4 4 _4 _5 | 4 4 _4 _5 |
+-----+-----+

```

The expression `<.x+0.5` gives the integer *nearest* to the real argument `x`, and `<.z+0.5j0.5` gives the Gaussian integer nearest to `z`. The number of digits needed to represent an integer is given by one plus the floor of its base ten logarithm:

```

      a ,. ( ,. 1: + <.) 10^. a=: 9 10 11 99 100 101
      9 0.954243 1
10      1 2
11 1.04139 2
99 1.99564 2
100      2 3
101 2.00432 3

```

**Decrement**

&lt;: 0 0 0

**Less than or Equal**

<:y is y-1 . For example:

```

    <: 2 3 5 7
1 2 4 6

```

Also see Not (-.).

x<:y is 1 if x is less than or equal to y, and is otherwise 0 . See Equal (=) for a discussion of tolerance. The fit conjunction (!.) applies to <: .

The inverse of <: is >: (Increment). For example:

```

n=: 5
    <: ^: _1 n
6

```

```

    <: ^: 0 1 2 n
2)
5 4 3

```

Here ^: applies to a noun right argument (0 1

```

    <: ^: i. n
5 4 3 2 1

```

Here ^: applies to a verb right argument (i.)

```

    */ <: ^: i. n
120

```

```

f=: */ @ (<: ^: i.)
f n
120

```

```

f"0 i. n
1 1 2 6 24

```

```

(f"0 = !) i. n
1 1 1 1 1

```

```

<:/ ~ i. 5
1 1 1 1 1
0 1 1 1 1
0 0 1 1 1

```

Table of the dyad <:

|   |   |   |   |   |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 |

**Open**

&gt; 0 0 0

**Larger Than**

Open is the inverse of box, that is,  $><y$  is  $y$ . When applied to an open array (that has no boxed elements), open has no effect. Opened atoms are brought to a common shape as discussed in [Section II B p64](#).

$x>y$  is 1 if  $x$  is tolerantly larger than  $y$ . See Equal (=) for a discussion of tolerance. For example:

```

      1 2 3 4 5 > 5 4 3 2 1
0 0 0 1 1

```

Tolerance  $t$  is provided by  $>!.t$ .

Since the rank of open is 0, it applies to each atom of its argument. For example:

```

]a=: 1 2 3;4 5 6;7 8 9
+-----+-----+-----+
|1 2 3|4 5 6|7 8 9|
+-----+-----+-----+

>a
1 2 3
4 5 6
7 8 9

```

Results of different shapes are padded as defined in [Section II B p64](#). For example:

```

(>1;2 3;4 5 6); (>'a';'bc';'def'); (<\i.4); (><\i.4)
+-----+-----+-----+-----+
1 0 0	a	+-+---+-----+-----+	0 0 0 0					
2 3 0	bc		0	0 1	0 1 2	0 1 2 3		0 1 0 0
4 5 6	def	+-+---+-----+-----+	0 1 2 0					
			0 1 2 3					
+-----+-----+-----+-----+

```

```

</~ i.5
0 1 1 1 1
0 0 1 1 1
0 0 0 1 1
0 0 0 0 1
0 0 0 0 0

```

Table of the dyad  $<$ :

1 < 1+10<sup>-8</sup>+i.15                      Tolerant comparison  
1 1 1 1 1 1 0 0 0 0 0 0 0 0 0

1 <!. (0) 1+10<sup>-8</sup>+i.15                      Exact comparison (0-tolerance)  
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0



**Ceiling**

&gt;. 0 0 0

**Larger Of (Max)**

`>.y` gives the *ceiling* of `y`, that is, the smallest integer greater than or equal to `y`. Thus:

```
>. 4.6 4 _4 _4.6
5 4 _4 _4
```

The implied comparison with integers is tolerant, as discussed under `Equal (=)`, and is controlled by `>.!t`. See `Floor (<.)` and [McDonnell \[10\] p212](#) for complex arguments.

`x>.y` is the larger of `x` and `y`. For example:

```
3>.4 _4
4 3

>./7 8 5 9 2
9

>./\7 8 5 9 2
7 8 8 9 9
```

The comparison `x = >. x` determines whether `x` is an integer. Thus:

```
Integer_test=: ] = >.
Integer_test 3 3.14 _5
1 0 1
```

See the definition of `fork` in [Section II F p68](#).

```
f=: = >.
f 3 3.14 _5
1 0 1
```

The same function may be defined by a hook.

The *ceiling* `>. y` is equivalent to `-<.-y`. In other words, it is the dual of *floor* with respect to (that is, *under*) arithmetic negation: `>. ↔ <.&.-` and `<. ↔ >.&.-`. For example:

```
(<.&.- ; >.) 4.6 4 _4 _4.6
+-----+-----+
| 5 4 _4 _4 | 5 4 _4 _4 |
+-----+-----+
```

**Increment****>: 0 0 0****Larger or Equal**

>:y is y+1 . For example:

```
>: 2 3 5 7
3 4 6 8
```

Also see Not (-.).

x>:y is 1 if x is tolerantly greater than or equal to y.

See Equal (=) for a discussion of tolerance. >:!.t uses tolerance t .

```
+ : i. 6
0 2 4 6 8 10
```

Even numbers

```
> : + : i. 6
1 3 5 7 9 11
```

Odd numbers

```
odds=: >:@+:@i.
odds 10
1 3 5 7 9 11 13 15 17 19
```

```
+ / odds 10
100
```

```
(+/@odds , *: ) 10
100 100
```

Sum of first n odds equals the square of n

```
>: / ~ i. 5
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 1 1 0
1 1 1 1 1
```

Table of the dyad >:

## Negative Sign and Infinity

The symbol `_` followed by a digit denotes a negative number (as in `_3.4`), and denotes infinity when used alone, or negative infinity (in `__`). It is also used in names, as discussed in [Part I p61](#) and in [Part II Section I p71](#).

For example:

|                                                                       |                            |
|-----------------------------------------------------------------------|----------------------------|
| <code>2 % 0</code>                                                    | Two divided by zero        |
| <hr/>                                                                 |                            |
| <code>10 ^. 0</code>                                                  | Base ten logarithm of zero |
| <hr/>                                                                 |                            |
| <code>_2 _ 3 + 5</code><br><code>3 _ 8</code>                         |                            |
| <code>integer_test=: =&lt;.</code><br><code>integer_test 3 3.5</code> | Use of break in name       |
| <code>1 0</code>                                                      |                            |

Although `-2` may sometimes be used instead of `_2`, it is important to understand that the former is the application of a function to the number `2`, whereas the symbol `_` is an indivisible part of the number representation, just as the period is an indivisible part of a number such as `8.9`.

## Indeterminate

— .

The indeterminate  $\infty - \infty$  results from expressions such as  $\infty - \infty$  (infinity minus infinity) and from expressions (such as  $3 + \infty$ ) in which an indeterminate argument occurs.

**Infinity**

$$\_ : \_ \_ \_$$
**Infinity**

$\_ :$  is a *constant* function that yields an infinite result, that is,  $\_ : y$  is  $\_$

$\_ :$  is a *constant* function that yields an infinite result, that is,  $x \_ : y$  is  $\_$

For example:

```
y=: 1 2 3 4
_ : y
```

—

```
_ : "0 y
```

Rank zero applies to each element

```
— — — —
```

Other constant functions include  $\_9 :$  and  $\_8 :$  etc. to  $9 :$ . More generally, the expression  $x " r$  defines a constant function of rank  $r$  that yields the constant value  $x$ . For example:

```
3.14"0 y
3.14 3.14 3.14 3.14
```

```
3.14"1 y
3.14
```

The specific constant functions mentioned can therefore be written alternatively as  $\_ "$  and  $\_9 "$  and  $0 "$  and  $9 "$ , etc.

**Conjugate**

+ 0 0 0

**Plus**

$+$   $y$  is the *conjugate* of  $y$ . For example,  $+3j4$  is  $3j\_4$ .

$+$  is defined as in elementary arithmetic, and is extended to complex numbers as usual.

A complex number  $y$  multiplied by its conjugate produces a real number equal to the square of its magnitude  $|y|$ . For example:

```
3j4 * 3j_4
25
```

The function `j.` multiplies its argument by the square root of negative one:

```
j i=: i. 5
0 1 2 3 4
```

```
j. i
0 0j1 0j2 0j3 0j4
```

```
y=: i + 2 * j. i
0 1j2 2j4 3j6 4j8
```

```
+y
0 1j_2 2j_4 3j_6 4j_8
```

```
y * +y
0 5 20 45 80
```

```
%: y * +y
0 2.23607 4.47214 6.7082 8.94427
```

```
|y
0 2.23607 4.47214 6.7082 8.94427
```

The conjugate of  $y$  can also be expressed as  $(|y*y)\%y$ . For example:

```
(|y*y)%y
0 1j_2 2j_4 3j_6 4j_8
```

**Real / Imaginary**

+. 0 0 0

**GCD (Or)**

`+.y` yields a two-element list of the real and imaginary parts of its argument. For example, `+.3j5` is `3 5`, and `+.3` is `3 0`.

`x+.y` is the *greatest common divisor* of `x` and `y`. If the arguments are boolean (0 or 1), the functions `+.`  and `*.`  are equivalent to logical *or* and *and*. The function `-.`  similarly restricted is *not*.

```

      ly=: i+2*j. i=: i.4
0 1j2 2j4 3j6

      +. y
0 0
1 2
2 4
3 6

```

The greatest common divisor divides both of its arguments `x` and `y` to produce results that have no common factor, that is, the GCD of the quotients is 1. Moreover, these quotients represent the fraction `x%y` in lowest form. For example:

```

      x=: 24 [ y=: 60
      x;y;(x +. y);((x , y) % (x +. y))
+---+---+---+---+
|24|60|12|2 5|
+---+---+---+---+

      lff=: , % +.
      x;y;(x lff y);(%/x lff y);(%/x,y);(+./x lff y)
+---+---+---+---+---+
|24|60|2 5|0.4|0.4|1|
+---+---+---+---+---+

```

Gives lowest form of fraction

Since the functions `=|` and `=<.` (tests for non-negative and for integer) produce boolean results, the phrase `(=|)+.(=<.)` is a test for non-negative *or* integer:

```

      (test=: (=|) +. (=<.) ) _2 _2.4 3 3.5
1 0 1 1

```

The duality of *or* and *and* may be shown as follows:

```

      d (+./ ; *.&.-./ ; *./ ; +.&.-./) d=: 0 1
+---+---+---+---+
| 0 1| 0 1| 0 0| 0 0|
| 1 1| 1 1| 0 1| 0 1|
+---+---+---+---+

```



**Double**

+ : 0 0 0

**Not-Or**

+ : y is twice y . For example:

```
+ : 3 0 _2
6 0 _4
```

x + : y is the negation of x *or* y .  
For example, 0 + : 0 is 1 .

Since the square of the sum of two arguments equals the sum of their squares and *twice* their product, the following functions are equivalent:

```
f = : + * +
g = : *:@[ + +:@* + *:@]
```

For example:

```
x = : 7 6 3 [ y = : 6 5 3
x (f ; g ; (f=g) ; (f-:g)) y
+-----+-----+-----+
|169 121 36|169 121 36|1 1 1|1|
+-----+-----+-----+

```

Since the domain of not-or is limited to zero and one, its entire behaviour can be seen in the following function tables:

```
d = : 0 1
d + : / d
1 0
0 0
```

Domain of nor  
Table of nor

```
d +. / d
0 1
1 1
```

Table of or

```
- . d +. / d
1 0
0 0
```

Negation of table of or

```
(+ :&.-. / ~d) ; (* : / ~d)
+---+---+
```

Nand and nor are duals under not

|           |   |   |   |  |
|-----------|---|---|---|--|
| 1         | 1 | 1 | 1 |  |
| 1         | 0 | 1 | 0 |  |
| +---+---+ |   |   |   |  |

**Signum**

\* 0 0 0

**Times**

\*<sub>y</sub> is <sub>-1</sub> if <sub>y</sub> is negative, 0 if it is zero, 1 if it is positive; more generally, \*<sub>y</sub> is the intersection of the unit circle with the line from the origin through the argument <sub>y</sub> in the complex plane. For example:

```
*_3 0 5 3j4
_1 0 1 0.6j0.8
```

The comparison with zero is tolerant, as defined by the phrase (<sub>y</sub>%|<sub>y</sub>)\*<sub>t</sub><:|<sub>y</sub> where <sub>t</sub> denotes the tolerance. The fit conjunction applies to signum, as in \*!.<sub>t</sub>.

\* denotes multiplication, defined as in elementary mathematics and extended to complex numbers as usual:

```
t=:+.x,y [ x=:2j4 [ y=:5j3
r=-:*/t [ i=:+/ . * t
(x,:y);t;r;i;(r j. i);(x*y)
+-----+
|2j4|2 4|_2|26|_2j26|_2j26|
|5j3|5 3| | | | |
+-----+
```

Signum is useful in effecting selections. For example:

```
* y=: _4 0 4
_1 0 1
```

```
>:@* y
0 1 2
```

```
f=: %:
f ^: * " 0 y
16 0 2
```

Inverse of <sub>f</sub>, Identity, or <sub>f</sub>

```
(* y) { ;:'Yes No Maybe'
+-----+
|Maybe|Yes|No|
+-----+
```

Select using indexing ({ )

```
g=: <:~-:~+:@.*"0
g y
_8 _1 2
```

See Agenda (@.)

The dyad `*` used on a list and a table illustrates the significance of *agreement*, as discussed in [Section II B p64](#):

```

m=: i. 3 4 [ v=: 3 2 1

m ; (v*m) ; (m*v) ; (+/ m*v) ; (v +/ . * m)
+-----+-----+-----+-----+-----+
0 1  2  3	0  3  6  9	0  3  6  9	16 22 28 34	16 22 28 34
4 5  6  7	8 10 12 14	8 10 12 14		
8 9 10 11	8  9 10 11	8  9 10 11		
+-----+-----+-----+-----+-----+

```

**Length / Angle**

\* . 0 0 0

**LCM (And)**

`*.y` is a two-element list of the length and angle (in radians) of the hypotenuse of a triangle with base and altitude equal to the real and imaginary parts `y`. For example, `*. 3j4` is `5 0.927295`.

`x*.y` is the least common multiple of `x` and `y`. For boolean arguments (0 and 1) it is equivalent to *and*. Thus:

```
0 1 *. / 0 1
0 0
0 1
```

Some properties of the length / angle are illustrated in the following, including the fact that the length (i.e. magnitude) of the product of two complex numbers is the product of their lengths, and the angle of the product is the sum of their angles:

```
( | ; *. ; r./@*.) y=: 3j4
+-+-----+----+
|5|5 0.927295|3j4|
+-+-----+----+
```

```
x=: 2j_6
*. x,y
6.32456 _1.24905
5 0.927295
```

Polar coordinates

```
f=: */@:({."1) , +/@:({."1)
last
f *. x , y
31.6228 _0.321751
```

Product over first col and sum over

```
*. x * y
31.6228 _0.321751
```

Length and angle of product

The least common multiple is the product divided by the GCD. For example:

```
24 (+. ; *. ; */ % +.) 60
+-+-----+----+
|12|120|120|
+-+-----+----+
```

**Square**

\*: 0 0 0

**Not-And**

\*: y is the square of y .

x \*: y is the negation of x *and* y .  
For example 0 \*: 0 is 1 .

The inverse of the square is the square root. For example:

```

*: ^: _1 (_2 _1 0 1 2)
0j1.41421 0j1 0 1 1.41421

```

```

3 +&.*: 4
5

```

Hypotenuse of triangle with sides 3 and 4

Since the domain of nand is limited to zero and one, its entire behaviour can be seen in the following function tables:

```

d=: 0 1
d *: / d
1 1
1 0

```

Domain of nand  
Table of nand

```

d *./ d
0 0
0 1

```

Table of and

```

-. d *./ d
1 1
1 0

```

Nand, Not and, and the dual of Nor all agree, as illustrated below:

```

(*:/~ ; -.@*./~ ; +:&.-./~) d
+---+---+---+
| 1 1 | 1 1 | 1 1 |
| 1 0 | 1 0 | 1 0 |
+---+---+---+

```

**Negate**

- 0 0 0

**Minus**

$-y$  is the negative of  $y$ . That is, it is defined as  $0 - y$ . Thus,  $-2$  is  $0 - 2$ .

$-$  is defined as in elementary arithmetic, and is extended to complex numbers as usual.

The function  $-$  is self-inverse, that is,  $-^{-1}$  is  $-$  itself.

Although  $-2$  may often be used instead of  $0 - 2$ , it is important to understand that the former is the application of a function to the number  $2$ , whereas the symbol  $-$  is an indivisible part of the number representation, just as the period is an indivisible part of a number such as  $8.9$ .

**Not**

- . 0 \_ \_

**Less**

`- . y` is `1-y`; for a boolean argument it is the complement (not); for a probability, it is the complementary probability.

`x- . y` includes all items of `x` except for those that are cells of `y`.

Tolerance `t` is provided by `- . ! . t`.

The function *less* applies to any conformable pair of arguments. For example:

```
(i. 9) - . 2 3 5 7
0 1 4 6 8
```

```
'abcdefghij' - . 'aeiou'
bcdfghj
```

```
]m=: i. 4 5
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```

```
m - . 5 6 7 8 9
0 1 2 3 4
10 11 12 13 14
15 16 17 18 19
```

```
b=: <\ 'abcdefg'
b
+--+---+---+---+---+---+---+
|a|ab|abc|abcd|abcde|abcdef|abcdefg|
+--+---+---+---+---+---+---+
```

```
b - . 'abc';'abcde';'cba'
+--+---+---+---+---+---+
|a|ab|abcd|abcdef|abcdefg|
+--+---+---+---+---+---+
```

```
2 3 4 5 - . 'abcdef'
2 3 4 5
```



**Halve**

-: 0 \_ \_

**Match**

-:  $y$  is one half of  $y$ . For example:

```
-: i. 5
0 0.5 1 1.5 2
```

$x -: y$  yields 1 if its arguments match: in shapes, boxing, and elements; but using tolerant comparison. See Equal (=).

Matching with a tolerance  $t$  can be obtained using the verb  $-: !. t$ .

For example:

```
x=: 0 1 2 3 4 5
, .&. > ( ] ; -: ; +:@ -: ; (%&2) ; (2: %~ ])) x
+-+----+-+----+-+----+
0	0	0	0	0
1	0.5	1	0.5	0.5
2	1	2	1	1
3	1.5	3	1.5	1.5
4	2	4	2	2
5	2.5	5	2.5	2.5
+-+----+-+----+-+----+

x = +: -: x
1 1 1 1 1 1

x -: +: -: x
1
```

**Reciprocal****%**    0   0   0**Divided by**

`% y` is the reciprocal of `y`, that is, `1/y`. For example, `%4 ↔ 0.25`.

`x % y` is division of `x` by `y` as defined in elementary math, except that `0%0` is `0`. See [McDonnell \[11\] p212](#), and the resulting pattern in the middle column and middle row of the table below.

We will illustrate the divide function by tables, using a function to generate lists symmetric about zero:

```
sym=: i.@>:@+:- ]          Symmetric integers
] a=: sym 3
_3 _2 _1 0 1 2 3
```

```
(] ; *) |. a%/a
+-----+-----+
_1 _1.5 _3  _  3  1.5      1	_1 _1 _1  1  1  1  1
_0.666667  _1 _2  _  2    1  0.666667	_1 _1 _1  1  1  1  1
_0.333333 _0.5 _1  _  1  0.5  0.333333	_1 _1 _1  1  1  1  1
0    0  0  0  0    0      0	0  0  0  0  0  0  0
0.333333  0.5  1  _  _1 _0.5 _0.333333	1  1  1 _1 _1 _1 _1
0.666667    1  2  _  _2  _1 _0.666667	1  1  1 _1 _1 _1 _1
1  1.5  3  _  _3 _1.5      _1	1  1  1 _1 _1 _1 _1
+-----+-----+
```

```
6j2 " : |. a %/ a
_1.00 _1.50 _3.00  _  3.00  1.50  1.00
_0.67 _1.00 _2.00  _  2.00  1.00  0.67
_0.33 _0.50 _1.00  _  1.00  0.50  0.33
 0.00  0.00  0.00  0.00  0.00  0.00  0.00
 0.33  0.50  1.00  _  _1.00 _0.50 _0.33
 0.67  1.00  2.00  _  _2.00 _1.00 _0.67
 1.00  1.50  3.00  _  _3.00 _1.50 _1.00
```

The final use of the format function gives a more readable result, with a width of six spaces per column and a uniform two digits after the decimal point.

```

|. a %/ x: a
 _1 _3r2 _3 _ 3 3r2 1
_2r3 _1 _2 _ 2 1 2r3
_1r3 _1r2 _1 _ 1 1r2 1r3
  0 0 0 0 0 0 0
1r3 1r2 1 __ _1 _1r2 _1r3
2r3 1 2 __ _2 _1 _2r3
  1 3r2 3 __ _3 _3r2 _1

```

## Matrix Inverse

`%.` 2 \_ 2

## Matrix Divide

If  $y$  is a non-singular matrix, then  $\% . y$  is the inverse of  $y$ . For example:

```
mp=: +/ . *           Matrix
product
(% . ; ] ; % . mp ] ) i. 2 2
+-----+-----+
|_1.5 0.5|0 1|1 0|
|   1   0|2 3|0 1|
+-----+-----+
```

More generally,  $\% . y$  is defined in terms of the dyadic case, with the left argument  $= i. \{ : \$ y$  (an identity matrix) or, equally, by the relation  $(\% . y) mp x \leftrightarrow x \% . y$ .

The shape of  $\% . y$  is  $| . \$ y$ .

The vector and scalar cases are defined by using the matrix  $\% . y$ , but the shape of the result is  $\$ y$ .

For a non-zero vector  $y$ , the result of  $\% . y$  is a vector collinear with  $y$  whose length is the reciprocal of that of  $y$ ; it is called the reflection of  $y$  in the unit circle (or sphere). Thus:

```
(% . , : ] % % .) 2 3 4
0.0689655 0.103448 0.137931
      29      29      29
```

If  $y$  is non-singular, then  $x \% . y$  is  $(\% . y) mp x$ . More generally, if the columns of  $y$  are linearly independent and if  $\#x$  and  $\#y$  agree, then  $x \% . y$  minimizes the difference:

$$d = \| x - y mp x \% . y \|$$

in the sense that the magnitudes  $\|d\| + \|d\|$  are minimized. Scalar and vector cases of  $y$  are treated as the one-column matrix  $\% . y$ .

Geometrically,  $y mp x \% . y$  is the *projection* of the vector  $x$  on the column space of  $y$ , the point nearest to  $x$  in the space spanned by the columns of  $y$ .

Common uses of  $\% .$  are in the solution of linear equations and in the approximation of functions by polynomials, as in  $c =: (f x) \% . x ^ / i. 4$ .

We will illustrate the use of  $\% .$  in function fitting by the sine function, showing,

in particular, the maximum over the magnitudes of the differences from the function being approximated:

```

sin=: 1&o.                                Function to be approximated
x=: 5 %~ i. 6
c=: (sin x) % . x ^/ i.4                  Use of matrix divide
,.&.>@[ ] ; c"_ i sin ; c&p. ; >./@:|@(sin-c&p.)) x
+---+-----+-----+-----+-----+
0	_5.30503e_5	0	_5.30503e_5	0.000167992
0.2	1.00384	0.198669	0.198826	
0.4	_0.018453	0.389418	0.389321	
0.6	_0.143922	0.564642	0.564523	
0.8		0.717356	0.717524	
1		0.841471	0.841416	
+---+-----+-----+-----+-----+

```

**Square Root****%:**    0   0   0**Root**

**%:**  $y$  is the square root of  $y$ . If  $y$  is negative, the result is an imaginary number. For example,  $\%:-4 \leftrightarrow 0j2$ .

**x %:**  $y$  is the  $x$  root of  $y$ . Thus,  $3\%:8$  is 2, and  $2\%:y$  is  $\%:y$ . In general,  $x \%: y \leftrightarrow y^{1/x}$ .

For example:

```

y=: i. 7
y
0 1 2 3 4 5 6

2 %: y
0 1 1.41421 1.73205 2 2.23607 2.44949

%: y
0 1 1.41421 1.73205 2 2.23607 2.44949

r=: 1 2 3 4
z=: r %:/ y
z
0 1      2      3      4      5      6
0 1 1.41421 1.73205      2 2.23607 2.44949
0 1 1.25992 1.44225 1.5874 1.70998 1.81712
0 1 1.18921 1.31607 1.41421 1.49535 1.56508

r ^~ z
~
0 1 2 3 4 5 6
0 1 2 3 4 5 6
0 1 2 3 4 5 6
0 1 2 3 4 5 6

```

See *agreement* in [Section II B p64](#), and note use of

**Exponential** $\hat{\phantom{x}} \_ 0 \ 0$ **Power**

$\hat{y}$  is equivalent to  $e^y$ , where  $e$  is *Euler's number*  $\hat{1}$  (approximately 2.71828). The *natural logarithm* ( $\wedge$ ) is inverse to  $\hat{\phantom{x}}$  (that is,  $y = \wedge. \hat{y}$  and  $y = \hat{\wedge. y}$ ).

The monad  $x \wedge$  is inverse to the monad  $x \hat{\phantom{x}}$ . For example:

```
10 &^ 10 &^ . 1 2 3 4 5
1 2 3 4 5
```

```
10 &^ . 10 &^ 1 2 3 4 5
1 2 3 4 5
```

$x^2$  and  $x^3$  and  $x^{0.5}$  are the *square*, *cube*, and *square root* of  $x$ .

In general,  $x^y$  is  $\hat{y} \hat{\phantom{x}}$ , applying for complex numbers as well as real.

For a non-negative integer  $y$ , the phrase  $x \hat{\phantom{x}} y$  is equivalent to  $*/y \# x$ ; in particular,  $*/$  on an empty list is 1, and  $x^0$  is 1 for any  $x$ , including 0.

The fit conjunction applies to  $\hat{\phantom{x}}$  to yield a stope defined as follows:  $x! . k$  is  $*/x + k * i. n$ . In particular,  $\hat{!} . _1$  is the *falling factorial* function.

The last result in the first example below illustrates the falling factorial function, formed by the fit conjunction. See [Chapter 4 of \[8\] p212](#) for the use of stope functions, stope polynomials, and Stirling numbers in the difference calculus:

```
e=: ^ 1 [ x=: 4 [ y=: 0 1 2 3
, .&.> x (e"_ ; e&^@] ; ^ ; ^@([ * ^.@]) ; ([^]) ; ^!._1) y

+-----+-----+---+---+---+---+
2.71828	1	1	_	1	1
	2.71828	4	1	1	4
	7.38906	16	4	4	12
	20.0855	64	27	27	24
+-----+-----+---+---+---+---+

S2=: %.@S1=: (^!._1/~ %. ^/~) @ i. @ x:
(S1;S2) 8

+-----+-----+-----+-----+
1 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0 1 _1 2 _6 24 _120 720	0 1 1 1 1 1 1 1
0 0 1 _3 11 _50 274 _1764	0 0 1 3 7 15 31 63
```

|               |   |   |   |    |     |      |      |   |   |   |   |   |    |    |     |  |
|---------------|---|---|---|----|-----|------|------|---|---|---|---|---|----|----|-----|--|
| 0             | 0 | 0 | 1 | _6 | 35  | _225 | 1624 | 0 | 0 | 0 | 1 | 6 | 25 | 90 | 301 |  |
| 0             | 0 | 0 | 0 | 1  | _10 | 85   | _735 | 0 | 0 | 0 | 0 | 1 | 10 | 65 | 350 |  |
| 0             | 0 | 0 | 0 | 0  | 1   | _15  | 175  | 0 | 0 | 0 | 0 | 0 | 1  | 15 | 140 |  |
| 0             | 0 | 0 | 0 | 0  | 0   | 1    | _21  | 0 | 0 | 0 | 0 | 0 | 0  | 1  | 21  |  |
| 0             | 0 | 0 | 0 | 0  | 0   | 0    | 1    | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 1   |  |
| +-----+-----+ |   |   |   |    |     |      |      |   |   |   |   |   |    |    |     |  |

s1 gives (signed) Stirling numbers of the first kind and s2 gives Stirling numbers of the second kind. They can be used to transform between ordinary and stope polynomials. Note that x: gives extended precision.



**Natural Log**

^ . 0 0 0

**Logarithm**

The *natural logarithm* (^.) is inverse to the exponential ^ (i.e.,  $y = ^.^y$  and  $y = ^.^{.^y}$ ).

The *base-x logarithm*  $x^{.^y}$  is the inverse of power (^) in the sense that  $y = x^{.^{x^y}}$  and  $y = x^{.^{x^{.^y}}}$ .

Certain properties of logarithms are illustrated below:

```

x=: 4 [ y=: 0 1 2 3
(x^y) ; (x^{.^y}) ; (x^{.^y}) ; (x^{.^{.^y}})
+-----+-----+-----+-----+
| 1 4 16 64 | 0 1 2 3 | _ 0 0.5 0.792481 | 0 1 2 3 |
+-----+-----+-----+-----+

logtable=: ^./~@i.
<6j2 ": logtable 6
+-----+
|  _ .  0.00  0.00  0.00  0.00  0.00 |
|  _ _  0.00      _      _      _      _ |
|  _ _  0.00  1.00  1.58  2.00  2.32 |
|  _ _  0.00  0.63  1.00  1.26  1.46 |
|  _ _  0.00  0.50  0.79  1.00  1.16 |
|  _ _  0.00  0.43  0.68  0.86  1.00 |
+-----+

```

The first derivative of the natural logarithm is the reciprocal. For example:

```

^ . d. 1 y=: 0 1 2 3 4 5 6
_ 1 0.5 0.333333 0.25 0.2 0.166667

% ^ . d. 1 y
0 1 2 3 4 5 6

```

**Power**  $u^{:n}$     \_ \_ \_

Two cases occur: a numeric integer  $n$ , and a gerund  $n$ .

**Numeric case.** The verb  $u$  (or  $x\&u$ ) is applied  $n$  times. For example:

( ] ; +/\ ; +/\^:2 ; +/\^:0 1 2 3 \_1 \_2 \_3 \_4) 1 2 3 4 5

|   |   |   |   |   |   |   |   |    |    |   |   |    |    |    |   |    |    |    |    |
|---|---|---|---|---|---|---|---|----|----|---|---|----|----|----|---|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 6 | 10 | 15 | 1 | 4 | 10 | 20 | 35 | 1 | 2  | 3  | 4  | 5  |
|   |   |   |   |   |   |   |   |    |    |   |   |    |    |    | 1 | 3  | 6  | 10 | 15 |
|   |   |   |   |   |   |   |   |    |    |   |   |    |    |    | 1 | 4  | 10 | 20 | 35 |
|   |   |   |   |   |   |   |   |    |    |   |   |    |    |    | 1 | 5  | 15 | 35 | 70 |
|   |   |   |   |   |   |   |   |    |    |   |   |    |    |    | 1 | 1  | 1  | 1  | 1  |
|   |   |   |   |   |   |   |   |    |    |   |   |    |    |    | 1 | 0  | 0  | 0  | 0  |
|   |   |   |   |   |   |   |   |    |    |   |   |    |    |    | 1 | _1 | 0  | 0  | 0  |
|   |   |   |   |   |   |   |   |    |    |   |   |    |    |    | 1 | _2 | 1  | 0  | 0  |

An infinite power  $n$  produces the limit of the application of  $u$ . For example, if  $x=:2$  and  $y=:1$ , then  $x\ o.^:_y$  is 0.73908, the solution of the equation  $y=\cos y$ . If  $n$  is negative, the obverse  $u^:_1$  is applied  $|n|$  times. The obverse (which is normally the inverse) is specified for six cases:

- 1. The self-inverse functions  $+ - -. \% \%. |. |: /: [ ] C. p.$
- 2. The pairs in the following tables:

|                    |           |                                |
|--------------------|-----------|--------------------------------|
| < >                | !         | 3 : '(-(!-y."_)%1e_3&* !"0 D:1 |
| <: >:              |           | ])^:_^.y.'                     |
| + . j./"1"__       | 3!:1 3!:2 |                                |
| +: -:              | 3!:3 3!:2 |                                |
| *. r./"1"__        | \:        | /:@ .                          |
| *: %:              | ".        | ":                             |
| ^ ^.               | j.        | %&0j1                          |
| \$ . \$.^:_1       | o.        | %&1p1                          |
| ,: {.              | p:        | $\pi(n)$                       |
| ;: ;@(',&' '&.>"1) | q:        | */                             |
| #. #:              | r.        | %&0j1@^.                       |
|                    | s:        | 5&s:                           |
|                    | u:        | 3&s:                           |
|                    | x:        | _1&x:                          |

+~ -:  
 \*~ %:  
 ^~ 3 : '(- -&b@(\*^.) % >:@^.)^:\_ b=.^.y.'"0  
 ,~ <.@-:@# { . ]  
 ,:~ { .  
 ;~ >@{ .  
 j.~ %&1j1

3. Obviously invertible bonded dyads such as  $-&3$  and  $10\&^.$  and  $1\ 0\ 2\&|:$  and  $3\&|.$  and  $1\&o.$  and  $a.\&i.$  as well as  $u@v$  and  $u\&v$  if  $u$  and  $v$  are invertible.

4. Monads of the form  $v/\backslash$  and  $v/\backslash.$  where  $v$  is one of  $+ \ * \ - \ \% \ = \ \sim :$

5. Obverses specified by the conjunction  $:.$

6. The following cases merit special mention:

$p:^:_1\ n$  gives the number of primes less than  $n$ , denoted by  $\pi(n)$  in math  
 $q:^:_1$  is  $*/$

$b\&\#^:_1$  where  $b$  is a boolean list is "Expand" (whose fill atom  $f$  can be specified by *fit*,  $b\&\#^:_1!.f$ )

$a\&\#.^:_1$  produces the base- $a$  representation

$!^:_1$  and  $!\&n^:_1$  and  $!\&n\&^:_1$  produce the appropriate results

**Gerund case.** (Compare with the gerund case of the merge adverb  $\}$ )

$$\begin{aligned} x\ u^:(v0\`v1\`v2)y &\leftrightarrow (x\ v0\ y)u^:(x\ v1\ y)\ (x\ v2\ y) \\ x\ u^:(\quad v1\`v2)y &\leftrightarrow x\ u^:([\`v1\`v2)\ y \\ u^:(\quad v1\`v2)y &\leftrightarrow u^:(v1\ y)\ (v2\ y) \end{aligned}$$

**Power** $u^{\wedge}:v \quad \_ \_ \_$ 

The case of  $^{\wedge}:$  with a verb right argument is defined in terms of the noun right argument case ( $u^{\wedge}:n$ ) as follows:

$$\begin{aligned} x \ u^{\wedge}:v \ y &\leftrightarrow x \ u^{\wedge}:(x \ v \ y) \ y \\ u^{\wedge}:v \ y &\leftrightarrow u^{\wedge}:(v \ y) \ y \end{aligned}$$

For example:

```

x=: 1 3 3 1
y=: 0 1 2 3 4 5 6
x p. y
1 8 27 64 125 216 343

x p. ^: (]>3:)"1 0 y
0 1 2 3 125 216 343

a=: _3 _2 _1 0 1 2 3
%: a
0j1.73205 0j1.41421 0j1 0 1 1.41421 1.73205

* a
_1 _1 _1 0 1 1 1

%: ^: * " 0 a
9 4 1 0 1 1.41421 1.73205

*: a
9 4 1 0 1 4 9

```

The following monads are equivalent. (See the example of  $^{\wedge} \ T. \_$  in the definition of the Taylor Approximation  $T. .$ )

```

g=: u ^: p ^: _
h=: 3 : 't=. y. while. p t do. t=. u t end.'

u=: -&3
p=: 0&<
(g"0 ; h"0) i. 10

```

|         |   |    |    |   |    |    |   |    |    |         |  |   |    |    |   |    |    |   |  |
|---------|---|----|----|---|----|----|---|----|----|---------|--|---|----|----|---|----|----|---|--|
| +-----+ |   |    |    |   |    |    |   |    |    | +-----+ |  |   |    |    |   |    |    |   |  |
|         | 0 | _2 | _1 | 0 | _2 | _1 | 0 | _2 | _1 | 0       |  | 0 | _2 | _1 | 0 | _2 | _1 | 0 |  |
| +-----+ |   |    |    |   |    |    |   |    |    | +-----+ |  |   |    |    |   |    |    |   |  |

## Shape Of

\$ \_ 1 \_

## Shape

\$ *y* yields the shape of *y* as defined in [Section II A p63](#). For example, the shape of a 2-by-3 matrix is 2 3, and the shape of the scalar 3 is an empty list (whose shape is 0).

The rank of an argument *y* is #@ \$ *y*. For example:

```
rank=: #@ $
(rank 3) , (rank ,3)
0 1

(rank 3 4) , (rank i. 2 3 4)
1 3
```

The shape of *x*\$*y* is *x*,*siy* where *siy* is the shape of an item of *y*; *x*\$*y* gives a length error if *y* is empty and *x*,*siy* does not contain a zero. For example:

```
y=: 3 4$'abcdefghijkl'
y ; 2 2$ y
+-----+-----+
abcd	abcd
efgh	efgh
ijkl	
	ijkl
	abcd
+-----+-----+
```

This example shows how the result is formed from the *items* of *y*, the last 1-cell (abcd) showing that the selection is cyclic. The fit conjunction (\$!.f) provides fill specified by the items of *f*.

Since *x* \$ *y* uses *items* from *y*, it is sometimes useful to ravel the right argument, as in *x* \$ ,*y*. For example (using the *y* defined above):

```
2 3 $ ,y
abc
def
```

The fit conjunction is often useful for appending zeros or spaces. For example:

```
8 $!.0 (2 3 4)
2 3 4 0 0 0 0 0

]z=: 8$!.'*' 'abc'
```

abc\*\*\*\*\*

    |. z  
\*\*\*\*\*cba

    2 5\$!.a: ;: 'zero one two three four five six'  
+-----+-----+-----+-----+-----+  
|zero|one|two|three|four|  
+-----+-----+-----+-----+-----+  
|five|six|   |   |   |   |  
+-----+-----+-----+-----+-----+



## Sparse

`$.`    `— — —`

## Sparse

`$.y` converts a dense array to sparse, and conversely `$.^:_1 y` converts a sparse array to dense.

The identities `f -> f&$.` and `f -> f&.(^:_1)` hold for any function `f`, with the possible exception of those (like `overtake { . }`) which use the sparse element as the fill.

`0$.y` applies `$.` or `$.^:_1` as appropriate; that is, converts a dense array to sparse and a sparse array to dense.

`1$.sh;a;e` produces a sparse array. `sh` specifies the shape. `a` specifies the sparse axes; negative indexing may be used. `e` specifies the "zero" element, and its type determines the type of the array. The argument may also be `sh;a` (`e` is assumed to be a floating point 0) or just `sh` (`a` is assumed to be `i.#sh` -- all axes are sparse -- and `e` a floating point 0).

`2$.y` gives the sparse axes (the `a` part);

`(2;a)$.y` (re-)specifies the sparse axes;

`(2 1;a)$.y` gives the number of bytes required for `(2;a)$.y`;

`(2 2;a)$.y` gives the number of items in the `i` part or the `x` part for the specified sparse axes `a` (that is, `#4$. (2;a)$.y`).

`3$.y` gives the sparse element (the `e` part); `(3;e)$.y` respecifies the sparse element.

`4$.y` gives the index matrix (the `i` part).

**5\$.y** gives the value array (the `x` part).

**6\$.y** gives the flag (the `flag` part).

**7\$.y** gives the number of non-sparse entries in array `y`; that is, `#4$.y` or `#5$.y`.

**8\$.y** removes any completely "zero" value cells and the corresponding rows in the index matrix.

The inverse of `n&$.` is `(-n)&$. .`

The remainder of this text is divided into the following sections: Introduction, Representation, Assertions, Further Examples, Sparse Linear Algebra, and Implementation Status.

## Introduction

We describe a sparse array extension to J using a representation that "does not store zeros". One new verb `$.` is defined to create and manipulate sparse arrays, and existing primitives are extended to operate on such arrays. These ideas are illustrated in following examples:

```
] d=: (? . 3 4$2) * ? . 3 4$100
0 75 0 53
0 0 67 67
93 0 51 83
```

```
] s=: $. d
0 1 | 75
0 3 | 53
1 2 | 67
1 3 | 67
2 0 | 93
2 2 | 51
2 3 | 83
```

convert `d` to sparse and assign to `s`

the display of `s` gives the indices of the "non-zero" cells and the corresponding values

```

d -: s
1

```

d and s match

```

o. s
0 1 | 235.619
0 3 | 166.504
1 2 | 210.487
1 3 | 210.487
2 0 | 292.168
2 2 | 160.221
2 3 | 260.752

```

$\pi$  times s

```

o. d
0 235.619
0
292.168
0 210.487 210.487
0 160.221 260.752

```

$\pi$  times d

```

(o. s) -: o. d
1

```

function results independent of representation

```

0.5 + o. s
0 1 | 236.119
0 3 | 167.004
1 2 | 210.987
1 3 | 210.987
2 0 | 292.668
2 2 | 160.721
2 3 | 261.252

```

```

<. 0.5 + o. s
0 1 | 236
0 3 | 167
1 2 | 210
1 3 | 210
2 0 | 292
2 2 | 160
2 3 | 261

```

```

(<. 0.5 + o. s) -: <. 0.5 + o. d
1

```

```

d + s
0 1 | 150
0 3 | 106
1 2 | 134

```

function arguments can be dense or sparse

|   |   |  |     |
|---|---|--|-----|
| 1 | 3 |  | 134 |
| 2 | 0 |  | 186 |
| 2 | 2 |  | 102 |
| 2 | 3 |  | 166 |

```
(d + s) -: 2*s
1
```

familiar algebraic properties are preserved

```
(d + s) -: 2*d
1
```

|   |      |     |
|---|------|-----|
|   | +/ s |     |
| 0 |      | 93  |
| 1 |      | 75  |
| 2 |      | 118 |
| 3 |      | 203 |

|   |         |     |
|---|---------|-----|
|   | +/ "1 s |     |
| 0 |         | 128 |
| 1 |         | 134 |
| 2 |         | 227 |

|   |   |   |    |
|---|---|---|----|
|   | . | s |    |
| 0 | 0 |   | 93 |
| 0 | 2 |   | 51 |
| 0 | 3 |   | 83 |
| 1 | 2 |   | 67 |
| 1 | 3 |   | 67 |
| 2 | 1 |   | 75 |
| 2 | 3 |   | 53 |

reverse

|   |   |      |    |
|---|---|------|----|
|   | . | "1 s |    |
| 0 | 0 |      | 53 |
| 0 | 2 |      | 75 |
| 1 | 0 |      | 67 |
| 1 | 1 |      | 67 |
| 2 | 0 |      | 83 |
| 2 | 1 |      | 51 |
| 2 | 3 |      | 93 |

|   |   |   |    |
|---|---|---|----|
|   | : | s |    |
| 0 | 2 |   | 93 |
| 1 | 0 |   | 75 |
| 2 | 1 |   | 67 |
| 2 | 2 |   | 51 |
| 3 | 0 |   | 53 |

transpose

```

3 1 | 67
3 2 | 83

```

```

$ | : s
4 3

```

```

$.^:_1 | : s
0 0 93
75 0 0
0 67 51
53 67 83

```

`$.^:_1` converts a sparse array to dense

```

(| : s) -: | : d
1

```

`ravel`; a sparse *vector*

```

, s
1 | 75
3 | 53
6 | 67
7 | 67
8 | 93
10 | 51
11 | 83

```

```

$ , s
12

```

## Representation

A sparse array `y` may be boolean, integer, floating point, complex, literal, or boxed, and has the (internal) parts `sh;a;e;i;x;flag` where:

- `sh` Shape, `$y`. Elements of the shape must be less than  $2^{31}$ , but the product over the shape may be larger than  $2^{31}$ .
- `a` Axe(s), a vector of the sorted sparse (indexed) axes.
- `e` Sparse element ("zero"). `e` is also used as the fill in any overtake of the array.
- `i` Indices, an integer matrix of indices for the sparse axes.
- `x` Values, a (dense) array of usually non-zero cells for the non-sparse axes corresponding to the index matrix `i`.
- `flag` Various bit flags.

For the sparse matrix `s` used in the introduction,

```
] d=: (? . 3 4$2) * ? . 3 4$100
0 75  0 53
0  0 67 67
93  0 51 83
```

```
] s=: $. d
0 1 | 75
0 3 | 53
1 2 | 67
1 3 | 67
2 0 | 93
2 2 | 51
2 3 | 83
```

The shape is `3 4`; the sparse axes are `0 1`; the sparse element is `0`; the indices are the first two columns of numbers in the display of `s`; and the values are the last column.

Scalars continue to be represented as before (densely). All primitives accept sparse or dense arrays as arguments (e.g. `sparse+dense` or `sparse$sparse`). The display of a sparse array is a display of the index matrix (the `i` part), a blank column, a column of vertical lines, another blank column, and the corresponding value cells (the `x` part).

Letting the sparse element be variable rather than fixed at zero makes many more functions closed on sparse arrays (e.g. `^y` or `10+y`), and familiar results can be produced by familiar phrases (e.g. `<.0.5+y` for rounding to the nearest integer).

## Assertions

The following assertions hold for a sparse array, and displaying a sparse array invokes these consistency checks on it.

|                              |                              |
|------------------------------|------------------------------|
| <code>imax =: _1+2^31</code> | the largest internal integer |
| <code>rank =: #@\$</code>    | rank                         |
| <code>type =: 3!:0</code>    | internal type                |
| <code>l = rank sh</code>     | vector                       |
| <code>sh -: &lt;. sh</code>  | integral                     |

|                                           |                                                                       |
|-------------------------------------------|-----------------------------------------------------------------------|
| <code>imax &gt;: #sh</code>               | at most <code>imax</code> elements                                    |
| <code>(0&lt;:sh) *. (sh&lt;:imax)</code>  | bounded by 0 and <code>imax</code>                                    |
| <code>1 = rank a</code>                   | vector                                                                |
| <code>a e. i.#sh</code>                   | bounded by 0 and rank-1                                               |
| <code>a -: ~. a</code>                    | elements are unique                                                   |
| <code>a -: /:~ a</code>                   | sorted                                                                |
| <code>0 = rank e</code>                   | atomic                                                                |
| <code>(type e) = type x</code>            | has the same internal type as <code>x</code>                          |
| <code>2 = rank i</code>                   | matrix                                                                |
| <code>4 = type i</code>                   | integral                                                              |
| <code>(#i) = #x</code>                    | as many rows as the number of items in <code>x</code>                 |
| <code>({: \$i) = #a</code>                | as many columns as there are sparse axes                              |
| <code>(#i) &lt;: */a{sh</code>            | # rows bounded by product over sparse axes lengths                    |
| <code>imax &gt;: */\$i</code>             | # elements is bounded by <code>imax</code>                            |
| <code>(0&lt;:i) *. (i &lt;"1 a{sh)</code> | <code>i</code> bounded by 0 and the lengths of the sparse axes        |
| <code>i -: ~.i</code>                     | rows are unique                                                       |
| <code>i -: /:~ i</code>                   | rows are sorted                                                       |
| <code>(rank x) = 1+(#sh)-#a</code>        | rank equals 1 plus the number of dense axes                           |
| <code>imax &gt;: */\$x</code>             | # elements is bounded by <code>imax</code>                            |
| <code>(}. \$x)-:((i.#sh)-.a){s</code>     | item shape is the dimensions of the dense axes                        |
| <code>(type x) e. 1 2 4 8 16 32</code>    | internal type is boolean, character, integer, real, complex, or boxed |

## Further Examples

```

] d=: (0=? . 2 3 4$3) * ? . 2 3 4$100
13 0 0 0
21 4 0 0
0 0 0 0

3 5 0 0

```

```
0 0 6 0
0 0 0 0
```

```
] s=: $. d
```

```
0 0 0 | 13
0 1 0 | 21
0 1 1 | 4
1 0 0 | 3
1 0 1 | 5
1 1 2 | 6
```

```
d -: s
1
```

```
2 $. s
0 1 2
```

```
3 $. s
0
```

```
4 $. s
axes
```

```
0 1 0
0 1 1
1 0 0
1 0 1
1 1 2
```

```
5 $. s
13 21 4 3 5 6
```

```
] u=: (2;2)$.s
```

```
0 | 13 21 0
  | 3 0 0
  |
1 | 0 4 0
  | 5 0 0
  |
2 | 0 0 0
  | 0 6 0
```

```
4 $. u
0
1
2
```

convert d to sparse and assign to s

match is independent of representation

sparse axes

sparse element

index matrix; columns correspond to the sparse

corresponding values

make 2 be the sparse axis

index matrix



```

      5 $. u
13 21 0
3  0 0

```

corresponding values

```

0  4 0
5  0 0

```

```

0  0 0
0  6 0

```

```

      ] t=: (2;0 1)$.s
0 0 | 13 0 0 0
0 1 | 21 4 0 0
1 0 |  3 5 0 0
1 1 |  0 0 6 0

```

make 0 1 be the sparse axes

```

      7 {. t
0 0 | 13 0 0 0
0 1 | 21 4 0 0
1 0 |  3 5 0 0
1 1 |  0 0 6 0

```

take

```

      $ 7 {. t
7 3 4

```

```

      7{."1 t
0 0 | 13 0 0 0 0 0 0
0 1 | 21 4 0 0 0 0 0
1 0 |  3 5 0 0 0 0 0
1 1 |  0 0 6 0 0 0 0

```

take with rank

```

      0 = t
0 0 | 0 1 1 1
0 1 | 0 0 1 1
1 0 | 0 0 1 1
1 1 | 1 1 0 1

```

```

      3 $. 0 = t
1

```

the sparse element of 0=t is 1

```

      +/ , 0 = t
18

```

```

      +/ , 0 = d
18

```

answers are independent of representation

```

0 { t
0 | 13 0 0 0
1 | 21 4 0 0

```

from

```

_2 (<1 2 3)}t
0 0 | 13 0 0 0
0 1 | 21 4 0 0
1 0 | 3 5 0 0
1 1 | 0 0 6 0
1 2 | 0 0 0 _2

```

amend

```

s=: 1 $. 20 50 1000 75 366
$ s
20 50 1000 75 366

```

20 countries, 50 regions, 1000 salespersons,  
75 products, 366 days in a year

```

*/ $ s
2^31
2.745e10

```

the product over the shape can be greater than

```

r=: ?. 1e5 $ 1e6
i=: ?. 1e5 5 $ $ s
s=: r (<"1 i)} s

```

revenues  
corresponding locations  
assign revenues to corresponding locations

```

7 {. " : s
0 0 5 30 267 | 128133
0 0 26 20 162 | 319804
0 0 31 37 211 | 349445
0 0 37 10 351 | 765935
0 0 56 6 67 | 457449
0 0 66 54 120 | 38186
0 0 71 74 246 | 515473

```

the first 7 rows in the display of s  
the first row says that for country 0, region 0,  
salesperson 5, product 30, day 267,  
the revenue was 128133

```

+/, s
|limit error
| +/, s

```

total revenue  
the expression failed on ,s because it would  
have required a vector of length 2.745e10

```

+/@, s
5.00289e10

```

total revenue  
f/,@, is supported by special code

```

+//+//+//+ s
5.00289e10

```

total revenue

```

+/^:5 s
5.00289e10

```

```

+/^:_ s

```

5.00289e10

+ / r  
5.00289e10

+ / "1^:4 s                      total revenue by country

|    |  |           |
|----|--|-----------|
| 0  |  | 2.48411e9 |
| 1  |  | 2.55592e9 |
| 2  |  | 2.55103e9 |
| 3  |  | 2.52089e9 |
| 4  |  | 2.49225e9 |
| 5  |  | 2.45682e9 |
| 6  |  | 2.52786e9 |
| 7  |  | 2.45425e9 |
| 8  |  | 2.48729e9 |
| 9  |  | 2.50094e9 |
| 10 |  | 2.51109e9 |
| 11 |  | 2.59601e9 |
| 12 |  | 2.49003e9 |
| 13 |  | 2.58199e9 |
| 14 |  | 2.44772e9 |
| 15 |  | 2.47863e9 |
| 16 |  | 2.46455e9 |
| 17 |  | 2.5568e9  |
| 18 |  | 2.43492e9 |
| 19 |  | 2.43582e9 |

t=: + / ^:2 + / "1^:2 s                      total revenue by salesperson

\$t

1000

7 { .t

|   |  |           |
|---|--|-----------|
| 0 |  | 4.58962e7 |
| 1 |  | 4.81548e7 |
| 2 |  | 3.97248e7 |
| 3 |  | 4.89981e7 |
| 4 |  | 4.85948e7 |
| 5 |  | 4.69227e7 |
| 6 |  | 4.22094e7 |

## Sparse Linear Algebra

Currently, only sparse matrix multiplication and the solutions of tri-diagonal linear system are implemented. For example:

```
f=: }. @ }:@ (,/ ) @ ( ,."_1 +/&_1 0 1) @ i.
```

```
f 5
```

indices for a 5 by 5 tri-diagonal matrix

```
0 0
0 1
1 0
1 1
1 2
2 1
2 2
2 3
3 2
3 3
3 4
4 3
4 4
```

```
s=: (? . 13$100) (<"1 f 5)} 1 $. 5 5;0 1
$s
```

```
5 5
```

The phrase `1$.5 5;0 1` makes a sparse array with shape `5 5` and sparse axes `0 1` (sparse in both dimensions); `<"1 f 5` makes boxed indices; and `x (<"1 f 5)}y` amends by `x` the locations in `y` indicated by the indices (scattered amendment).

```
s
```

```
0 0 | 13
0 1 | 75
1 0 | 45
1 1 | 53
1 2 | 21
2 1 | 4
2 2 | 67
2 3 | 67
3 2 | 93
3 3 | 38
3 4 | 51
4 3 | 83
4 4 | 3
```

```
] d=: $.^:_1 s
```

the dense representation of `s`

```
13 75 0 0 0
45 53 21 0 0
```

```
0 4 67 67 0
0 0 93 38 51
0 0 0 83 3
```

```
] y=: ?. 5$80
10 60 36 42 17
```

```
y %. s
1.27885 _0.0883347 0.339681 0.202906 0.0529263
```

```
y %. d                                answers are independent of representation
1.27885 _0.0883347 0.339681 0.202906 0.0529263
```

```
s=: (?. (_2+3*1e5)$1000) (<"1 f 1e5)} 1 $. 1e5 1e5;0 1
```

```
$ s                                s is a 1e5 by 1e5 matrix
100000 100000
```

```
y=: ?. 1e5$1000
```

```
ts=: 6!:2 , 7!:2@]                time and space for execution
```

```
ts 'y %. s'
0.28 5.2439e6                    0.28 seconds; 5.2 megabytes (Pentium 266 Mhz)
```

## Implementation Status

As of 2002-06-27, the following facilities support sparse arrays:

|    |     |     |
|----|-----|-----|
| =  | =.  | =:  |
| <  | <.  | <:  |
| >  | >.  | >:  |
| —  | —.  | —:  |
| +  | +.  | +:  |
| *  | *.  | *:  |
| —  | —.  | —:  |
| %  | %.  | %:  |
| ^  | ^.  |     |
| \$ | \$. | \$: |
| ~  | ~.  | ~:  |
|    | .   | :   |
|    | ..  | ..: |

```

:      :.      ::
,      ,.      ,:
      ;.

#
!      !.      !:
/ m    /. d    /: m
\ m    \. m    \: m

[      [ :
]
{ d    { .      { :
} d    }.      }:

"      ".      ": m
`      `:

@      @.      @:
&      &.      &:

e. d
i.
i:
j.
o.
r.
_9: to 9:

3!:0
3!:1
3!:2
3!:3
4!:55

```

## Notes:

- Sparse literal and boxed arrays not yet implemented.
- The dyad `%.` only implements the case of triadiagonal matrices.
- Boxed left arguments for `|:` (diagonal slices) not yet implemented.
- The monads `f/` and `f/"r` are only implemented for `+ * > . < . +. *. = ~:`, (and only boolean arguments for `=` and `~:`); on an axis of length 2, the monads `f/` and `f/"r` are implemented for any function.
- The monads `f/@`, (and `f/@:`, and `f/&`, and `f/&:`,) are supported by special code.
- `{` and `}` only accept the following index arguments: integer arrays, `<"1`

on integer arrays, and scalar boxed indices (respectively, item indexing, scattered indexing, and index lists `a0;a1;a2;...`); and (`{` only) sparse arrays.

## Self Reference

\$ :    —   —   —

\$ : denotes the longest verb that contains it. For example:

```
1: `( ] * $:@<: )@. * 5
120
```

In the foregoing expression, the agenda (@.) chooses the verb ] \* \$:@<: as long as the argument (reduced by one each time by the application of the decrement) remains non-zero. When the argument becomes zero, the result of the right argument of @. is zero, and the constant function 1: is chosen.

If \$:@ were omitted from the expression, it would execute once only as follows:

```
1: `( ] * <: )@. * 5
20
```

The inclusion of self-reference ensures that the entire function is re-executed after decrementing the argument.

$\nabla \text{ f.}$  will not fix any part of  $\nabla$  that contains \$: .



Reflexive

$u \sim \_ \text{ru } l u$

Passive

$u \sim y \leftrightarrow y u y$ . For example,  $\wedge \sim 3$  is 27, and  $+/\sim i. n$  is an addition table.

$\sim$  *commutes* or *crosses* connections to arguments:  $x u \sim y \leftrightarrow y u x$ .

Certain uses of the reflexive and passive are illustrated below:

```
x=: 1 2 3 4 [ y=: 4 5 6
x (,.@[ ; ^/ ; ^/~ ; ^/~@[ ; ]) y

+--+-----+-----+-----+-----+
1	1      1      1	4 16   64   256	1  1  1  1	4 5 6
2	16     32     64	5 25  125  625	2  4  8  16	
3	81    243    729	6 36  216 1296	3  9 27  81	
4	256 1024 4096		4 16 64 256	
+--+-----+-----+-----+-----+

into=: %~
(i. 6) % 5
0 0.2 0.4 0.6 0.8 1

5 into i. 6
0 0.2 0.4 0.6 0.8 1

from=: -~
(i.6) - 5
_5 _4 _3 _2 _1 0

5 from i.6
_5 _4 _3 _2 _1 0

(x %/ y);(x %~/ y);(x %/~ y)

+-----+-----+-----+-----+
0.25 0.2 0.166667	4      5  6	4  2 1.33333  1
0.5 0.4 0.333333	2      2.5 3	5 2.5 1.66667 1.25
0.75 0.6      0.5	1.33333 1.66667 2	6  3      2  1.5
1 0.8 0.666667	1      1.25 1.5	
+-----+-----+-----+-----+
```

**Evoke**
 $m \sim \_$ 

If  $m$  is a name, then  $'m' \sim$  is equivalent to  $m$ . For example:

```
m=: 2 3 4
```

```
'm'~
```

```
2 3 4
```

```
m=: +/
```

```
'm'~ 2 3 5 7
```

```
17
```

```
m=: /
```

```
+ 'm'~ 2 3 5 7
```

```
17
```

## Nub

~. \_

`~.y` selects the *nub* of `y`, that is, all of its distinct items. For example:

```
y=: 3 3 $ 'ABCABCDEF'
y ; (~.y) ; (~.3) ; ($~.3)

+---+---+---+
ABC	ABC	3	1
ABC	DEF		
DEF			
+---+---+---+
```

More precisely, the nub is found by selecting the leading item, suppressing from the argument all items tolerantly equal to it, selecting the next remaining item, and so on. The `fit` conjunction applies to nub to specify the tolerance used.

If `f` is a costly function, it may be quicker to evaluate `f y` by first evaluating `f~. y` (which yields all of the distinct results required), and then distributing them to their appropriate positions. The inner product with the self-classification table (produced by `=`) can be used to effect this distribution. For example:

```
f=: *:
f y=: 2 7 1 8 2 8 1 8
4 49 1 64 4 64 1 64

, .&. > (~. ; f@~. ; = ; (f@~. (+/.*)=) ; f) y

+---+---+---+---+---+---+---+
2	4	1 0 0 0 1 0 0 0	4	4				
7	49	0 1 0 0 0 0 0 0	49	49				
1	1	0 0 1 0 0 0 1 0	1	1				
8	64	0 0 0 1 0 1 0 1	64	64				
							4	4
							64	64
							1	1
							64	64
+---+---+---+---+---+---+---+
```

```
NUB=: 1 : 'x.@~. +/ . * ='
*: NUB y
4 49 1 64 4 64 1 64
```

Adverb

```
nubindex=: ~. i. ]
(nubindex ; (nubindex { ~.)) y
+-----+-----+
|0 1 2 3 0 3 2 3|2 7 1 8 2 8 1 8|
+-----+-----+
```

**Nub Sieve** $\sim: \quad \_ \quad 0 \quad 0$ **Not Equal**

$\sim:y$  is the boolean list  $b$  such that  $b\#y$  is the nub of  $y$ . For example:

```

~: 'Mississippi'
1 1 1 0 0 0 0 0 1 0 0

```

$x\sim:y$  is 1 if  $x$  is tolerantly unequal to  $y$ . See Equal (=).

The fit conjunction may be used to specify tolerance, as in  $\sim:!.t$ .

The result of nub-sieve can be used to select the nub as follows:

```

y=: 8 1 8 2 8 1 7 2
~. y
8 1 2 7

```

```

~: y
1 1 0 1 0 0 1 0

```

```

(~: y) # y
8 1 2 7

```

```

y #~ ~: y
8 1 2 7

```

The dyad  $\sim:$  applies to any argument, but for booleans it is called *exclusive-or*. For example:

```

d=: 0 1
d ~:/ d
0 1
1 0

```

Not-equal, not equal, and the dual of equal with respect to not, all agree as illustrated below.

```

(~:/ ; -.@=/ ; =&.-./)~ d
+---+---+---+
| 0 1|0 1|0 1|
| 1 0|1 0|1 0|
+---+---+---+

```

**Magnitude**

| 0 0 0

**Residue**

$|y \leftrightarrow \%:y*+y$ . For example:

6 6 5 | 6 \_6 3j4

The familiar use of residue is in determining the remainder on dividing a non-negative integer by a positive:

3 | 0 1 2 3 4 5 6 7  
0 1 2 0 1 2 0 1

The definition  $y-x*<. y \% x+0=x$  extends the residue to a zero left argument, and to negative and fractional arguments. For example:

```
over =: ({ . ,.@; }.)@":@,
by    =: ' ' &i@, .@[ ,. ]
```

```
x=: 3 2 1 0 _1 _2 _3
y=: 0 1 2 3 4 5 6 7 8
```

x by y over x | / y

```
+---+-----+
| | 0 1 2 3 4 5 6 7 8 |
+---+-----+
3	0 1 2 0 1 2 0 1 2
2	0 1 0 1 0 1 0 1 0
1	0 0 0 0 0 0 0 0 0
0	0 1 2 3 4 5 6 7 8
_1	0 0 0 0 0 0 0 0 0
_2	0 _1 0 _1 0 _1 0 _1 0
_3	0 _2 _1 0 _2 _1 0 _2 _1
+---+-----+
```

To produce a true zero for cases such as  $(\%3) | (2\%3)$  the residue is made tolerant as shown in the definition of `res` below:

```
res=: f`g@.agenda"0
agenda=: ([ = 0:) +. (<. = >.)@S
S=: ] % [ + [ = 0:
f=: ] - [ * <.@S
g=: ] * [ = 0:
```

```

0.1 res 2.5 3.64 2 _1.6
0 0.04 0 0

```

```

(, . ; res/~ ; |/~) a=: 2 -- i.5
+---+-----+-----+
_2	0 _1 0 _1 0	0 _1 0 _1 0
_1	0  0 0  0 0	0  0 0  0 0
0	_2 _1 0  1 2	_2 _1 0  1 2
1	0  0 0  0 0	0  0 0  0 0
2	0  1 0  1 0	0  1 0  1 0
+---+-----+-----+

```

The dyad `|` applies to complex numbers. Moreover, the fit conjunction may be applied to control the tolerance used. The dyad `m&|@^` on integer arguments is computed in a way that avoids large intermediate numbers. For example: `2 (1e6&|@^) 10^100x`

**Reverse**

| . \_ 1 \_

**Rotate**

| . y reverses the order of the items of y . For example:

```
| . t=: 'abcdefg'
gfedcba
```

The *right shift* is the dyadic case of | . ! . f with the left argument \_1 . For example:

```
| . ! . '# ' t
#abcdef

| . ! . 10 i . 3 3
10 10 10
0 1 2
3 4 5
```

x | . y rotates successive axes of y by successive elements of x . Thus:

```
1 2 | . i . 3 5
7 8 9 5 6
12 13 14 10 11
2 3 4 0 1
```

The phrase x | . ! . f y produces a *shift*: the items normally brought around by the cyclic rotation are replaced by f unless f is empty (0=#f), in which case they are replaced by the normal fill defined under { . (take):

```
2 _2 | . ! . '# '"0 1 t
cdefg##
##abcde
```

```
y=: a.{~ (a. i. 'A') + i. 5 6
```

```
( ] ; 2&| . ; _2&| . ; 2&| ."1 ; 2&( | . ! . '*' "1)) y
+-----+-----+-----+-----+-----+
ABCDEF	MNOPQR	STUVWX	CDEFAB	CDEF**
GHIJKL	STUVWX	YZ[\]^	IJKLGH	IJKL**
MNOPQR	YZ[\]^	ABCDEF	OPQRMN	OPQR**
STUVWX	ABCDEF	GHIJKL	UVWXST	UVWX**
YZ[\]^	GHIJKL	MNOPQR	[\]^YZ	[\]^**
+-----+-----+-----+-----+-----+
```

```
( ] ; | . ; | ."1 ; | . ! . '*' "1 ; (2: | . ])) y
+-----+-----+-----+-----+-----+
| ABCDEF | YZ[\]^ | FEDCBA | *ABCDE | MNOPQR |
| GHIJKL | STUVWX | LKJIHG | *GHIJK | STUVWX |
```



|         |         |         |         |         |
|---------|---------|---------|---------|---------|
| MNOPQR  | MNOPQR  | RQPONM  | *MNOPQ  | YZ[\]^  |
| STUVWX  | GHIJKL  | XWVUTS  | *STUVW  | ABCDEF  |
| YZ[\]^  | ABCDEF  | ^]\[ZY  | *YZ[\]  | GHIJKL  |
| +-----+ | +-----+ | +-----+ | +-----+ | +-----+ |

1\_2|.!. '\*' 3{. y  
\*\*GHIJ  
\*\*MNOP  
\*\*\*\*\*

**Transpose**

| : \_ 1 \_

**Transpose**

| : reverses the order of the axes of its argument. For example:

```
(| ; |:) i. 3 4
+-----+-----+
0 1 2 3	0 4 8
4 5 6 7	1 5 9
8 9 10 11	2 6 10
	3 7 11
+-----+-----+
```

$x|:y$  moves axes  $x$  to the tail end. If  $x$  is boxed, the axes in each box are *run together* to produce a single axis:

```
y=: 3 4$'abcdefghijkl'
y;(1 0|:y);(0|:y);((<0
1)|:y)
+-----+-----+-----+-----+
abcd	aei	aei	afk
efgh	bfj	bfj	
ijkl	cgk	cgk	
	dhl	dhl	
+-----+-----+-----+-----+
```

For example:

```
y=: a.{~ (a. i. 'a') + i. 2 3 4
z=: y;(2 1 |: y);((<2 1) |: y);(|: i. 4 5)
z ,&< |:&.> z
+-----+-----+-----+-----+
+-----+-----+-----+-----+	+-----+-----+-----+-----+									
	abcd	aei	afk	0 5 10 15		am	am	am	0 1 2 3 4	
	efgh	bfj	mrw	1 6 11 16		eq	bn	fr	5 6 7 8 9	
	ijkl	cgk		2 7 12 17		iu	co	kw	10 11 12 13 14	
		dhl		3 8 13 18			dp		15 16 17 18 19	
	mnop			4 9 14 19		bn				
	qrst	mqu				fr	eq			
	uvwx	nrw				jv	fr			
		osw				gs				
		ptx				co	ht			
+-----+-----+-----+-----+	gs									
						kw	iu			
							jv			
						dp	kw			
						ht	lx			
						lx				
+-----+-----+-----+-----+	+-----+-----+-----+-----+									
```

+-----+-----+

## Determinant

 $u \cdot v \quad 2 \quad \_ \quad \_$ 

## Dot Product

The phrases  $-/ \cdot *$  and  $+/ \cdot *$  are the *determinant* and *permanent* of square matrix arguments. More generally, the phrase  $u \cdot v$  is defined in terms of a recursive expansion by minors along the first column, as discussed below.

For vectors and matrices, the phrase  $x \cdot y$  is equivalent to the *dot*, *inner*, or *matrix* product of math; other rank-0 verbs such as  $< \cdot$  and  $* \cdot$  are treated analogously. In general,  $u \cdot v$  is defined by  $u@(v"(1+1v, \_))$ , restated in English below.

For example:

```
x=: 1 2 3 [ m=: >1 6 4;4 1 0;6 6 8
det=: -/ . *
mp=: +/ . *
x ([ ; ] ; det@[ ; mp ; mp~ ; mp~@[) m
```

|   |   |   |   |   |   |      |    |    |    |    |   |    |    |    |    |
|---|---|---|---|---|---|------|----|----|----|----|---|----|----|----|----|
| 1 | 2 | 3 | 1 | 6 | 4 | _112 | 27 | 26 | 28 | 25 | 6 | 42 | 49 | 36 | 36 |
|   |   |   | 4 | 1 | 0 |      |    |    |    |    |   |    | 8  | 25 | 16 |
|   |   |   | 6 | 6 | 8 |      |    |    |    |    |   |    | 78 | 90 | 88 |

The monad  $u \cdot v$  is defined as illustrated below:

```
DET=: 2 : 'v./@,`({."1 u. . v. $:@minors)@.(0:<{:@$) @ ,. "2'
minors=: }. "1 @ (1&([\.))

-/ DET * m
_112

-/ DET * 1 16 64
49

-/ DET * i.3 0
1

+/ DET * m
320
```

The definition  $u@(v"(1+1v, \_))$  given above for the dyadic case may be re-stated

in words as follows:  $u$  is applied to the result of  $v$  on lists of "left argument cells" and the right argument *in toto*. The number of items in a list of left argument cells must agree with the number in the right argument. Thus, if  $v$  has ranks  $2\ 3$  and the shapes of  $x$  and  $y$  are  $2\ 3\ 4\ 5\ 6$  and  $4\ 7\ 8\ 9\ 10\ 11$ , then there are  $2\ 3$  lists of left argument cells (each shaped  $4\ 5\ 6$ ); and if the shape of a result cell is  $sr$ , the overall shape is  $2\ 3, sr$ .

## Even , Odd

$$u \ . \ . \ v \quad u \ . \ : \ v$$

```
u . . v ↔ (u + u&v) % 2:
u . : v ↔ (u - u&v) % 2:
```

In the most commonly used case,  $v$  is arithmetic negation, and  $f =: u \ . \ . \ v$  is therefore  $f =: (u + u&-) \% 2:$ ; that is, one-half the sum of  $u \ y$  and  $u \ -y$ . The resulting function is therefore *even* in the sense that  $f \ y \leftrightarrow f \ -y$  for any  $y$ ; its graph is reflected in the vertical axis. Similarly,  $u \ . \ : \ -$  is *odd* ( $f \ y \leftrightarrow -f \ -y$ ), and its graph is reflected in the origin. Less commonly,  $v$  is matrix transpose ( $| :$ ), and may be any monadic function.

```
y=: _2 _1 0 1 2
1 2 3 4 5 & p. y
```

Polynomial with odd and even terms

```
57 3 1 15 129
```

```
1 2 3 4 5 & p. . . - y
```

Even part of polynomial

```
93 9 1 9 93
```

```
1 0 3 0 5 & p. y
```

Polynomial with even terms only

```
93 9 1 9 93
```

```
E=: . . -
O=: . : -
d=: 5j2&" :@, .&.>
```

Even adverb  
Odd adverb  
Display as columns with two digits

```
d (5&o. ; ^O ; 6&o. ; ^E ; ^ ; (^E + ^O) ; 2&o. ; ^@j.E) y
```

|       |       |      |      |      |      |       |       |
|-------|-------|------|------|------|------|-------|-------|
| _3.63 | _3.63 | 3.76 | 3.76 | 0.14 | 0.14 | _0.42 | _0.42 |
| _1.18 | _1.18 | 1.54 | 1.54 | 0.37 | 0.37 | 0.54  | 0.54  |
| 0.00  | 0.00  | 1.00 | 1.00 | 1.00 | 1.00 | 1.00  | 1.00  |
| 1.18  | 1.18  | 1.54 | 1.54 | 2.72 | 2.72 | 0.54  | 0.54  |
| 3.63  | 3.63  | 3.76 | 3.76 | 7.39 | 7.39 | _0.42 | _0.42 |

```
m=: ?. 4 4 $ 9
(] ; (] . . |:) ; (] . : |:)) m
```

|         |       |     |       |      |
|---------|-------|-----|-------|------|
| 1 6 4 4 | 1 3.5 | 6 2 | 0 2.5 | _2 2 |
|---------|-------|-----|-------|------|

|                     |   |   |   |     |     |   |     |     |   |      |  |    |      |  |   |  |
|---------------------|---|---|---|-----|-----|---|-----|-----|---|------|--|----|------|--|---|--|
| 1                   | 0 | 6 | 6 | 3.5 |     | 0 | 4.5 |     | 3 | _2.5 |  | 0  | 1.5  |  | 3 |  |
| 8                   | 3 | 4 | 7 | 6   | 4.5 |   | 4   | 5.5 | 2 | _1.5 |  | 0  | 1.5  |  |   |  |
| 0                   | 0 | 4 | 6 | 2   |     | 3 | 5.5 |     | 6 | _2   |  | _3 | _1.5 |  | 0 |  |
| +-----+-----+-----+ |   |   |   |     |     |   |     |     |   |      |  |    |      |  |   |  |

## Explicit Definition

m : n \_ \_ \_

As defined and illustrated in [Section II H p70](#), the phrase `s=: 0 : 0` may be used to define `s` as a script, and the explicit definition conjunction can be further used to produce a dyadic-only verb (`4 : s`), verb (`3 : s`), conjunction (`2 : s`), adverb (`1 : s`), or noun (`0 : s`). The left argument `13` may be used instead of `3` to produce equivalent results in tacit form if possible. The right argument `0` may be used in each case to make the corresponding definitions directly from the keyboard: `k : 0` is equivalent to `k : (0 : 0)`. Furthermore, the boxed representation `b=: < ; . _2 s` or the table representation `t=: > b` (or `t=: [ ; . _2 s`) may be used in lieu of the script `s` in every case. Thus:

```
f=: 3 : 0
a=: 2+b=. y. ^ 2
a+a*b
:
x.*x.+y.
)
```

```
a=: b=: 19
f 3
110
```

```
a,b
11 19
```

Only the globally assigned name is changed.

As illustrated by the foregoing:

1. The definitions of the monadic and dyadic cases produced by `3 : 0` are separated by a colon on a line by itself; if none occurs, the domain of the dyadic case is empty.
2. The explicit result is the result of the last non-test block sentence executed; that result must be a noun in the `3 :` and `4 :` cases. See *Control Structures* for the definition of a test block.
3. A name assigned by the copula `=.` is made *local*; values assigned to it have no effect on the use of the same name *without* the entity defined or *within* other entities invoked by it. A name assigned by `=:` is global.



4. A locative cannot be localized.
5. The names `x.` and `y.` denote the left and right arguments. In defining a conjunction it may be necessary to refer to *its* left and right arguments (using `u.` and `v.`) as well as to the arguments of the resulting function (`x.` and `y.`). The use of `m.` instead of `u.` restricts the corresponding argument to being a noun, as does the use of `n.` instead of `v.`. For example:

```

conj=: 2 : '(u. y.)+ (v. y.)'
mc=: 2 : 0
(u.y.)+(v.y.)
)

dc=: 2 : 0                                Dyadic case
:
(u.y.)+(v.x.)
)

(!conj% 2 4 5);(!mc% 2 4 5);(1 2 3 !dc% 2 4 5)
+-----+-----+-----+
| 2.5 24.25 120.2|2.5 24.25 120.2|3 24.5 120.333|
+-----+-----+-----+

```

**Control Structures.** The sequence of execution of an explicit definition may be determined by *control words* such as `if.` `do.` `else.` `end.` and `while.`. For example, a function to find the root of a function `f` from a two-element list that brackets the root may be written and executed as follows:

```

root=: 3 : 0
m=.%#while.~/y.do.if.~/*f b=.(m,{.)y.do.y=.b
else.y=.(m,{:)y.end.end.m y.
)

f=: 4:-%:
b=: 1 32
root b
16

```

Such a definition may also be written on successive lines by breaking it before or after any control word, and the foregoing definition may be made more readable as follows:

```

root=: 3 : 0
m=. +/ % #

```

```

while. ~:/y.
  do. if. ~:/*f b=. (m,{.) y.
    do. y.=. b
    else. y.=. (m,{:) y.
  end.
end. m y.
)

```

As illustrated by the foregoing, the word `if.` and a matching `end.` mark the beginning and end of a *control structure*, as do `while.` and a matching `end.`; such structures may be *nested* as is the `if.` structure within the `while.` structure.

The control words `for.`, `if.`, `select.`, `try.`, `while.`, and `whilst.` mark the beginnings of control structures that are each terminated by a matching `end.`, and therefore provide a form of punctuation. In the foregoing example, `do.` and `else.` break the `if.` structure into three simple blocks, each comprising a sentence, whereas the `do.` in the `while.` structure breaks it into two blocks, the first being a simple sentence, and the second being itself an `if.` control structure.

In general, a *block* comprises zero or more simple sentences and control structures. The role of blocks is summarized as follows:

```

for.      T do. B end.
for_xyz.  T do. B end.
if. T do. B end.
if. T do. B else. B1 end.
if. T do. B elseif. T1 do. B1 elseif. T2 do. B2 end.
select. T case. T0 do. B0 fcase. T1 do. B1 case. T2 do. B2 end.
try. B catch. B1 end.
while. T do. B end.
whilst. T do. B end.

```

Like `while.`, but Skips Test first time.

Words beginning with **B** or **T** denote blocks. The last sentence executed in a **T** block is tested for a non-zero value in its leading atom, and the result of the test determines the block to be executed next. (An empty **T** block result or an omitted **T** block tests true.) If an error occurs in a `try.` block, execution continues in the matching `catch.` block. The final result is the result of the last sentence executed that was not in a **T** block, and if there is no such last executed sentence, the final result is `i.0 0.`

The behaviour of the remaining control words may be summarized as follows:

|             |                                                         |
|-------------|---------------------------------------------------------|
| assert. T   | signal assertion failure if T is not all 1s             |
| break.      | Go to end of for., while., or whilst. control structure |
| continue.   | Go to top of for., while., or whilst. control structure |
| goto_name.  | Go to label of same name                                |
| label_name. | Target of goto_name.                                    |
| return.     | Exit the function                                       |

Additional explanations and examples can be found in the [Control Structures p200](#) section.

The following example uses control words as well as u. and n. in modelling the Level conjunction L: :

```

Level=: 2 : 0
m=. 0{ 3&$&.|. n.
ly=. L. y. if. 0>m do. m=.0>.m+ly end.
if. m>:ly do. u. y. else. u. Level m&.> y. end.
:
'l r'=. 1 2{ 3&$&.|. n.
lx=. L. x. if. 0>l do. l=.0>. l + lx end.
ly=. L. y. if. 0>r do. r=.0>. r + ly end.
b=. (l,r)>:lx,ly
if. b=: 0 0 do. x. u. Level(l,r)&.>y.
elseif. b=: 0 1 do. x. u. Level(l,r)&.>y.
elseif. b=: 1 0 do. (<x.) u. Level(l,r)&.>y.
elseif. 1 do. x. u. y.
end.
)

```

```

] a=: (i.2 3);(<<2 3 4) ; 3 +-----+-----+--+ |0 1 2|+-----
+|3| |3 4 5|+-----+|| | | ||2 3 4||| | | |+-----+|| | | |+-----
-+| | +-----+-----+--+ +: level 0 a +-----+-----+--+ |0 2
4|+-----+|6| |6 8 10|+-----+|| | | ||4 6 8||| | | |+-----+||
| | |+-----+| | +-----+-----+--+ +: 1: 0 a +-----+-----
--+ |0 2 4|+-----+|6| |6 8 10|+-----+|| | | ||4 6 8||| | | |+-----
----+|| | | |+-----+| | +-----+-----+--+

```

## Monad / Dyad

u : v \_ \_ \_

The first argument specifies the monadic case and the second argument the dyadic case.

For example:

```
y=: 10 64 100
^. y
2.30259 4.15888 4.60517
```

Natural logarithm

```
10&^. y
1 1.80618 2
```

Base ten logarithm

```
log=: 10&^. : ^.
log y
1 1.80618 2
```

```
8 log y
1.10731 2 2.21462
```

```
(^1) log y
2.30259 4.15888 4.60517
```

```
LOG=: 10&$: : ^.
LOG y
1 1.80618 2
```

Use of self-reference

```
f=: %: : ($:@-)
(f y),: 100 f y
3.16228 8 10
9.48683 6 0
```

```
ABS=: | : [:
ABS _4
4
```

```
3 ABS _4
|valence error: ABS
```

| 3 ABS \_4

The domain of the dyad ABS is empty because the domain of [ : is empty.

**Obverse**

$$u :. v \quad \text{mu lu ru}$$

The result of  $u :. v$  is the verb  $u$ , but with an assigned obverse  $v$  (used as the "inverse" under the conjunctions  $\&.$  and  $\wedge:$ ).

For example:

```

y=: _4 0 4 3j4
rp=: <@(%: , -@%:)"0      Root pairs
rp y
+-----+-----+-----+-----+
|0j2 0j_2|0 0|2 _2|2j1 _2j_1|
+-----+-----+-----+-----+

I=: ^: _1
rp I      No assigned obverse
rp^:_1

rp I rp y
|domain error
|      rp I rp y

inv=: *:@{. @, @>
inv rp y
_4 0 4 3j4

RP=: rp :. inv      Assigned obverse in RP
RP I RP y
_4 0 4 3j4

rc=: <@(: +)@(: -)@%:"0      Root companions
rc y
+-----+-----+-----+-----+
| 0j2 0j_2|0 0|2 _2| 2j1 _2j_1|
|0j_2 0j2|0 0|2 _2|2j_1 _2j1|
+-----+-----+-----+-----+

RC=: rc :. inv
RC I RC y
_4 0 4 3j4

```

## Adverse

$$u \vdash v \quad \_ \_ \_$$

The result of  $u \vdash v$  is that of  $u$ , provided that  $u$  completes without error; otherwise the result is the result of  $v$ .

For example:

```
p=: 3 1 0 2
x=: 'ABCD'
p{x
DBAC
```

A permutation vector

```
]i=: A. p
20
```

Atomic index in ordered list of permutations

```
i A. x
DBAC
```

Permutation by atomic representation

```
q=: 3 1 1 0
q{x
DBBA
```

Not a permutation

```
A. q
|index error
|      A.q
```

```
A=: A. :: (!@#)
A p
20
```

Give index outside range in case of error

```
A q
24
```

```
24 A. x
|index error
| 24 A.x
```

## Ravel

, — — —

## Append

`,y` gives a list of the atoms of `y` in "normal" order: the result is ordered by items, by items within items, etc. The result shape is `1$*/$ y`. Thus:

```
y=: 2 4 $ 'abcdefgh'

y
abcd
efgh

,y
abcdefgh
```

`x,y` appends items of `y` to items of `x` after:

1. Reshaping an atomic argument to the shape of the items of the other,
2. Bringing the arguments to a common rank (of at least 1) by repeatedly *itemizing* (`,:`) any of lower rank, and
3. Bringing them to a common shape by padding with fill elements in the manner described in [Section II B p64](#).

The fit conjunction (`,!f`) provides fill specified by the items of `f`.

```
]a=: i. 2 3 3
0 1 2
3 4 5
6 7 8

9 10 11
12 13 14
15 16 17

,a
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

,"2 a
0 1 2 3 4 5 6 7 8
9 10 11 12 13 14 15 16 17
```

The following examples illustrate the dyadic case:



( 'abc' , 'de' ); ( 'abc' , "0/'de' ); ( 5 6 7 , i.2 3 ); ( 7 , i.2 3 )

|       |    |       |       |
|-------|----|-------|-------|
| abcde | ad | 5 6 7 | 7 7 7 |
|       | ae | 0 1 2 | 0 1 2 |
|       |    | 3 4 5 | 3 4 5 |
|       | bd |       |       |
|       | be |       |       |
|       | cd |       |       |
|       | ce |       |       |

**Ravel Items**

, . \_ \_ \_

**Stitch**

If  $y$  is an atom, then  $,.y$  is  $1\ 1\$y$ ; otherwise,  $,.y$  is  $,\_1\ y$ , the table formed by ravelling each item of  $y$ .

$x,.y$  is equivalent to  $x,\_1\ y$ . In other words, items of  $x$  are stitched to corresponding items of  $y$ .

The fit conjunction  $(,.\!.f)$  provides fill specified by the items of  $f$ .

For example:

```
a=: i. 2 3 2
($,.3) ; (,.2 3 5 7 11) ; ($,.<'abcd') ; a ; (,.a)
```

|   |   |    |   |   |    |    |   |   |   |   |    |    |
|---|---|----|---|---|----|----|---|---|---|---|----|----|
| 1 | 1 | 2  | 1 | 1 | 0  | 1  | 0 | 1 | 2 | 3 | 4  | 5  |
|   |   | 3  |   |   | 2  | 3  | 6 | 7 | 8 | 9 | 10 | 11 |
|   |   | 5  |   |   | 4  | 5  |   |   |   |   |    |    |
|   |   | 7  |   |   |    |    |   |   |   |   |    |    |
|   |   | 11 |   |   | 6  | 7  |   |   |   |   |    |    |
|   |   |    |   |   | 8  | 9  |   |   |   |   |    |    |
|   |   |    |   |   | 10 | 11 |   |   |   |   |    |    |

The following examples illustrate the dyadic case:

```
b=:3 4$'abcdefghijkl' [ c=:3 4$'ABCDEFGHIJKL'
b ; c ; (b,.c) ; (b,c) ; a ; (a ,. |."1 a) ; (,./a) ; (,./a)
```

|      |      |          |      |    |    |    |    |    |    |   |   |    |    |
|------|------|----------|------|----|----|----|----|----|----|---|---|----|----|
| abcd | ABCD | abcdABCD | abcd | 0  | 1  | 0  | 1  | 0  | 1  | 0 | 1 | 6  | 7  |
| efgh | EFGH | efghEFGH | efgh | 2  | 3  | 2  | 3  | 2  | 3  | 2 | 3 | 8  | 9  |
| ijkl | IJKL | ijklIJKL | ijkl | 4  | 5  | 4  | 5  | 4  | 5  | 4 | 5 | 10 | 11 |
|      |      |          | ABCD |    |    | 1  | 0  | 6  | 7  |   |   |    |    |
|      |      |          | EFGH | 6  | 7  | 3  | 2  | 8  | 9  |   |   |    |    |
|      |      |          | IJKL | 8  | 9  | 5  | 4  | 10 | 11 |   |   |    |    |
|      |      |          |      | 10 | 11 |    |    |    |    |   |   |    |    |
|      |      |          |      |    |    | 6  | 7  |    |    |   |   |    |    |
|      |      |          |      |    |    | 8  | 9  |    |    |   |   |    |    |
|      |      |          |      |    |    | 10 | 11 |    |    |   |   |    |    |
|      |      |          |      |    |    | 7  | 6  |    |    |   |   |    |    |

[illegible]

**Itemize**

, : — — —

**Laminate**

,:y adds a leading unit axis to y, giving a result of shape 1,\$y. Thus:

```
$ ,: 2 3 4
1 3
```

An atomic argument in x,:y is first reshaped to the shape of the other (or to a list if the other argument is also atomic); the results are then itemized and catenated, as in (,:x),(,:y).

The fit conjunction (:!.) provides fill specified by the items of f.

```
s=: 3 [ v=: 2 3 4 [ m=: i. 3 3
(, :s); ($, : s); (:v); ($, :v); ($, :m); ($, :^:4 v)
+---+-----+---+-----+-----+
|3|1|2 3 4|1 3|1 3 3|1 1 1 1 3|
+---+-----+---+-----+-----+
```

The following examples compare the dyadic cases of Append and Laminate:

```
a=: 'abcd' [ A=: 'ABCD' [ b=: 'abcdef'
(a,A) ; (a,:A) ; (a,:b) ; (m,m) ; (m ,: m)
+-----+-----+-----+-----+
abcdABCD	abcd	abcd	0 1 2	0 1 2
	ABCD	abcdef	3 4 5	3 4 5
			6 7 8	6 7 8
			0 1 2	
			3 4 5	0 1 2
			6 7 8	3 4 5
				6 7 8
+-----+-----+-----+-----+
```

```
t=: i. 3 2 2
t ; (:,/t) ; (:./t) ; (:,/t)
+-----+-----+-----+
0 1	0 1	0 1 4 5 8 9	0 1
2 3	2 3	2 3 6 7 10 11	2 3
	4 5		
4 5	6 7		0 0
+-----+-----+-----+
```

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 6  | 7  | 8  | 9  | 0  | 0  |
| 8  | 9  | 10 | 11 |    |    |
| 10 | 11 |    |    | 4  | 5  |
|    |    |    |    | 6  | 7  |
|    |    |    |    | 8  | 9  |
|    |    |    |    | 10 | 11 |

Raze

i    \_   \_   \_

Link

i y assembles along a leading axis the opened elements of the ravel of y .

x i y is ( < x ) , y if y is boxed, and ( < x ) , < y if y is open.

```
]bv=: 1 2 3;4 5 6;7 8 9
+-----+-----+-----+
|1 2 3|4 5 6|7 8 9|
+-----+-----+-----+

;bv
1 2 3 4 5 6 7 8 9

]m=: >bv
1 2 3
4 5 6
7 8 9

i/ m
+-----+-----+-----+
|1 2 3|4 5 6|7 8 9|
+-----+-----+-----+

( ;/1 2 3 4 5) ,&< ( ;/i. 3 4)
+-----+-----+-----+
+--+--+--+--+	+-----+-----+-----+											
	1	2	3	4	5			0 1 2 3	4 5 6 7	8 9 10 11		
+--+--+--+--+	+-----+-----+-----+											
+-----+-----+-----+

]txt=: '3 %: 4 ^. 5'
3 %: 4 ^. 5

]s=: i: txt
+--+--+--+--+
|3|%:|4|^.|5|
+--+--+--+--+

;s
3%:4^.5
```

Word formation

```
(boxifopen=: <^:(< -: {:@;~)) 3 4
+----+
| 3 4 |
+----+
```

```
(<3 4) = boxifopen <3 4
1
```

**Cut**

$$m; .n \quad u; .n \quad \_1/2 \quad \_$$
**Cut**

$u; .0 \ y$  applies  $u$  to  $y$  after reversing  $y$  along each axis; it is equivalent to  $(0 \ \_1 \ * / \$y) \ u; .0 \ y$ .

The *fret*  $0\{y$  (the leading item of  $y$ ) marks the start of an interval of items of  $y$ ; the phrase  $u; .1 \ y$  applies  $u$  to each such interval. The phrase  $u; .\_1 \ y$  differs only in that frets are excluded from the result. In  $u; .2$  and  $u; .\_2$  the fret is the *last* item, and marks the *ends* of intervals.

The monads  $u; .3$  and  $u; .\_3$  apply  $u$  to tessellation by "maximal cubes", that is, they are defined by their dyadic cases using the left argument  $(\$ \$y) \$< . / \$y$ .

$m; .n \ y$  applies successive verbs from the gerund  $m$  to the cuts of  $y$ , extending  $m$  cyclically as required.

$x \ u; .0 \ y$  applies  $u$  to a rectangle or cuboid of  $y$  with one vertex at the point in  $y$  indexed by  $v = 0\{x$ , and with the opposite vertex determined as follows: the dimension is  $|1\{x$ , but the rectangle extends *back* from  $v$  along any axis  $j$  for which the index  $j\{v$  is negative. Finally, the order of the selected items is reversed along each axis  $k$  for which  $k\{1\{x$  is negative. If  $x$  is a vector, it is treated as the matrix  $0, :x$ .

The frets in the dyadic cases  $1$ ,  $\_1$ ,  $2$ , and  $\_2$  are determined by the  $1$ s in boolean vector  $x$ ; an empty vector  $x$  and non-zero  $\#y$  indicates the entire of  $y$ . In general, boolean vector  $>j\{x$  specifies how axis  $j$  is to be cut.

$u; .3$  and  $u; .\_3$  yield (possibly overlapping) tessellations.  $x \ u; .\_3 \ y$  applies  $u$  to each *complete* rectangle of size  $|1\{x$  beginning at integer multiples of (each item of) the movement vector  $0\{x$ , with an infinite size being replaced by the signed length of the corresponding axis. As in  $u; .0$ , reversal occurs along each axis for which the size  $1\{x$  is negative. The case of a list  $x$  is equivalent to  $1, :x$ , and therefore provides a complete tessellation of size



$x$ . The case  $u; .3$  differs in that shards of length less than  $|1\{x$  are included.

$x\ m; .n\ y$  applies successive verbs from the gerund  $m$  to the cuts of  $y$ , extending  $m$  cyclically as required.

The 0- and 3-cuts have a left rank of 2; the 1- and 2-cuts have a left rank of 1.

```

y=: 'worlds on worlds '
(< i.2 y) ; ($ i._2 y) ; (3 5$ i.10) ; (+/ i.1 (3 5$ i.10))
+-----+-----+-----+
+-----+---+-----+	6	0 1 2 3 4	5 7 9 11 13				
	worlds	on	worlds		2	5 6 7 8 9	0 1 2 3 4
+-----+---+-----+	6	0 1 2 3 4					
+-----+-----+-----+

1 0 1 0 0 < i.1 i.5 7
+-----+
|0 1 2 3 4 5 6 |
|7 8 9 10 11 12 13 |
+-----+
|14 15 16 17 18 19 20|
|21 22 23 24 25 26 27|
|28 29 30 31 32 33 34|
+-----+
('';1 0 0 0 1 0 1) < i.1 i.5 7
+-----+-----+-----+
0 1 2 3	4 5	6
7 8 9 10	11 12	13
14 15 16 17	18 19	20
21 22 23 24	25 26	27
28 29 30 31	32 33	34
+-----+-----+-----+		
(1 0 1 0 0;1 0 0 0 1 0 1) < i.1 i.5 7		
+-----+-----+-----+		
0 1 2 3	4 5	6
7 8 9 10	11 12	13
+-----+-----+-----+		
14 15 16 17	18 19	20
21 22 23 24	25 26	27

```

```
|28 29 30 31|32 33|34|
+-----+-----+-----+
```

```
x=:1 _2,:_2 3 [ z=: i. 5 5
x ; (x ]i.0 z) ; z
```

```
+-----+-----+-----+
1 _2	11 12 13	0 1 2 3 4
_2 3	6 7 8	5 6 7 8 9
		10 11 12 13 14
		15 16 17 18 19
		20 21 22 23 24
+-----+-----+-----+
```

```
(y=: a. {~ (a. i. 'a') + i. 4 4);(a=: 1 1 ,: 2 2)
```

```
+-----+-----+
abcd	1 1
efgh	2 2
ijkl	
mnop	
+-----+-----+
```

```
(<i.3 y) ; ((($y)$<./$y)<i.3 y) ; (a <i.3 y) ; <(a <i._3 y)
```

```
+-----+-----+-----+-----+
+-----+-----+-----+	+-----+-----+-----+	+-----+-----+-----+	+-----+-----+-----+																			
	abcd	bcd	cd	d			abcd	bcd	cd	d			ab	bc	cd	d			ab	bc	cd	
	efgh	fgh	gh	h			efgh	fgh	gh	h			ef	fg	gh	h			ef	fg	gh	
	ijkl	jkl	kl	l			ijkl	jkl	kl	l		+-----+-----+	+-----+-----+									
	mnop	nop	op	p			mnop	nop	op	p			ef	fg	gh	h			ef	fg	gh	
+-----+-----+-----+	+-----+-----+-----+		ij	jk	kl	l			ij	jk	kl											
	efgh	fgh	gh	h			efgh	fgh	gh	h		+-----+-----+	+-----+-----+									
	ijkl	jkl	kl	l			ijkl	jkl	kl	l			ij	jk	kl	l			ij	jk	kl	
	mnop	nop	op	p			mnop	nop	op	p			mn	no	op	p			mn	no	op	
+-----+-----+-----+	+-----+-----+-----+	+-----+-----+-----+	+-----+-----+-----+																			
	ijkl	jkl	kl	l			ijkl	jkl	kl	l			mn	no	op	p						
	mnop	nop	op	p			mnop	nop	op	p		+-----+-----+-----+										
+-----+-----+-----+	+-----+-----+-----+																					
	mnop	nop	op	p			mnop	nop	op	p												
+-----+-----+-----+	+-----+-----+-----+																					
+-----+-----+-----+-----+
```

**Word Formation****i : 1**

i:y is the list of boxed words in the list y according to the rhematic rules of [Part I p61](#). The function also applies reasonably well to ordinary text.

```
s=: '*: @ -: @ i. 2 3'
do=: ".
do s
0 0.25      1
2.25      4 6.25
```

```
i: s
+---+---+---+---+---+
|*:@|-:@|i.|2 3|
+---+---+---+---+---+
```

```
i i: s
*:@-:@i.2 3
```

```
p=: 'When eras die, their legacies/'
q=: 'are left to strange police'
r=: 'Professors in New England guard'
s=: 'the glory that was Greece'
```

```
i: p
+-----+-----+-----+-----+
|When|eras|die|,|their|legacies|/|
+-----+-----+-----+-----+
```

```
> i: p,q
When
eras
die
,
their
legacies
/
```

are  
left  
to  
strange  
police

|. & . i : p  
/ legacies their , die eras When

**Tally**

# \_ 1 \_

**Copy**

#y is the number of items in y. Thus:

```
(#'' ); (#'a' ); (#'octothorpe')
+---+---+
|0|1|10|
+---+---+
```

```
(#3); (#,3); (# 3 4)
```

```
+---+---+
|1|1|2|
+---+---+
```

```
(#i.4 5 6); (#$i.4 5 6)
```

```
+---+---+
|4|3|
+---+---+
```

If the arguments have an equal number of items, then  $x\#y$  copies  $+/x$  items from  $y$ , with  $i\{x$  repetitions of item  $i\{y$ . Otherwise, if one is an atom it is repeated to make the item count of the arguments equal.

The complex left argument  $a\ j.\ b$  copies  $a$  items followed by  $b$  fills. The fit conjunction provides specified fills, as in  $\#!.f$ .

Copy is illustrated by the following examples:

```
0 1 2 3 4 5 # 0 1 2 3 4 5
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
```

```
t=: 3 4 $'abcdefghijkl' [ n=: i. 3 4
t ; n ; (3 0 1 # t) ; (3 0 1 # n) ; (3 1 4 2 # "1 t)
+---+---+---+---+---+---+---+---+
abcd	0 1 2 3	abcd	0 1 2 3	aaabccccdd
efgh	4 5 6 7	abcd	0 1 2 3	eeefgggghh
ijkl	8 9 10 11	abcd	0 1 2 3	iiijkkkkl1
		ijkl	8 9 10 11	
+---+---+---+---+---+---+---+---+
```

```
k=: 2j1 0 1j2
(k # t);(k # n);(k #!.'*' t);(k #!.4 n)
+---+---+---+---+---+---+---+---+
abcd	0 1 2 3	abcd	0 1 2 3
abcd	0 1 2 3	abcd	0 1 2 3
	0 0 0 0	****	4 4 4 4
ijkl	8 9 10 11	ijkl	8 9 10 11
```

|   |   |   |   |   |   |   |   |      |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|------|---|---|---|---|---|---|
|   |   |   | 0 | 0 | 0 | 0 |   | **** |   | 4 | 4 | 4 | 4 |   |
|   |   |   | 0 | 0 | 0 | 0 |   | **** |   | 4 | 4 | 4 | 4 |   |
| + | - | - | - | + | - | - | - | -    | + | - | - | - | - | + |

**Base Two**

#. 1 1 1

**Base**

#.y is the base-2 value of y, that is, 2#.y. For example:

```
    #. 1 0 1 0
```

```
10
```

```
    #. 2 3$ 0 0 1,1 0 1
```

```
1 5
```

x#.y is a weighted sum of the items of y; that is,  $\sum w_i y_i$ , where w is the product scan  $\text{*/}\backslash\text{.}\}$ .x,1. An atomic argument is reshaped to the shape of the other argument.

```
    ]a=: i. 3 4
```

```
0 1 2 3
```

```
4 5 6 7
```

```
8 9 10 11
```

```
    10 #.a
```

```
123 4567 9011
```

```
    8 #. a
```

```
83 2423 4763
```

```
    ]time=: 0 1 3,1 1 3,:2 4 6
```

```
0 1 3
```

```
1 1 3
```

```
2 4 6
```

```
    x=: 24 60 60
```

```
    x #. time
```

```
63 3663 7446
```

```
    x,1
```

```
24 60 60 1
```

```
    ]w=: */\.\}. x,1
```

```
3600 60 1
```

```
    w *"1 time
```

```
0 60 3
```

```
3600 60 3
7200 240 6
```

```
    +/"1 w *"1 time
63 3663 7446
```

```
    w +/@"* "1 time
63 3663 7446
```

```
    c=: 3 1 4 2 [ y=: 0 1 2 3 4 5
    c p. y          Polynomial with coefficients c
3 10 37 96 199 358
```

```
    y #."0 1 |.c
3 10 37 96 199 358
```



**Antibase Two**

#: \_ 1 0

**Antibase**

#:  $y$  is the binary representation of  $y$ , and is equivalent to  $(m\#2)\#y$ , where  $m$  is the maximum of the number of digits needed to represent the atoms of  $y$  in base 2. For example:

```
i. 8
0 1 2 3 4 5 6 7
```

```
#: i. 8
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
```

In simple cases  $r\&\#:$  is inverse to  $r\&\#.$ . Thus:

```
r=: 24 60 60
r #: r #. 2 3 4
2 3 4
```

But if  $r\&\#.y$  exceeds  $(*/r)-1$  (the largest integer representable in the radix  $r$ ), then the result of  $r\#:$  is reduced modulo  $*/r$ . For example:

```
r #: r #. 29 3 4
5 3 4
```

A representation in an arbitrary base that is analogous to the base-2 representation provided by the monadic use of  $\#:$  may be provided as illustrated below:

```
ndr=: 1: + <.@^.
```

Number of digits required

```
10 ndr y=: 9 10 11 100 99 100
1 2 2 3 2 3
```

```
(y#::~~10 #~ >./10 ndr y);(y#::~~8 #~ >./8 ndr y)
```

```
+-----+-----+
0 0 9	0 1 1
0 1 0	0 1 2
0 1 1	0 1 3
1 0 0	1 4 4
0 9 9	1 4 3
1 0 0	1 4 4
```

+-----+-----+

(10&#x207f;^:\_1 ; 8&#x207f;^:\_1) y

+-----+-----+

|0 0 9|0 1 1|

|0 1 0|0 1 2|

|0 1 1|0 1 3|

|1 0 0|1 4 4|

|0 9 9|1 4 3|

|1 0 0|1 4 4|

+-----+-----+

## Factorial                      !   0   0   0                      Out Of (Combinations)

For a non-negative integer argument  $y$ , the definition is  $y! = \Gamma(1+y)$ . In general,  $y!$  is  $\Gamma(1+y)$  (the gamma function). Thus:

```
(*/1 2 3 4 5) , (!5)
120 120

]x=: 2 %~ 3 -- i. 2 4
_1.5 _1 _0.5 0
0.5 1 1.5 2

!x
_3.54491 _ 1.77245 1
0.886227 1 1.32934 2

]fi=:!^:_1(24 25 2.1 9876)
4 4.02705 2.05229 7.33019

! fi
24 25 2.1 9876
```

For non-negative arguments  $x!y$  is the number of ways that  $x$  things can be chosen out of  $y$ . More generally,  $x!y$  is  $(y!)/(x!(y-x)!)$ . Thus:

```
3!5
10
(!5)%(!3)*(!5-3)
10
1j2 ! 3.5
8.64269j16.9189

]y=:2&!^:_1 (45 4.1 30 123)
10 3.40689 8.26209 16.1924
2 ! y
45 4.1 30 123

]x=:!&10^:_1 (2.5 45)
0.3433618 2
x ! 10
2.5 45
```

The first table below illustrates the relation between the dyad  $!$  and the table of binomial coefficients; the last two illustrate its relation to the *figurate numbers*:

```
h=: 0,i=: i.5 [ j=: -1+i.5 [ k=: 5#1
tables=: ( ,.h);(i,i!/i);(j,i!/j);(k,i(+/\^:)k)
format=: ({. ,:&< }. )@":&.>
format tables
```

|   |   |   |   |   |   |   |   |   |    |    |   |   |   |    |    |   |
|---|---|---|---|---|---|---|---|---|----|----|---|---|---|----|----|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 1 | 1 | 2 | 3  | 4  | 5 | 1 | 1 | 1  | 1  | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1  | 1  | 1 | 1 | 1 | 1  | 1  | 1 |
| 1 | 0 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4  | 5  | 1 | 2 | 3 | 4  | 5  |   |
| 2 | 0 | 0 | 1 | 3 | 6 | 1 | 3 | 6 | 10 | 15 | 1 | 3 | 6 | 10 | 15 |   |

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |    |    |     |     |     |   |   |   |   |   |    |    |    |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|----|-----|-----|-----|---|---|---|---|---|----|----|----|---|---|
|   |   | 3 |   |   |   | 0 | 0 | 0 | 1 | 4 |   |   |   | -1 | -4 | -10 | -20 | -35 |   |   |   | 1 | 4 | 10 | 20 | 35 |   |   |
|   |   | 4 |   |   |   | 0 | 0 | 0 | 0 | 1 |   |   |   | 1  | 5  | 15  | 35  | 70  |   |   |   | 1 | 5 | 15 | 35 | 70 |   |   |
|   |   | + | - | + |   | + | - | - | - | - | + |   | + | -  | -  | -   | -   | -   | + |   | + | - | - | -  | -  | -  | + |   |
| + | - | - | + | - | - | - | - | - | - | - | + | + | - | -  | -  | -   | -   | -   | - | + | + | - | - | -  | -  | -  | + | + |

Figurate numbers of order zero are all ones; those of higher orders result from successive applications of subtotals (that is, sums over prefixes, or  $+/ \backslash$ ). Those of order two are the *triangular numbers*, resulting from subtotals over the integers beginning with one.

## Fit (Customize)

! .

This conjunction modifies certain verbs in ways prescribed in their definitions. For example,  $=!.\tau$  is the relation of equality using tolerance  $\tau$ , and  $^!.\tau$  is the *factorial function* so defined that  $x ^!.\tau n$  is  $x! / (x + \tau)^n$ . Consequently,  $^!._1$  is the *falling factorial function*.

Fit applies to the following verbs (to produce *variants*). The monadic case is shown before a bullet, and the dyadic case after it:

|                              |                                               |
|------------------------------|-----------------------------------------------|
| $< <: > >: +. *. -. -:   E.$ | • Tolerance                                   |
| $i. i:$                      |                                               |
| $<. >. * ~.$                 | Tolerance •                                   |
| $\#:$                        | • Tolerance                                   |
| $= ~: \#: e.$                | Tolerance • Tolerance                         |
| $^ p.$                       | • Stope function and polynomial based thereon |
| $\$  . , ,. ,: \# \{.$       | • Fill                                        |
| $" :$                        | Print precision •                             |

## Foreign

! :

This conjunction is used to communicate with the host system as well as with the keyboard (as an input file) and with the screen (as an output file). It is also used to provide a variety of extra-lingual facilities, such as setting the form of function display, determining the class of a name (noun, verb, adverb, or conjunction), and listing all existing names in specified classes.

```
(mean=: +/ % #) a=: 2 3 5 7 11 13
6.83333
```

```
mean
+ / % #
```

```
9!:3 (4)
mean
+- / --- +
--+- %
+- #
```

Tree display of verb

```
9!:3 (2 4 5)
mean
```

Boxed display

```
+-----+---+
+-+--+	%	#			
	+				
+-+--+					
+-----+---+
```

```
+- / --- +
--+- %
+- #
+ / % #
```

Tree display

Linear display

```
4!:0 'a'; 'mean'
0 3
```

Classes of names (noun 0, verb 3)

```
4!:1 (3)
+-----+
|mean|
+-----+
```

List of names in class 3

[Appendix A p214](#) shows all uses of the foreign conjunction.

**Insert**

m/ u/ \_ \_ \_

**Table**

$u/y$  applies the dyad  $u$  between the items of  $y$ . Thus:

```
m=: i. 3 2
m; (+/m); (+/"1 m); (+/2 3 4)
+---+---+-----+--+
0 1	6 9	1 5 9	9
2 3			
4 5			
+---+---+-----+--+
```

$m/y$  inserts successive verbs from the gerund  $m$  between items of  $y$ , extending  $m$  cyclically as required. Thus,  $+`*/i.6$  is  $0+1*2+3*4+5$ .

If  $x$  and  $y$  are numeric lists, then  $x */ y$  is their multiplication table. Thus:

```
1 2 3 */ 4 5 6 7
4 5 6 7
8 10 12 14
12 15 18 21
```

In general, each cell of  $x$  is applied to the entire of  $y$ . Thus  $x u/ y$  is equivalent to  $x u" (lu,_) y$  where  $lu$  is the left rank of  $u$ .

The case  $*/$  is called *outer product* in tensor analysis.

If  $y$  has no items (that is,  $0=\#y$ ), the result of  $u/y$  is the *neutral* or *identity element* of the function  $u$ . A neutral of a function  $u$  is a value  $e$  such that  $x u e \leftrightarrow x$  or  $e u x \leftrightarrow x$ , for every  $x$  in the domain (or some significant sub-domain such as boolean) of  $u$ . This definition of insertion over an argument having *zero* items extends partitioning identities of the form  $u/y \leftrightarrow (u/k\{.y\}) u (u/k\{.y\})$  to the cases  $k \in . 0, \#y$ .

The *identity function* of  $u$  is a function  $ifu$  such that  $ifu y \leftrightarrow u/y$  if  $0=\#y$ . The identity functions used are:

**Identity function****For** $\$&0@ \}. @\$$  $< > + - +. \sim: | (2\ 4\ 5\ 6\ b.)$  $\$&1@ \}. @\$$  $= <: >: * \% *. \%: ^ ! (1\ 9\ 11\ 13\ b.)$  $\$&_@ \}. @\$$  $<.$  $\$&__@ \}. @\$$  $>.$



|                         |            |
|-------------------------|------------|
| i.@(0&, )@(2&}. )@\$    | ,          |
| i.@(1&{. )@}.@\$        | C. {       |
| =@i.@(1&{. )@}.@\$      | % . +/ . * |
| ifu@#                   | u/         |
| \$&(v^:_1 ifu\$0)@}.@\$ | u&.v       |

**Oblique**

$$m/. \quad u/. \quad \_ \_ \_$$
**Key**

$u/.y$  applies  $u$  to each of the oblique lines of a table  $y$ . For example:

```
i.3 4
0 1 2 3
4 5 6 7
8 9 10 11
```

```
</. i.3 4
+---+---+---+---+---+
|0|1 4|2 5 8|3 6 9|7 10|11|
+---+---+---+---+---+
```

In general,  $u/.y$  is the result of applying  $u$  to the oblique lines of  $\_2$ -cells of  $y$ . If the rank of  $y$  is less than two,  $y$  is treated as the table  $,.y$ .

$m/.y$  applies successive verbs from the gerund  $m$  to the oblique lines of  $\_2$ -cells of  $y$ , extending  $m$  cyclically as required. Thus:

```
<`(<@|.)/. i.3 4
+---+---+---+---+---+
|0|4 1|2 5 8|9 6 3|7 10|11|
+---+---+---+---+---+
```

$x u/.y \leftrightarrow (=x) u@# y$ , that is, items of  $x$  specify keys for corresponding items of  $y$  and  $u$  is applied to each collection of  $y$  having identical keys. For example:

```
1 2 3 1 3 2 1 </. 'abcdefg'
+---+---+---+
|adg|bf|ce|
+---+---+---+
```

$x m/.y$  applies successive verbs from the gerund  $m$  to the collections of  $y$ , extending  $m$  cyclically as required.

The application of a function to diagonals of a table is commonly useful, as in correlation, in convolution, and in products of polynomial coefficients (or, equivalently, products of numbers in a fixed base). For example:

```
t=: p */ q [ p=: 1 2 1 [ q=: 1 3 3 1

t ; (+//.t) ; 1 1 &(+//. @(*//)) ^: (i.6) 1
```

|   |   |   |   |   |   |    |    |   |   |   |   |    |    |   |   |   |
|---|---|---|---|---|---|----|----|---|---|---|---|----|----|---|---|---|
| 1 | 3 | 3 | 1 | 1 | 5 | 10 | 10 | 5 | 1 | 1 | 0 | 0  | 0  | 0 | 0 | 0 |
| 2 | 6 | 6 | 2 |   |   |    |    |   |   | 1 | 1 | 0  | 0  | 0 | 0 | 0 |
| 1 | 3 | 3 | 1 |   |   |    |    |   |   | 1 | 2 | 1  | 0  | 0 | 0 | 0 |
|   |   |   |   |   |   |    |    |   |   | 1 | 3 | 3  | 1  | 0 | 0 | 0 |
|   |   |   |   |   |   |    |    |   |   | 1 | 4 | 6  | 4  | 1 | 0 | 0 |
|   |   |   |   |   |   |    |    |   |   | 1 | 5 | 10 | 10 | 5 | 1 | 0 |

```
((10#.p)*10#.q), 10 #. +//. p */ q
161051 161051
```

Unlike polynomial coefficients, the diagonal sums of a multiplication table of digits should be "normalized" if any equal or exceed the radix.

**Grade Up**

/: \_ \_ \_

**Sort Up**

/: grades *any* argument, yielding a permutation vector; (/:y){y sorts y in ascending order. For example:

```
n=: 3 1 4 2 1 3 3
]g=: /: n
1 4 3 0 5 6 2

g { n
1 1 2 3 3 3 4
```

x/:y is (/:y){x; i.e., x is sorted to an order specified by y. In particular, y/:y (or /:~y) sorts y. For example:

```
y=: 'popfly'
y /: 3 1 4 1 5 9
ofpply

y /: y
floppy
```

Elements of /:y that select equal elements of y are in ascending order. If y is a table, /:y grades the base value of the rows, using a base larger than twice the magnitude of any of the elements. Higher ranks are treated as ,.y, (as if its items were each ravelled).

If y is literal, /:y grades according to the collating sequence determined by the alphabet a.; another collating sequence cs can be imposed by grading cs i. y. For example:

```
]n=: 3 1 4 1 6,2 7 1 8 3,:6 1 8 0 3
3 1 4 1 6
2 7 1 8 3
6 1 8 0 3
```

```
/: n
1 0 2
```

```
Aa=: ' ',. a. {~ 65 97 +/ i. 26
x=: words=: >: 'When eras die'
j=: <./Aa i."1 _ x
x ; (x/:x) ; (x/:j) ; Aa
```

```
+-----+-----+-----+-----+
|When|When|die | ABCDEFGHIJKLMNOPQRSTUVWXYZ|
|eras|die |eras| abcdefghijklmnopqrstuvwxyz|
```

```
|die |eras|When|  
+-----+-----+-----+-----+-----+-----+
```

The three types: numeric or empty, literal, and boxed, are so ordered; within them, a lower rank precedes a higher, and a smaller shape precedes a larger. Complex arguments are ordered by real part, then by imaginary. Boxed arrays are ordered according to the opened elements.

**Prefix**

$$m \backslash \quad u \backslash \quad \_ 0 \_$$
**Infix**

$u \backslash y$  has  $\#y$  items resulting from applying  $u$  to each of the prefixes  $k\{.y$ , for  $k$  from 1 to  $\#y$ .

$m \backslash y$  applies successive verbs from the gerund  $m$  to the prefixes of  $y$ , extending  $m$  cyclically as required.

If  $x > 0$ , the items of  $x \ u \backslash y$  result from applying  $u$  to each infix of length  $x$ . If  $x < 0$ ,  $u$  is applied to *non-overlapping* infixes of length  $|x|$ , including any final shard.

$x \ m \backslash y$  applies successive verbs from the gerund  $m$  to the infixes of  $y$ , extending  $m$  cyclically as required.

```

+/\a=: 1 2 4 8 16
1 3 7 15 31

```

Subtotals, or partial sums

```

*/\a
1 2 8 64 1024

```

Partial products

```

<\a
+---+---+---+---+
|1|1 2|1 2 4|1 2 4 8|1 2 4 8 16|
+---+---+---+---+

```

```

<\i.3 4
+---+---+---+
0 1 2 3	0 1 2 3	0 1 2 3
	4 5 6 7	4 5 6 7
		8 9 10 11
+---+---+---+

```

```

(+/\^:_1 +/\ a) ,: */\^:_1 a
1 2 4 8 16
1 2 2 2 2

```

The following examples illustrate the use of the dyad *infix*:

```

((2: -/\ ] ) ; (2: -~/\ ] )) a
+---+---+

```

Backward and forward differences

```
|_1 _2 _4 _8|1 2 4 8|
+-----+-----+
```

```
      ((3: <\ ]) ,&< (_3: <\ ])) 'abcdefgh'
+-----+-----+
+---+---+---+---+---+---+	+---+---+---+											
	abc	bcd	cde	def	efg	fgh			abc	def	gh	
+---+---+---+---+---+---+	+---+---+---+											
+-----+-----+
```

**Suffix**

$$m \backslash . \quad u \backslash . \quad \_ 0 \_$$
**Outfix**

$u \backslash . y$  has  $\#y$  items resulting from applying  $u$  to suffixes of  $y$ , beginning with one of length  $\#y$  (that is,  $y$  itself), and continuing through a suffix of length 1.

$m \backslash . y$  applies successive verbs from the gerund  $m$  to the suffixes of  $y$ , extending  $m$  cyclically as required.

If  $x > 0$  in  $x u \backslash . y$ , then  $u$  applies to outfixes of  $y$  obtained by suppressing successive infixes of length  $x$ . If  $x < 0$ , the outfixes result from suppressing non-overlapping infixes, the last of which may be a shard.

$x m \backslash . y$  applies successive verbs from the gerund  $m$  to the outfixes of  $y$ , extending  $m$  cyclically as required.

```
*/\ . y=: 1 2 3 4 5
120 120 60 20 5
```

```
<\ . y
+-----+-----+-----+-----+
| 1 2 3 4 5 | 2 3 4 5 | 3 4 5 | 4 5 | 5 |
+-----+-----+-----+-----+
```

```
3 <\ . 'abcdefgh'
+-----+-----+-----+-----+-----+
| defgh | aefgh | abfgh | abcgh | abcdh | abcde |
+-----+-----+-----+-----+-----+
```

```
_3 <\ . 'abcdefgh'
+-----+-----+-----+
| defgh | abcgh | abcdef |
+-----+-----+-----+
```

```
]m=: i.3 3
0 1 2
3 4 5
6 7 8
```

```
<"_2 (minors=: 1&( |: \ . )"2^:2) m
```



```
+----+----+----+
| 4 5|3 5|3 4|
| 7 8|6 8|6 7|
+----+----+----+
| 1 2|0 2|0 1|
| 7 8|6 8|6 7|
+----+----+----+
| 1 2|0 2|0 1|
| 4 5|3 5|3 4|
+----+----+----+
```

**Grade Down** $\backslash :$  — — —**Sort Down**

$\backslash :$  grades *any* argument, yielding a permutation vector;  $(\backslash :y)\{y$  sorts  $y$  in descending order. For example:

```
]g=: \:y=: 3 1 4 2 1 3 3
2 0 5 6 3 1 4

g{y
4 3 3 3 2 1 1
```

$x\backslash :y$  is  $(\backslash :y)\{x$ ; i.e.,  $x$  is sorted to an order specified by  $y$ . In particular,  $y\backslash :y$  (or  $\backslash :~y$ ) sorts  $y$ .

For example:

```
\:~"1 'dozen',: 'disk'
zoned
skid
```

Elements of  $\backslash :y$  that select equal elements of  $y$  are in ascending order. If  $y$  is a table,  $\backslash :y$  grades the base value of the rows, using a base larger than twice the magnitude of any of the elements. Higher ranks are treated as  $,.y$  (as if the items were each ravelled).

If  $y$  is literal,  $\backslash :y$  grades according to the collating sequence specified by the alphabet  $a.$ ; another collating sequence  $cs$  can be imposed by grading  $cs i.$   $y$ . For example:

```
]n=: 3 1 4 1 6,2 7 1 8 3,:6 1 8 0 3
3 1 4 1 6
2 7 1 8 3
6 1 8 0 3

\: n
2 0 1

\::~>:'when eras die, their legacies'
when
their
legacies
eras
die
,
```

See Grade Up ( $/:$ ) for the treatment of complex and boxed arguments.



Same

[ ] \_ \_ \_

Left, Right

The monads [ and ] are each *identity* functions; each yields its argument.

x [ y (*left bracket*) yields the left argument x, and x ] y (*right bracket*) yields the right argument y.

For example:

```
n=: i. 2 3
a=: 'abcde'

]n
0 1 2
3 4 5

[a
abcde

n[a
0 1 2
3 4 5

n]a
abcde

([ \ ; ] \ ; [ \ . ; ] \ .) 'ABCDEF'
```

|        |        |        |        |
|--------|--------|--------|--------|
| A      | A      | ABCDEF | ABCDEF |
| AB     | AB     | BCDEF  | BCDEF  |
| ABC    | ABC    | CDEF   | CDEF   |
| ABCD   | ABCD   | DEF    | DEF    |
| ABCDE  | ABCDE  | EF     | EF     |
| ABCDEF | ABCDEF | F      | F      |

**Cap**

[ : \_ \_ \_

**Cap**

[ : caps a left branch of a fork, as described in [Section II F p68](#). For example, the function `p=: [: +/ + * -` applies the monad `+/` to the result of the fork `+ *` - .

Caps make it possible to define a wider range of functions as unbroken trains. For example, the maximum divided by the product of the sum and difference would be defined by a single train, whereas (without the use of the cap) the definition of the maximum divided by the (monad) floor of the product of the sum and difference would require the use of trains interrupted by the monad. Thus:

```
f=: >. % + * -
g=: >. % <. @ (+ * -)

2.5 f 4
_0.410256

2.5 g 4
_0.4
```

The cap makes possible the use of an unbroken train as follows:

```
h=: >. % [: <. + * -

2.5 h 4
_0.4
```

Since the domain of the cap is empty, it can be used (with `:`) to define a function whose monadic or dyadic case invokes an error. For example:

```
abs=: | : [:
res=: [: : |

res _4 0 5
|valence error: res
|   res _4 0 5

abs _4 0 5
```

4 0 5

3 res \_4 0 5  
2 0 2

3 abs \_4 0 5  
|valence error: abs  
| 3 abs \_4 0 5

## Catalogue

{ 1 0 \_

## From

{<sub>y</sub> forms a catalogue from the atoms of its argument, its shape being the chain of the shapes of the opened items of <sub>y</sub>. The common shape of the boxed results is \$<sub>y</sub>. For example:

```

      { 'ht'; 'ao'; 'gtw'
+---+---+---+
|hag|hat|haw|
+---+---+---+
|hog|hot|how|
+---+---+---+
+---+---+---+
|tag|tat|taw|
+---+---+---+
|tog|tot|tow|
+---+---+---+

```

The Cartesian product is readily defined in terms of {, thus:

```

      CP=: { @ ( , & < )
      0 1 CP 2 3 4
+---+---+---+
|0 2|0 3|0 4|
+---+---+---+
|1 2|1 3|1 4|
+---+---+---+

```

If <sub>x</sub> is an integer in the range from -<sub>n</sub>=: #<sub>y</sub> to <sub>n</sub>-1, then <sub>x</sub>{<sub>y</sub> selects item <sub>n</sub>|<sub>x</sub> from <sub>y</sub>. Thus:

```

      2 0 _1 _3 { 'abcdefg'
cage

      t=: 3 4 $ 'abcdefghijkl'
      1 { t
efgh

```

More generally, ><sub>x</sub> may be a list whose successive elements are (possibly) boxed arrays that specify selection along successive axes of <sub>y</sub>.

Finally, if any <sub>r</sub>=: ><sub>j</sub>{><sub>x</sub> used in the selection is itself boxed, selection is made by the indices along that axis that *do not* occur in ><sub>r</sub>.

Note that the result in the very last dyadic example, that is, (<<<\_1){<sub>m</sub>, is all *except* the last item.

```

      t=: 3 4 $ 'abcdefghijkl'
      t; (1{t); (2 1{t); (1{"1 t); ((,1){"1 t); (2 1{"1 t)
+---+---+---+---+---+---+
abcd	efgh	ijkl	bfj	b	cb
efgh		efgh		f	gf
ijkl				j	kj

```

+-----+-----+-----+-----+-----+

t; (2 0{t); ((<2 0){t); ((2 0;1 3){t); ((<2 0;1 3){t)

+-----+-----+-----+-----+

|      |      |   |    |    |
|------|------|---|----|----|
| abcd | ijkl | i | ih | jl |
| efgh | abcd |   |    | bd |
| ijkl |      |   |    |    |

+-----+-----+-----+-----+

(\_1{m); (\_1{"2 m); (\_1{"1 m); (<<<\_1){m=:i.2 3 4

+-----+-----+-----+-----+

|             |             |          |           |
|-------------|-------------|----------|-----------|
| 12 13 14 15 | 8 9 10 11   | 3 7 11   | 0 1 2 3   |
| 16 17 18 19 | 20 21 22 23 | 15 19 23 | 4 5 6 7   |
| 20 21 22 23 |             |          | 8 9 10 11 |

+-----+-----+-----+-----+



**Head**

{ . \_ 1 \_

**Take**

{.y selects the leading item of y, or an item of fills if y has no items; that is, {.y  $\leftrightarrow$  0{1{.y. Thus:

```

a=: i. 3 2 3
a;({.a);({."2 a);({."1 a)
+-----+-----+-----+-----+
0 1 2	0 1 2	0 1 2	0 3
3 4 5	3 4 5	6 7 8	6 9
		12 13 14	12 15
6 7 8			
9 10 11			
12 13 14			
15 16 17			
+-----+-----+-----+-----+

]b=: ;/a
+-----+-----+-----+
|0 1 2|6 7 8|12 13 14|
|3 4 5|9 10 11|15 16 17|
+-----+-----+-----+

{.&> b
0 1 2
6 7 8
12 13 14

{. i.0 3
0 0 0

```

If x is an atom, x{.y takes from y an interval of |x items; beginning at the front if x is positive, ending at the tail if it is negative.

In an *overtake* (in which the number to be taken exceeds the number of items), extra items consist of *fills*; zeros if y is numeric, a: if it is boxed, spaces if literal, and s: ' ' if symbol. The fill atom f is also specified by *fit*, as in {.!f.

In general, if y is not an atom, x may be a list of length not more than \$\$y, and if y is an atom, it is replaced by ((#x)\$1)\$y. Element k produces (k{x){."(((\$y)-k) y; an infinite value is replaced by the length of the corresponding axis.

The following examples illustrate the use of the dyad *take*:

```

y=: i. 3 4
y;(2{.y);(5{.y);(_5{.y);(_6{.'abcd');(2 _3{.y)
+-----+-----+-----+-----+
|0 1 2 3|0 1 2 3|0 1 2 3|0 0 0 0| abcd|1 2 3|

```

|                                 |         |           |           |       |
|---------------------------------|---------|-----------|-----------|-------|
| 4 5 6 7                         | 4 5 6 7 | 4 5 6 7   | 0 0 0 0   | 5 6 7 |
| 8 9 10 11                       |         | 8 9 10 11 | 0 1 2 3   |       |
|                                 |         | 0 0 0 0   | 4 5 6 7   |       |
|                                 |         | 0 0 0 0   | 8 9 10 11 |       |
| +-----+-----+-----+-----+-----+ |         |           |           |       |

```

      2 {."1 y
0 1
4 5
8 9

```

```

    _ 2 {. y
0 1
4 5
8 9

```

```

      6{.'ab';'cde';'fghi'
+---+---+---+---+
|ab|cde|fghi|||
+---+---+---+---+

```

Map

{ ::    \_ 1 \_

Fetch

{ :: y has the same boxing as y and its elements are the paths to each leaf (each open array).

x{ :: y fetches a subarray of y according to path x ; the selection at each level is based on { and, except at the last level, must result in an atom.

Map and Fetch can be modeled as follows:

```
cat  =: { @: (i.&.>) @: $
mapp =: 4 : 'if. L. y. do. (<"0 x.,&.><"0 cat y.) mapp&.> y.
      else. >x. end.'
```

```
map  =: a:&mapp
fetch=: >@({&>/)@(<"0@|.@[ , <@]) " 1 _
```

The following phrases illustrate the use of Map and Fetch:

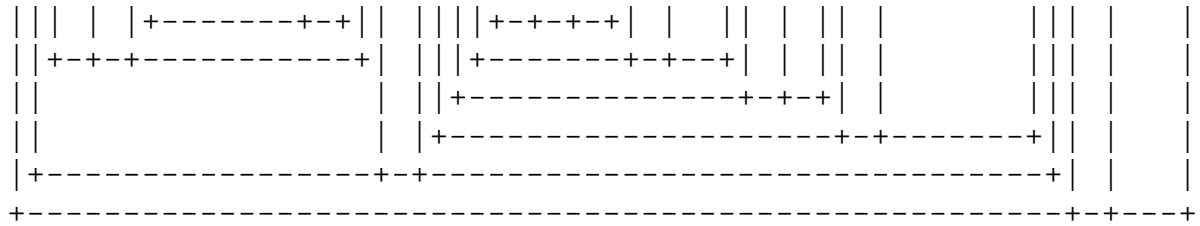
```
] y=: 1 2 3;4 5;i.4 5
+-----+-----+-----+
1 2 3	4 5	0 1 2 3 4
		5 6 7 8 9
		10 11 12 13 14
		15 16 17 18 19
+-----+-----+-----+
```

- (2;\_1 \_1){ :: y
- (\_1;3 4) { :: y
- { :: y
- { :: cat L: 0 y
- The number 19
- The number 19
- Paths to each open array
- Paths to each open scalar

] t=: 5!:2 <'fetch'

An array with an interesting structure

```
+-----+-----+-----+
+-----+-----+-----+	" 1 _																		
	+--+-----+--+	@	+-----+-----+--+																
		>	@	+-----+--+			+-----+-----+--+	,	+--+--+										
				+--+--+	/			+-----+-----+--+	@	[			<	@	]				
				{	&	>					+--+--+	@	.			+--+--+			
				+--+--+					<	" 0									
```



`(0;2;0;0;0){:: t`

Fetch the subarray corresp. to `<"0` in `t`

`(0;2;0;0;0;_1){:: t`

Fetch the `0` in that

`t ,&< L: 0 1 {:: t`

Label each leaf with its path

`< S: 0 t`

The boxed leaves of `t`

`< S: 1 {:: t`

The boxed paths of `t`

`t ,&< S: 0 1 {:: t`

A 2-column table of leaves and paths

`# 0: S: 0 t`

The number of leaves in `t`

**Item Amend**

$$m\} \quad \_ \_ \_$$
**Amend**

If  $m$  is numeric and  $z =: m\} y$ , then  $\$z$  equals  $\$m$ , which equals the shape of an *item* of  $y$ . The atom  $j\{z$  is  $j\{(j\{m)\}y$ . For example:

```

y=: a.{~(a.i.'A')+i.4 5
m=: 3 1 0 2 1
y ; m ; m}y
+-----+-----+-----+
ABCDE	3 1 0 2 1	PGCNJ
FGHIJ		
KLMNO		
PQRST		
+-----+-----+-----+

```

If  $m$  is not a gerund,  $x\ m\} y$  is formed by replacing by  $x$  those parts of  $y$  selected by  $m\&\{$ . Thus:

```

y; '%*(1 3;2 _1)} y
+-----+-----+
ABCDE	ABCDE
FGHIJ	FGH%J
KLMNO	KLMN*
PQRST	PQRST
+-----+-----+

```

$\$x$  must be a suffix of  $\$m\{y$ , and  $x$  has the same effect as  $(\$m\{y)\$,x$ .

Thus:

```

y; 'think' 1 2} y
+-----+-----+
ABCDE	ABCDE
FGHIJ	think
KLMNO	think
PQRST	PQRST
+-----+-----+

```

If  $m$  is a gerund, one of its elements determines the index argument to the adverb  $\}$ , and the others modify the arguments  $x$  and  $y$ :

```

x (v0`v1`v2)} y ↔ (x v0 y) (x v1 y)} (x v2 y)
(v0`v1`v2)} y ↔ (v1 y)} (v2 y)
(v1`v2)} y ↔ (v1 y)} (v2 y)

```

For example, the following functions  $E1$ ,  $E2$ , and  $E3$  interchange two rows of a matrix, multiply a row by a constant, and add a multiple of one row to another:

```

E1=: <@] C. [

```

```

E2=: f`g`[]
E3=: F`g`[]
f=: {:@] * {.@] { [
F=: [: +/ (1:,{:@]) * (}:{:@] { [)
g=: {.@]
M=: i. 4 5
M;(M E1 1 3);(M E2 1 10);(M E3 1 3 10)
+-----+-----+-----+-----+
+
| 0 1 2 3 4| 0 1 2 3 4| 0 1 2 3 4| 0 1 2 3
4|
| 5 6 7 8 9|15 16 17 18 19|50 60 70 80 90|155 166 177 188
199|
|10 11 12 13 14|10 11 12 13 14|10 11 12 13 14|
10 11 12 13 14|
|15 16 17 18 19| 5 6 7 8 9|15 16 17 18 19|
15 16 17 18 19|
+-----+-----+-----+-----+
+

```

**Item Amend**

u} \_ \_ \_

**Amend**

u} is defined in terms of the noun case m} , the verb u applying to the argument or arguments to provide the numeric indices required by it.

For example:

```

x=: 100 + i. 2 4
u=: */@$@] | (5: * i.@$@[)
y=: i. 3 2 4
x ; y ; (x u y) ; (x u} y)

```

|                 |             |           |              |
|-----------------|-------------|-----------|--------------|
| 100 101 102 103 | 0 1 2 3     | 0 5 10 15 | 100 105 2 3  |
| 104 105 106 107 | 4 5 6 7     | 20 1 6 11 | 4 101 106 7  |
|                 | 8 9 10 11   |           | 8 9 102 107  |
|                 | 12 13 14 15 |           | 12 13 14 103 |
|                 | 16 17 18 19 |           | 16 17 18 19  |
|                 | 20 21 22 23 |           | 104 21 22 23 |

The positions selected by x u} y may be made to depend on either or both of the arguments x and y , and related adverbs can be defined for convenient use in common cases. For example:

```

A=: @(i.@$@])
u=: (<0 1)&|:
x=: 'DIAG' [ y=: a. {~ (a. i. 'a') + i. 4 5
x;y;(x u A y);(x u A} y)

```

|      |       |           |       |
|------|-------|-----------|-------|
| DIAG | abcde | 0 6 12 18 | Dbcde |
|      | fghij |           | fIhij |
|      | klmno |           | klAno |
|      | pqrst |           | pqrGt |

Also see the case m} for the use of gerunds.

# Behead

$$\} \cdot \quad \_ \quad \underline{1} \quad \_$$

# Drop

} . drops the leading item of its argument.

$x$  } .  $y$  drops (at most)  $|x|$  items from  $y$ , dropping from the front if  $x$  is positive and from the tail if negative.

In general, if  $y$  is not an atom,  $x$  may be a list of length at most  $r := \$y$ , and the effect of element  $k$  is  $(k\{x\} \dots \{x\})$ . If  $y$  is an atom, the result is  $(0=x)y$ .

```

      ]y=: a. {~ (a. i. 'A') + i. 4 5
ABCDE
FGHIJ
KLMNO
PQRST

```

$$\mathbf{f} = \begin{Bmatrix} f_x \\ f_y \end{Bmatrix} ; \begin{Bmatrix} \cdot \\ \cdot \end{Bmatrix}$$

```

+-----+-----+
FGHIJ	ABCDE
KLMNO	
PQRST	
+-----+-----+

```

g=: } . , .@; { .  
g y

```

+-----+
| FGH IJ |
| KLMNO |
| PQRST |
+-----+
| ABCDE |
+-----+

```

```
(2}.y) ; (_2}.y) ; (6}.y) ; ($ 6}.y) ; (}. "1 y) ; (3}. "1 y)
+-----+-----+-----+---+-----+---+
```



|         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|
| KLMNO   | ABCDE   |         | 0 5     | BCDE    | DE      |
| PQRST   | FGHIJ   |         |         | GHIJ    | IJ      |
|         |         |         |         | LMNO    | NO      |
|         |         |         |         | QRST    | ST      |
| +-----+ | +-----+ | +-----+ | +-----+ | +-----+ | +-----+ |

**Curtail** } : \_

}:y drops the last item of y, and is equivalent to \_1 }. y. Thus:

```
]y=: a. {~ (a. i. 'A') + i. 4 5
ABCDE
FGHIJ
KLMNO
PQRST
```

```
f=: }: ; {:
f y
+-----+-----+
ABCDE	PQRST
FGHIJ	
KLMNO	
+-----+-----+
```

```
g=: }: ,.@; {:
g y
+-----+
|ABCDE|
|FGHIJ|
|KLMNO|
+-----+
|PQRST|
+-----+
```

```
h=: {. ,.@; }.
h y
+-----+
|ABCDE|
+-----+
|FGHIJ|
|KLMNO|
|PQRST|
+-----+
```

}:"1 y  
ABCD  
FGHI  
KLMN  
PQRS

## Rank

 $m^n$ 

The verb  $m^n$  produces the constant result  $m$  for each cell to which it applies. The rank used is  $3 \text{ \$&}. | . n$ . For example, if  $n=:2$ , the three ranks are  $2 \ 2 \ 2$ , and if  $n=:2 \ 3$ , they are  $3 \ 2 \ 3$ . A negative rank is complementary:  $m^{(-r)} y$  is equivalent to  $m^{(0>.(#\$y)-r)}\_y$ .

Thus:

```
v=: 2 3 5 7
m=: i. 2 3
m ; (m"0 v) ; (m"1 v) ; (m"1 m)
```

|   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |
| 3 | 4 | 5 | 3 | 4 | 5 | 3 | 4 | 5 | 3 | 4 | 5 |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   | 0 | 1 | 2 |   |   |   | 0 | 1 | 2 |
|   |   |   | 3 | 4 | 5 |   |   |   | 3 | 4 | 5 |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   | 0 | 1 | 2 |   |   |   |   |   |   |
|   |   |   | 3 | 4 | 5 |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   | 0 | 1 | 2 |   |   |   |   |   |   |
|   |   |   | 3 | 4 | 5 |   |   |   |   |   |   |

```
v m" 1 2 m
0 1 2
3 4 5
```

The verbs  $_9:$  through  $9:$  are *constant* verbs, equivalent to  $_9"$  through  $9"$ . For example:

```
odds=: 1: + 2: * i.
odds 5
1 3 5 7 9
```

Rank

u"n

The verb `u"n` applies `u` to each cell as specified by the rank `n`. The full form of the rank used is `3 $&. | . n`. For example, if `n=:2`, the three ranks are `2 2 2`, and if `n=: 2 3`, they are `3 2 3`. A negative rank is complementary: `u"(-r) y` is equivalent to `u"(0>.(#$y)-r)"_ y`.

`(| ; , ; , "2) y=: a. {~ (a.i.'A') + i. 2 3 4`

|      |          |          |          |          |          |
|------|----------|----------|----------|----------|----------|
| ABCD | ABCDEFGH | IJKLMNOP | QRSTUVWX | ABCDEFGH | IJKL     |
| EFGH |          |          |          | MNOP     | QRSTUVWX |
| IJKL |          |          |          |          |          |
|      |          |          |          |          |          |
| MNOP |          |          |          |          |          |
| QRST |          |          |          |          |          |
| UVWX |          |          |          |          |          |

`(<"0 ; <"1 ; <"2 ; <"3 ,&< <"_1) y`      Boxing of ranks `0 1 2 3 _1`

|   |   |   |   |      |      |      |      |      |      |      |      |
|---|---|---|---|------|------|------|------|------|------|------|------|
| A | B | C | D | ABCD | EFGH | IJKL | ABCD | MNOP | ABCD | ABCD | MNOP |
| E | F | G | H | MNOP | QRST | UVWX | IJKL | UVWX | IJKL | IJKL | UVWX |
| I | J | K | L |      |      |      |      |      | MNOP |      |      |
|   |   |   |   |      |      |      |      |      | QRST |      |      |
|   |   |   |   |      |      |      |      |      | UVWX |      |      |
| M | N | O | P |      |      |      |      |      |      |      |      |
| Q | R | S | T |      |      |      |      |      |      |      |      |
| U | V | W | X |      |      |      |      |      |      |      |      |

`('*#' , "0 1 ' abcde') ; (+/"2 i. 2 3 4) ; (+/"_1 i. 2 3 4)`

|         |    |    |    |    |    |    |    |    |
|---------|----|----|----|----|----|----|----|----|
| * abcde | 12 | 15 | 18 | 21 | 12 | 15 | 18 | 21 |
|---------|----|----|----|----|----|----|----|----|



## Assign Rank

`m"v`   `u"v`   `mv`   `lv`   `rv`

The verbs `m"v` and `u"v` are equivalent to `m"r` and `u"r`, where `r` is the list of ranks of `v`. The results may be examined by using the *basic characteristics* adverb `b.` to obtain ranks.

For example:

```
, b. 0
- - -

%. b. 0
2 _ 2

ravel=: , " %.

ravel b. 0
2 _ 2

ly=: a. {~ (a. i. 'A') + i. 2 3 4
ABCD
EFGH
IJKL

MNOP
QRST
UVWX

,Y
ABCDEFGH IJKLMN OPQRST UVWX

ravel y
ABCDEFGH IJKL
MNOPQRST UVWX
```

**Do**

". 1 \_ \_

**Numbers**

".y executes the sentence y. If the execution results in a noun, the result of ".y is that noun; if the execution result is a verb, adverb, or conjunction, the result of ".y is an empty vector.

x".y converts character array y into numbers. The shape of the result is (}:\$y),(1=n)}.n where n is the maximum number of numbers in any row. x is a scalar number used to replace illegal numbers and to pad narrow rows. In the conversion, the normal rules for numeric constants are relaxed as follows:

- the negative sign can be - or \_
- commas within numbers are ignored
- fractions need not have a digit 0 before the decimal point

For example:

```
". s=: '5 * a=: 3 + i. 6'
15 20 25 30 35 40

a
3 4 5 6 7 8

do=: ".
do t=: '3 % 5'
0.6
do |. t
1.66667
$ do ''
0

]program=: 'a=: 2^3' ,: '5*a'
a=: 2^3
```



5\*a

do program  
8 40

do 'sum=: +/'

sum 1 2 3 4  
10

s ; \_999". s=: '1 2 3', '-4 .5', ':bad 3,141'

```
+-----+-----+
1 2 3	1      2      3
-4 .5	_4    0.5 _999
bad 3,141	_999 3141 _999
+-----+-----+
```

**Default Format**

": \_ 1 \_

**Format**

Default output is identical to this monadic case, providing a minimum of one space between columns. For example:

```
]text=: ": i. 2 5
```

```
0 1 2 3 4
```

```
5 6 7 8 9
```

```
$ text
```

```
2 9
```

```
3 + text
```

```
|domain error
```

```
| 3 +text
```

```
'*#' ,. text
```

```
*0 1 2 3 4
```

```
#5 6 7 8 9
```

```
": 'abcd'
```

```
abcd
```

```
$ ": ''
```

```
0
```

$x":y$  produces a literal representation of  $y$  in a format specified by  $x$ . Each element of  $x$  is a complex number  $w + j \cdot d$ , controlling the representation of the corresponding column of  $y$  as follows:

$|w$  specifies the width allocated; if this space is inadequate, the entire space is filled with asterisks. If  $w$  is zero, enough space is allocated.

$|d$  specifies the number of digits following the decimal point (itself included only if  $d$  is not zero).

Any negative sign is placed just before the leading digit. If  $w > 0$  and  $d > 0$ , the result is right-justified in the space. Otherwise (if  $w < 0$  or  $d < 0$ ), the result is put in exponential form (with one digit before the decimal point) and is left-justified except for two fixed spaces on the left (including one for a possible negative sign).

The format  $w+d\%10$  (or its negation) is obsolescent; e.g. `_7.2` instead of `_7j2`.

See below for boxed arguments.

```
n ; 6j2 ": n=: % i. 2 4
```

```

+-----+-----+
|   _   1       0.5 0.333333|   _   1.00  0.50  0.33|
|0.25 0.2 0.166667 0.142857| 0.25  0.20  0.17  0.14|
+-----+-----+

```

```
(7j2 ":-n) ; (3j2 " : n)
```

```

+-----+-----+
|   _   _1.00 _0.50 _0.33|   _*****|
| _0.25 _0.20 _0.17 _0.14|*****|
+-----+-----+

```

```

6j3 0j_6 " : 1r2 ^ 1 1000 *"1 i.5 2
1.000 9.332636e_302
0.250 8.128549e_904
0.063 7.079811e_1506
0.016 6.166381e_2108
0.004 5.370801e_2710

```

The fit conjunction (!.) and [9!:10 p223](#) specify the number of digits for floating-point numbers. For example:

```

(" : ; " :!.6 ; " :!.4 ; " :!.15) %7
+-----+-----+-----+-----+
|0.142857|0.142857|0.1429|0.142857142857143|
+-----+-----+-----+-----+

```

For a boxed right argument, a two-element left argument specifies position in the display, using 0, 1, and 2 for top/center/bottom, and left/center/right. [9!:16 p223](#) and [9!:17 p223](#) specify the default positions. [9!:6 p223](#) and [9!:7 p223](#) specify the box drawing characters used.

```

x=: 2 3 $ (2 #&.> 1+i.6) $&.> 'abcdef'
(" : x) ,. (2 1 " : x)
+-----+-----+-----+-----+
a	bb	ccc			ccc	
	bb	ccc			bb	ccc
		ccc		a	bb	ccc
+-----+-----+-----+-----+						
dddd	eeee	ffffff			ffffff	
dddd	eeee	ffffff			eeee	ffffff
dddd	eeee	ffffff	dddd	eeee	ffffff	
dddd	eeee	ffffff	dddd	eeee	ffffff	
		eeee	ffffff	dddd	eeee	ffffff
			ffffff	dddd	eeee	ffffff

```

+-----+-----+-----++-----+-----+-----+

## Tie

$m`n \quad m`v \quad u`n \quad u`v$

In English, a gerund is a noun that carries the force of a verb, as does the noun *cooking* in *the art of cooking*. The tie applies to two verbs to produce a gerund. Gerunds are commonly used with *insert (/)* and with *agenda (@.)*:

```
]g=: +`*
+---+
|+|*|
+---+

(g/1 2 3 4 5) ; (1+2*3+4*5)
+---+
|47|47|
+---+
```

More generally, tie produces gerunds as follows:  $u`v$  is  $au,av$ , where  $au$  and  $av$  are the (boxed noun) *atomic representations* ([5!:1 p220](#)) of  $u$  and  $v$ .

Moreover,  $m`n$  is  $m,n$  and  $m`v$  is  $m,av$  and  $u`n$  is  $au,n$ . See [Bernecky and Hui \[12\] p212](#). Gerunds may also be produced directly by boxing. Thus:

```
]h=: '+' ; '*'
+---+
|+|*|
+---+

h/1 2 3 4 5
47
```

The atomic representation of a noun (used so as to distinguish a noun such as '+' from the verb +) is given by the following function:

```
(ar=: [: < (,'0')"_ ; ]) '+'
+-----+
|+---+|
||0|+||
|+---+|
+-----+
```

```
      *^(ar '+' )
+-+-----+
*	+-+--+			
		0	+	
	+-+--+			
+-+-----+
```

## Evoked Gerund

$m^{\cdot} : n$      $\_ \_ \_$

This conjunction is defined for three cases:

- $m^{\cdot} : 0$     *Append*    Appends the results of the individual verbs; the ranks are the maxima over their ranks.
- $m^{\cdot} : 3$     *Insert*    Inserts verbs between items. Equivalent to  $m/$
- $m^{\cdot} : 6$     *Train*    Result is the train of individual verbs.

For example:

```
<+:`-:.`% `: 0 a=: 1 2 3 4 5
+-----+
|  2    4          6    8  10 |
| 0.5    1        1.5    2 2.5 |
|  1 0.5 0.333333 0.25 0.2 |
+-----+

(+ b.0) ; (%. b.0) ; (+`%.`:0 b.0)
+-----+
| 0 0 0 | 2 _ 2 | _ _ _ |
+-----+

(+`* `:3 a) ; (+`*/a) ; (1+2*3+4*5)
+---+---+
| 47 | 47 | 47 |
+---+---+

(+`*`- `: 6 a) ; ((+ * -) a)
+-----+-----+
| _1 _4 _9 _16 _25 | _1 _4 _9 _16 _25 |
+-----+-----+
```

**Atop**

u@v    mv   lv   rv

$u@v \ y \leftrightarrow u \ v \ y$ . For example,  $+:@-7$  is  $\_14$  (double the negation). Moreover, the monadic uses of  $u@v$  and  $u\&v$  are equivalent.

$x \ u@v \ y \leftrightarrow u \ x \ v \ y$ . For example,  $3 \ +:@-7$  is  $\_8$  (double the difference).

Because adverbs and conjunctions are (as stated more precisely in [Section II E p67](#)) executed before verbs, phrases involving them are commonly used in trains without parentheses. For example:

```
mean=: +/ % #
mean 1 2 3 4
2.5
```

```
f=: +:@*: +/ -:@%:
f 1 2 3 4
2.5 2.70711 2.86603 3
8.5 8.70711 8.86603 9
18.5 18.7071 18.866 19
32.5 32.7071 32.866 33
```

Addition table of doubled square and halved sqrt

Because a conjunction applies to the entity immediately to its right, expressions to the right of conjunctions commonly require parenthesization. For example:

```
g=: *:@(+/)
h=: *:@+/
g 1 2 3 4
100
h 1 2 3 4
6770404
```

```
k=: *:@+
k/ 1 2 3 4
6770404
```

Compare the behaviour of  $@$  with that of  $@:$ . They differ only in the ranks of the verbs that they produce.



## Agenda

`m@.n`   `m@.v`   `mv`   `lv`   `rv`

`m@.n` is a verb defined by the gerund `m` with an *agenda* specified by `n`; that is, the verb represented by the train selected from `m` by the indices `n`. If `n` is boxed, the train is parenthesized accordingly. The case `m@.v` uses the result of the verb `v` to perform the selection.

For example:

```
dorh=: +: ` -: @. (]>9:)           Double or halve (Case statement)
dorh " 0 primes=: 2 3 5 7 11 13 17 19
4 6 10 14 5.5 6.5 8.5 9.5

_:`%:`*: @. * "0 a=: 2 1 0 _1 _2
1.41421 1 _ 1 4

g=: +`-`*
x=: 1 2 3 [ y=: 6 5 4
(x g@.2: y)
6 10 12

(] * <:) y=: 5 4 3 2 1 0           Basis of factorial
20 12 6 2 0 0

1:`(] * <:) @. (1: < ]) "0 y       Case statement
20 12 6 2 1 1

1:`(] * $:@<:)@.(1: < ]) "0 y     Self-reference for recursion
120 24 6 2 1 1

+`-`*`% @. (1 0 3;2 0)
(- + %) (* +)

3 +`-`*`% @. (1 0 3;2 0) 4
_12.8125
```

**At** $u@:v \quad \_ \_ \_$ 

$@:$  is equivalent to  $@$  except that ranks are infinite.

For example:

```
x=: 1 2 3 4
y=: 7 5 3 2
x */ @: + y
2016
```

Applies product over sums to the entire lists

```
x */ @ + y
8 7 6 6
```

Applies product over sums to each item of the list

```
+ b. 0
0 0 0
```

```
*/ @: + b. 0
_ _ _
```

```
*/ @ + b. 0
0 0 0
```

**Bond**  $m \& v$   $u \& n$  —  
— —

$m \& v \ y$  is defined as  $m \ v \ y$ ; that is, the left argument  $m$  is bonded with the dyad  $v$  to produce a monadic function.

$x \ m \& v \ y \leftrightarrow m \& v^{:x} \ y}$   
 $x \ u \& n \ y \leftrightarrow u \& n^{:x} \ y}$

For example:

```
10&^. 2 3 10 100 200
0.30103 0.477121 1 2 2.30103
```

```
base10log=: 10&^.
base10log 2 3 10 100 200
0.30103 0.477121 1 2 2.30103
```

```
sine=: 1&o.
sine o. 0 0.25 0.5 1.5 2
0 0.707107 1 _1 0
```

Similarly,  $u \& n \ y$  is defined as  $y \ u \ n$ ; in other words, as the dyad  $u$  provided with the right argument  $n$  to produce a monadic function. For example:

```
^&3 (1 2 3 4 5)
1 8 27 64 125
```

```
^&2 3"0 (1 2 3 4 5)
1 1
4 8
9 27
16 64
25 125
```

Use of the bond conjunction is often called *Currying* in honor of Haskell Curry.

The phrase  $x \leftarrow f @ [0 \ y]$  is equivalent to  $f^x y$ , apply the monad  $f$   $x$  times to  $y$ . For example:

```
fib=: (0 1,:1 1)&(+/ .* )@[0 & 0 1
fib i.10
0 1
1 1
1 2
2 3
3 5
5 8
8 13
13 21
21 34
34 55
```

## Compose

$$u \& v \quad m v \quad m v \quad m v$$

$u \& v \ y \leftrightarrow u \ v \ y$ . Thus  $+: \&-$  7 is  $\_14$  (double the negation). Moreover, the monads  $u \& v$  and  $u @ v$  are equivalent.

$x \ u \& v \ y \leftrightarrow (v \ x) \ u \ (v \ y)$ . For example,  $3 \ +\&! \ 4$  is 30, the sum of factorials.

The monadic case is equivalent to the composition used in mathematics, but the dyadic case opens up other possibilities. For example:

$3 \ +\&^\cdot \ 4$   
2.48491

Sum of natural logarithms

$^\wedge \ 3 \ +\&^\cdot \ 4$   
12

Multiplication using natural logs

$3 \ +\&(10\&^\cdot) \ 4$   
1.07918

Sum of base ten logarithms

$10 \ ^\wedge \ 3 \ +\&(10\&^\cdot) \ 4$   
12

Multiplication using base ten logs

$3 \ +\&^\cdot \ 4$   
12

See the related conjunction *under*  $(\&^\cdot)$

$3 \ +\&^\cdot(10\&^\cdot) \ 4$   
12

Compare the behaviour of  $\&$  with that of  $\&^\cdot$ . They differ only in the ranks of the verbs that they produce.

**Under**

$$u \& . v \quad m v \quad m v \quad m v$$

The verb  $u \& . v$  is equivalent to the composition  $u \& v$  except that the verb obverse to  $v$  is applied to the result for each cell. The obverse is normally the inverse, as discussed more fully under the power conjunction  $^:$ .

3 +&.^ . 4                      Inverse of natural log is the exponential  
12

(^.^:\_1) (^ . 3) + (^ . 4)  
12

(<b), <|. b=: 1 2 3 ; 2 3 5 7 ; 'abcde'  
+-----+-----+-----+  
|+-----+-----+-----+|+-----+-----+-----+| | | | | | | | |
||1 2 3|2 3 5 7|abcde|||abcde|2 3 5 7|1 2 3||  
|+-----+-----+-----+|+-----+-----+-----+|  
+-----+-----+-----+

each=: &.>                      An adverb  
(<|. & . > b), (<|. each b)      Reversal under open  
+-----+-----+-----+  
|+-----+-----+-----+|+-----+-----+-----+| | | | | | | | |
||3 2 1|7 5 3 2|edcba|||3 2 1|7 5 3 2|edcba||  
|+-----+-----+-----+|+-----+-----+-----+|  
+-----+-----+-----+

In mathematics, certain cases of *under* are called *dual* or, *dual with respect to*:

f=: +. & . -.                      Dual with respect to boolean negation  
f/~ d=: 0 1  
0 0  
0 1

D=: & . -.                      The adverb dual with respect to negation  
(+.D/~d);(\*./~d);(=D/~d);(~:/~d)  
+---+---+---+---+  
|0 0|0 0|0 1|0 1|  
|0 1|0 1|1 0|1 0|

+---+---+---+---+

DWL=: & . ^ .

Dual with respect to natural logarithm

DAN=: & . -

Dual with respect to arithmetic negation

( 3 + DWL 4 ) , ( 3 \* 4 ) , ( 3 < . DAN 4 ) , ( 3 > . 4 )

12 12 4 4

Under

$$u \& . : v \quad \_ \_ \_$$

$u \& . : v$  is equivalent to  $u \& . (v \_)$  .

$$\begin{array}{l} +/\& . : ^ . \ 2 \ 3 \ 4 \\ 24 \\ +/\& . ^ . \ 2 \ 3 \ 4 \\ 2 \ 3 \ 4 \end{array}$$



## Appose

$$u \& : v \quad \_ \_ \_$$

$\&:$  is equivalent to  $\&$  except that the ranks of the resulting function are infinite; the relation is similar to that between  $@:$  and  $@$  .

For example:

```
a=: 'abcd' ; 'efgh'
b=: 'ABCD' ; 'EFGH'
```

```
      a
+-----+-----+
|abcd|efgh|
+-----+-----+
```

```
      b
+-----+-----+
|ABCD|EFGH|
+-----+-----+
```

```
      a ,&:> b
abcd
efgh
ABCD
EFGH
```

```
      a ,&> b
abcdABCD
efghEFGH
```

```
      > b. 0
0 0 0
```

```
      , & > b. 0
0 0 0
```

```
      , &: > b. 0
_ _ _
```

**Roll**

? ? . 0 0 0

**Deal**

? y yields a uniform random selection from the population i.y. The *random seed* used begins at 7^5 (16807) and is unaffected by the use of ? . .

x ? y is a list of x items randomly chosen *without repetition* from i.y. The corresponding use of ? . does not affect the random seed.

Use of the fixed seed function ? . makes examples reproducible:

```
? . 6
0
```

```
? . 6 6 6 6 6 6 6 6
0 4 2 3 1 0 4 4
```

```
6 ? . 6
5 1 2 4 3 0
```

A random permutation

```
mean=: +/ % #
mean ? . 1000 # 6
2.497
```

```
m=: ? . 4 4 $ 9
m
1 6 4 4
1 0 6 6
8 3 4 7
0 0 4 6
```

A random matrix for experimentation

```
-/ . * m
588
```

The determinant of m

```
f=: ? .@$ % ] - 1:
<3 6 f 9
resolution 9
```

Random 3 by 6 table in range zero to one with

```
+-----+
|0.125 0.75 0.5 0.5 0.125 0|
| 0.75 0.75 1 0.375 0.5 0.875|
| 0 0 0.5 0.75 0 0.375|
+-----+
```

The *random seed* (a beginning value for the pseudo-random number generator) is set by the foreign conjunction using  $9! : 1$ , and queried with  $9! : 0$ .

## Alphabet/Ace

a.    a:

a. is a list of the elements of the alphabet; it determines the *collating sequence* used in grading and sorting (/: and \:). The content of a. as well as the ordering of its elements may differ on different computing systems.

The number of elements in the alphabet is given by \$a., and a 32-column display is given by an expression such as 8 32\$a. . The inclusion of certain *control* characters (such as the carriage return) and *non-printing* characters make such a display difficult to decipher, but the major alphabet is usually given by:

```
1 2 3 { 8 32 $ a.
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~•
```

The index of the carriage return is commonly 13 (as may be tested by entering 13{a.), and the indices of the space and other characters may be determined as illustrated below:

```
a. i. 'aA +-*%'
97 65 32 43 45 42 37
```

The *ace* (a unit, from Latin *as*) is denoted by a: . It is the boxed empty list <\$0 .

**Anagram Index**

A. 1 0 \_

**Anagram**

If  $T$  is the table of all  $n!$  permutations of order  $n$  arranged in lexical order (i.e.,  $i:T$  is  $i.n!$ ), then  $k$  is said to be the *anagram index* of the permutation  $k\{T$ .

A. applied to a cycle or direct permutation yields its anagram index:

A. 0 3 2 1 is 5, as are A. 3 2 1 and A.<3 1 and A.0;2;3 1.

The expression  $k A. b$  permutes items of  $b$  by the permutation of order  $\#b$  whose anagram index is  $k$ .

For example:

(A. 0 3 2 1) , (A. <3 1)  
5 5

A. |. i.45  
119622220865480194561963161495657715064383733759999999999

<: ! 45x  
119622220865480194561963161495657715064383733759999999999

tap=: i.@! A. i.

Table of all permutations

(tap 3);(/: tap 3);({/\ tap 3);(/:{/\ tap 3)  
+-----+-----+-----+-----+  
|0 1 2|0 1 2 3 4 5|0 1 2|0 1 5 2 4 3|  
|0 2 1| |0 2 1| |  
|1 0 2| |1 2 0| |  
|1 2 0| |2 0 1| |  
|2 0 1| |1 2 0| |  
|2 1 0| |1 0 2| |  
+-----+-----+-----+-----+

In particular, 1 A. b transposes the last two items of b, and \_1 A. b reverses the list of items, and 3 A. b and 4 A. b rotate the last three items of b. For example:

```

b=: 'ABCD'

(0 3 2 1{b);(0 3 2 1 C.b);((<3 1)C.b);(3 4 A.b)
+-----+-----+-----+-----+
|ADCB|ADCB|ADCB|ACDB|
|      |      |      |ADBC|
+-----+-----+-----+-----+

(_19 5 A. b) ; (_19 |~ ! # b)
+-----+--+
|ADCB|5|
|ADCB| |
+-----+--+

```

Boolean

m b. \_ 0 0

The monad is equivalent to a zero left argument; that is,  
m b. y ↔ 0 m b. y

If *f* is a dyadic boolean function and *d* =: 0 1, then *d f* / *d* (or *f* / ~*d*) is its complete table. For example the tables for *or*, *nor*, *and*, and *not-and* appear as follows:

(+./~ ; +:/~ ; \*./~ ; \*/~)

d=: 0 1

|                   |   |   |   |   |   |   |   |
|-------------------|---|---|---|---|---|---|---|
| +---+---+---+---+ |   |   |   |   |   |   |   |
| 0                 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1                 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| +---+---+---+---+ |   |   |   |   |   |   |   |

If ordered by their tables, each of the sixteen possible boolean dyads can be characterized by its index *k*; the phrase *k b.* produces the corresponding function. Moreover, negative indexing and array arguments may be used.

*m* = 16 + *k* specifies bitwise boolean functions on integer arguments. An argument atom is treated as a list of length *w* of bits, where *w* is the word size of the underlying machine. For example, 17 *b.* specifies bitwise *and*.

Finally, 32 *b.* specifies *rotate*, 33 *b.* specifies *shift*, and 34 *b.* specifies *signed shift*.

The following table lists all the possible boolean functions:

|    |     |    | Ravelled |   |   |   |              |
|----|-----|----|----------|---|---|---|--------------|
|    | m   |    | Table    |   |   |   | Function     |
| 0  | _16 | 16 | 0        | 0 | 0 | 0 | 0            |
| 1  | _15 | 17 | 0        | 0 | 0 | 1 | x *. y       |
| 2  | _14 | 18 | 0        | 0 | 1 | 0 | x > y        |
| 3  | _13 | 19 | 0        | 0 | 1 | 1 | x            |
| 4  | _12 | 20 | 0        | 1 | 0 | 0 | x < y        |
| 5  | _11 | 21 | 0        | 1 | 0 | 1 | y            |
| 6  | _10 | 22 | 0        | 1 | 1 | 0 | x ~: y       |
| 7  | _9  | 23 | 0        | 1 | 1 | 1 | x +. y       |
| 8  | _8  | 24 | 1        | 0 | 0 | 0 | x +: y       |
| 9  | _7  | 25 | 1        | 0 | 0 | 1 | x = y        |
| 10 | _6  | 26 | 1        | 0 | 1 | 0 | -. y         |
| 11 | _5  | 27 | 1        | 0 | 1 | 1 | x >: y       |
| 12 | _4  | 28 | 1        | 1 | 0 | 0 | -. x         |
| 13 | _3  | 29 | 1        | 1 | 0 | 1 | x <: y       |
| 14 | _2  | 30 | 1        | 1 | 1 | 0 | x *: y       |
| 15 | _1  | 31 | 1        | 1 | 1 | 1 | 1            |
|    |     | 32 |          |   |   |   | rotate       |
|    |     | 33 |          |   |   |   | shift        |
|    |     | 34 |          |   |   |   | signed shift |

Further examples:

```
(7 b./~ ; 8 b./~ ; 1 b./~ ; 14 b./~) d=: 0 1
```

```
+-----+
|0 1|1 0|0 0|1 1|
|1 1|0 0|0 1|1 0|
+-----+
```

```
(_1 b./~ ; _3 b./~ ; _15 b./~) d
```

negative indexing

```
+-----+
|1 1|1 1|0 0|
|1 1|0 1|0 1|
+-----+
```

```
(<"2) 2 0 1 | : 7 8 1 15 b./~ d
```

array arguments

```
+-----+
|0 1|1 0|0 0|1 1|
```



```
|1 1|0 0|0 1|1 1|
+---+---+---+---+
```

```
12345 (17 b.) 67890
48
```

```
f=: (32#2)&#: { '.x' "_
f 12345 67890 48
.....xx.....xxx..x
.....x....x..x..xx..x.
.....xx.....
```

bitwise *and*

```
_12345 (23 b.) 67890
_12297
```

```
f _12345 67890 _12297
xxxxxxxxxxxxxxxxxxxxx..xxxxxx...xxx
.....x....x..x..xx..x.
xxxxxxxxxxxxxxxxxxxxx..xxxxxx..xxx
```

bitwise *or*

```
20 b./~ i.10
0 1 2 3 4 5 6 7 8 9
0 0 2 2 4 4 6 6 8 8
0 1 0 1 4 5 4 5 8 9
0 0 0 0 4 4 4 4 8 8
0 1 2 3 0 1 2 3 8 9
0 0 2 2 0 0 2 2 8 8
0 1 0 1 0 1 0 1 8 9
0 0 0 0 0 0 0 0 8 8
0 1 2 3 4 5 6 7 0 1
0 0 2 2 4 4 6 6 0 0
```

bitwise *less than* table

```
23 b./\ 2^i.10
1 3 7 15 31 63 127 255 511 1023
```

cumulative bitwise *or*

```
_5 (33 b.) 12345
385
```

```
f 12345 385
.....xx.....xxx..x
.....xx.....x
```

shift

```
_5 (33 b.) _12345
134217342
```

```
f _12345 134217342
xxxxxxxxxxxxxxxxxxxxx..xxxxxx...xxx
.....xxxxxxxxxxxxxxxxx..xxxxxx.
```

shift

```
_5 (34 b.) _12345
```

signed shift

\_386

f \_12345 \_386

xxxxxxxxxxxxxxxxxxxxx..xxxxxx...xxx

xxxxxxxxxxxxxxxxxxxxx..xxxxxx.

## Basic Characteristics

$u \ b. \ \_$

$u \ b. \ y$  gives the obverse of  $u$  if  $y$  is  $\_1$ ; its ranks if  $y$  is  $0$ ; and its identity function if  $y$  is  $1$ .

For example:

```
^ b. _1
^.
```

```
^ b. 0
0 0 0
```

```
^ b. 1
$&1@({}.@$)
```

```
g=: +&2@(*&3@*:)
ly=: g 5
77
```

```
g ^:_1 y
5
```

```
g b. _1
%:@(%&3)@(-&2)
```

```
%:@(%&3)@(-&2) y
5
```

```
g b. 0
0 0 0
```

## Characteristic or Eigenvalues

$C = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$

$C.Y$  yields the *characteristic, own, or eigen* values of its argument, arranged in ascending order on imaginary part within real within magnitude. An atom or list  $Y$  is treated as the table  $,.Y$ .

$0.C.Y$  is a diagonal matrix with the eigenvalues  $C.Y$  on the diagonal. Also,  $_1.C.Y$  and  $1.C.Y$  are the left and right eigenvectors. If  $i =: _1 0 1$ , then  $+/. . */ i.C.Y$  is  $Y$ .

Not implemented in Release 4.01.

## Cycle

C. 1 1 \_

## Permute

If  $p$  is a permutation of the atoms of  $i.n$ , then  $p$  is said to be a permutation vector of order  $n$ , and if  $n=\#b$ , then  $p\{b$  is a permutation of the items of  $b$ .

$C.p$  yields a list of boxed lists of the atoms of  $i.\#p$ , called the *standard cycle representation* of the permutation  $p$ . Thus, if  $p=:4\ 5\ 2\ 1\ 0\ 3$ , then  $C.p$  is  $(,2);4\ 0;5\ 3\ 1$  because the permutation  $p$  moves to position 2 the item 2, to 4 the item 0, to 0 the item 4, to 5 the item 3, to 3 the item 1, and to 1 the item 5. The monad  $C.$  is self-inverse; applied to a standard cycle it gives the corresponding direct representation.

A given permutation could be represented by cycles in a variety of ways; the standard form is made unique by the following restrictions: the cycles are disjoint and exhaustive (i.e., the atoms of the boxed elements together form a permutation vector); each boxed cycle is rotated to begin with its largest element; and the boxed cycles are put in ascending order on their leading elements.

$C.$  is extended to non-negative non-standard cases by treating any argument  $q$  as a representation of a permutation of order  $1+>./q$ .

If  $p$  and  $c$  are standard and cycle representations of order  $\#b$ , then  $p\ C.b$  and  $c\ C.b$  produce the permutation of  $b$ . The arguments  $p$  and  $c$  can be *non-standard* in ways to be defined. In particular, negative integers down to  $-\#b$  may be used, and are treated as their residues modulo  $\#b$ .

If  $q$  is not boxed, and the elements of  $(\#b)|q$  are distinct, then  $q\ C.b$  is equivalent to  $p\ C.b$ , where  $p$  is the standard form of  $q$  that is given by  $p=:((i.n)-.n|q),n|q$ , for  $n=\#b$ . In other words, positions occurring in  $q$  are moved to the tail end. If  $q$  is boxed, the elements of  $(\#b)|>j\{q$  must be distinct for each  $j$ , and the boxes are applied in succession. For example:

```
(2 1;3 0 1) C. i.5
1 2 3 0 4
```

```
(<2 1) C. (<3 0 1) C. i.5
1 2 3 0 4
```

```
q=: C. p=: 1 2 3 0 4 [ a=:
'abcde'
q ; (q C. a) ; (p C. a) ; (p
{ a)
```

The monad `C.!.` computes the *parity* of a permutation `p`; it is `1` or `_1` as the number is even or odd of pairwise interchanges necessary to get `p` from the identity permutation `i.#p` (and `0` if `p` is not a permutation). For example:

```
] x=: 2 , (i.4) ,: 1 0 2 3
2 2 2 2
0 1 2 3
1 0 2 3
C.!.
```

```
+-----+-----+-----+-----+
+-----+--+	bcdae	bcdae	bcdae			
	3 0 1 2	4				
+-----+--+						
+-----+-----+-----+-----+
```

```
a ; (<0 _1) C. a
+-----+-----+
|abcde|ebcda|
+-----+-----+
```

Further examples:

```
]p=: 22 ? . 22
19 5 10 8 14 16 20 4 0 18 15 1 9 12 3 2 11 7 17 21 13 6
```

A random permutation of order 22

```
C. p
```

Its cycles

```
+-----+-----+-----+-----+
|15 2 10|16 11 1 5|21 6 20 13 12 9 18 17 7 4 14 3 8 0 19|
+-----+-----+-----+-----+
```

```
*./ #&> C. p
60
```

LCM of the cycle lengths

```
# ~. p&{^(i.200) i.#p
60
```

Size of the subgroup generated by `p`

The verb `CT` computes the *complete tensor* of order `n` as a sparse array; entry `(<i){CT n` is the parity of the index `i`.

```
CT=: 3 : '(C.!.
```

```
CT 3
0 1 2 | 1
0 2 1 | _1
1 0 2 | _1
1 2 0 | 1
2 0 1 | 1
2 1 0 | _1
```

$$(\$.^{:\_1 \text{ CT } 3}) ; , "2 \text{ ' ' } , "1 \text{ '012'} \{ \sim > \{ i . \& . > \$ \sim 3$$

|    |    |    |     |     |     |
|----|----|----|-----|-----|-----|
| 0  | 0  | 0  | 000 | 001 | 002 |
| 0  | 0  | 1  | 010 | 011 | 012 |
| 0  | _1 | 0  | 020 | 021 | 022 |
| 0  | 0  | _1 | 100 | 101 | 102 |
| 0  | 0  | 0  | 110 | 111 | 112 |
| 1  | 0  | 0  | 120 | 121 | 122 |
| 0  | 1  | 0  | 200 | 201 | 202 |
| _1 | 0  | 0  | 210 | 211 | 212 |
| 0  | 0  | 0  | 220 | 221 | 222 |

(CT 3) -: C.!.2&> { i.&.> \$~ 3  
1

] m=: ? . 3 3\$10  
1 7 4  
5 2 0  
6 6 9

+ / , (CT #m) \* \* // m  
\_225  
- / . \* m  
\_225

Determinant

**Derivative**

u d. n 0

u d. n is like u D. n except that u is treated as a rank-0 function.

```

      *: d. 1
+:
      *: d. 2
2"0
      (1: + _3&* + *: ) d. 1
_3 2&p.
      (1: + _3&* + *: ) d. 2
2"0
      (1: + _3&* + *: ) d. 3
0"0
      ^. d. 1
%
      (^. * *: ) d. 1
(% * *: ) + ^. * +:
      (^. % *: ) d. 1
((% * *: ) - ^. * +: ) % 0 0 0 0 1&p.
      (^. @ *: ) d. 1
+: * %@*:
      *: d. _1
0 0 0 0.33333333333333331&p.
      % d. _1
^..
      *: d. 1 x=: _3 _2 _1 0 1 2 3
_6 _4 _2 0 2 4 6
+: x

```



\_6 \_4 \_2 0 2 4

## Derivative

$m \ D. \ n \quad u \ D. \ n \quad \mu$

$u \ D. \ n$  is the  $n$ -th derivative of  $u$ , and  $u \ v \ D. \ n$  is  $u$  with an assigned  $n$ -th derivative  $v$ . For example:

```
(cube D.1;cube D.2; (cube=: ^&3"0) D.3)y=: 2 3 4
+-----+-----+-----+
|12 27 48|12 18 24|6 6 6|
+-----+-----+-----+
```

The derivative applies to constant functions, polynomials, the exponential  $^$ , the integral powers  $^&n$ , and those assigned by  $u \ v \ D. \ n$ . It also applies to functions derived from these by addition, subtraction, multiplication, and division ( $u+v$ , etc.); by the composition  $u@v$ ; and by the inverse  $u^:_1$ . Since functions such as  $j.$  and  $-$  (negation) and  $^:$  (square root) and  $1&o.$  (sin) and  $6&o.$  (cosh) may all be so derived, they are also in the domain of the derivative. Others are treated by approximation. The derivative of an arbitrary function may also be treated by a polynomial approximation, (provided by the matrix divide), or by approximations using the secant slope  $D:$ .

If the argument rank of  $u$  is  $a$  and the result rank is  $r$ , then the argument rank of  $u \ D.1$  is also  $a$ , but its result rank is  $r+a$ : the result of  $u \ D.1$  is the derivative of each atom of the result of  $u$  with respect to each element of its argument, giving what is commonly referred to as the *partial derivatives*. For example:

```
volume=: */"1
VOLUMES=: */\"1
(volume;volume D.1;VOLUMES;VOLUMES D.1) y
+---+-----+-----+-----+
24	12 8 6	2 6 24	1 3 12			
						0 2 8
						0 0 6
+---+-----+-----+-----+

determinant=: -/ . *
permanent=: +/ . *
```

```
( ) ; (determinant D.1) ; (permanent D.1) ) m = : * : i . 3 3
```

|    |    |    |      |      |      |      |      |      |
|----|----|----|------|------|------|------|------|------|
| 0  | 1  | 4  | _201 | 324  | _135 | 2249 | 1476 | 1017 |
| 9  | 16 | 25 | 132  | _144 | 36   | 260  | 144  | 36   |
| 36 | 49 | 64 | _39  | 36   | _9   | 89   | 36   | 9    |

The adverbs `D=: 1 : 'x.'"0 D.1'` and `VD=: 1 : 'x.'"1 D.1'` assign ranks to their arguments, then take the first derivative; they are convenient for use in scalar and vector calculus:

```
sin=: 1&o.
x=: 0.5p1 _0.25p1
(* /\ VD y) ; (sin x) ; (sin D x) ; (sin D D x)
```

|   |   |    |   |           |   |          |    |          |
|---|---|----|---|-----------|---|----------|----|----------|
| 1 | 3 | 12 | 1 | _0.707107 | 0 | 0.707107 | _1 | 0.707107 |
| 0 | 2 | 8  |   |           |   |          |    |          |
| 0 | 0 | 6  |   |           |   |          |    |          |

$u \ D: \ n \ \mu \ \mu$ 

## Secant Slope

$x \ u \ D: \ 1 \ y$  is the *secant slope* of the function  $u$  through the points  $y$  and  $y+x$ . The secant slope is generalized to the case  $x \ u \ D: \ n \ y$  in the manner of the derivative  $D: \ .$ . The argument  $x$  may be a list, giving several slopes.

In the general case, each item of  $x$  has the shape  $\{ . \ \$\$ "r \ y$ , where  $r$  is the rank of  $u$ , therefore specifying the increment in each possible direction. An argument  $x$  of lower rank is extended in the usual manner. For example,  $x=: \ 1e\_8$  provides the same increment in each direction and, because of the small magnitude, yields an approximation to the derivative.

```
1 log D:1 y
0.405465 0.287682 0.223144
```

```
incr=: 1 0.1 0.01 1e_8
incr log D:1/y
0.405465 0.287682 0.223144
0.487902 0.327898 0.246926
0.498754 0.332779 0.249688
      0.5 0.333333      0.25
```

```
log D.1 y
0.5 0.333333 0.25
%y
0.5 0.333333 0.25
```

```
f=: +/@: *: "1
g=: +/@: *: "\"1
```

```

      (f y) ; (1 f D:1 y) ; (1 0.1 1e_8 f D:1 y)
+---+-----+-----+
29	5 7 9	5 0.61 8e_8
		50 6.1 8e_7
		5e8 6.1e7 8
+---+-----+-----+

```

```

      (g y) ; (1 g D:1 y)
+-----+-----+
4 13 29	5 0 0
	5 7 0
	5 7 9
+-----+-----+

```

**Raze In**

e.    \_ \_ \_

**Member (In)**

`e.y` produces a boolean result that determines for each atom of `y` whether its open contains an item of the raze of `y`.

If `x` has the shape of an item of `y`, then `x e. y` is 1 if `x` matches an item of `y`. In general,  $x e. y \leftrightarrow (\#y) > y \text{ i. } x$ .

The fit conjunction provides tolerant comparison, as in `e.!t`.

For example:

```
]y=: 'abc' ; 'dc' ; 'a'
+---+---+--+
|abc|dc|a|
+---+---+--+
```

```
i y
abcdca
```

```
e. y
1 1 1 0 1 1
0 0 1 1 1 0
1 0 0 0 0 1
```

```
f=: ] e.~&>/ ;
f y
1 1 1 0 1 1
0 0 1 1 1 0
1 0 0 0 0 1
```

```
'cat' e. 'abcd'
1 1 0
```

```
]z=: 2 3$'catdog'
cat
dog
```

```
'cat' e. z
1
```

## Member of Interval

E.    — —

The *ones* in  $x \text{ E. } y$  indicate the beginning points of occurrences of the pattern  $x$  in  $y$ .

For example:

```
'co' E. 'cocoa'
1 0 1 0 0
```

```
s #~ -. '**' E. s=: 'Remove***multiple**stars.'
Remove*multiple*stars.
```

```
] x =: 0 1 2 ,: 2 3 4
0 1 2
2 3 4
```

```
] y=: 5 | i. 5 7
0 1 2 3 4 0 1
2 3 4 0 1 2 3
4 0 1 2 3 4 0
1 2 3 4 0 1 2
3 4 0 1 2 3 4
```

```
x E. y
1 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 0 0 0
0 0 0 0 1 0 0
0 0 0 0 0 0 0
```

```
($x) x&-: ;. 3 y
1 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 0 0 0
0 0 0 0 1 0 0
0 0 0 0 0 0 0
```

**Fix**

m f.    u f.

If  $x$  is a proverb, then  $y =: x \text{ f.}$  is equivalent to it, except that any names that occur in the definition of  $x$  are (recursively) replaced by their referents. Consequently, any subsequent change in these referents that might change the definition of  $x$  will not affect the definition of  $y$ .

If  $x$  is the name of any entity (that is, a pronoun, proverb, pro-adverb, or pro-conjunction), then ' $x$ ' f. is equivalent, but with all names in its definition recursively replaced by their referents.

$x \text{ f.}$  will not fix any part of  $x$  that contains  $\$$ .

For example:

```
sum=: +/
mean=: sum % #
norm=: - mean
norm a=: 2 3 4 5
_1.5 _0.5 0.5 1.5
```

```
N=: norm f.
N a
_1.5 _0.5 0.5 1.5
```

```
norm
- mean
```

```
N
- (+/ % #)
```

```
sum=: -/
norm a
2.5 3.5 4.5 5.5
```

```
N a
_1.5 _0.5 0.5 1.5
```

```
adv=: norm@
*: adv
```



norm@\*:

adv

norm@

'adv' f.

(- (-/ % #))@

'a' f.

2 3 4 5

## Hypergeometric

$m \ H. \ n \ 0 \ 0 \ 0$

The conjunction  $H.$  applies to two numeric lists to produce a monad which is the hypergeometric function defined in Section 15 of [Abramowitz and Stegun \[13\] p212](#); it is the limit of the dyadic case, whose left argument restricts the number of terms of the approximating series.

As discussed in [Iverson \[14\] p212](#), the conjunction is defined as follows:

```
rf=: 1 : '(,m.)"_ ^!.1/ i.@['          Rising factorial
L1=: 2 : 'm. rf %&(*) n. rf'
L2=: (i.@[ ^~ ])% (!@i.@[)
H =: 2 : '(m. L1 n. +/ . * L2) " 0'
```

For example:

```
'a b'=: 2 3 5; 6 5

a L1 b
(2 3 5"_ ^!.1/ i.@[) %&(*) 6 5"_ ^!.1/ i.@[

t=: 4 [ z=: 7

t a L1 b z
1 1 1.71429 4.28571

t (a H b , a H. b) z
295 295

8 (1 H. 1) i. 6
1 2.71825 7.38095 19.8464 51.8063 128.619
(1 H. 1) i. 6
1 2.71828 7.38906 20.0855 54.5982 148.413
^ i. 6
1 2.71828 7.38906 20.0855 54.5982 148.413

erf    =: 1 H. 1.5@*: * 2p_0.5&* % ^@:*:      error function
n01cdf=: -: @: >: @: erf @: ((%:0.5)&*)        CDF of normal 0,1
```

```
    erf 0.5 1 1.5
0.5205 0.842701 0.966105
    n01cdf _2 _1.5 _1 _0.5 0 0.5 1 1.5 2
0.0227501 0.0668072 0.158655 0.308538 0.5 0.691462 0.841345
0.933193 0.97725
```

## Integers

i. 1 \_ \_

## Index Of

The shape of `i.y` is `|y`, and its atoms are the first `*/|y` non-negative integers. A negative element in `y` causes reversal of the atoms along the corresponding axis. For example:

```
i. 5
0 1 2 3 4

i. 2 _5
4 3 2 1 0
9 8 7 6 5
```

If `rix` is the rank of an item of `x`, then the shape of the result of `x i. y` is `(-rix)}. $y`. Each atom of the result is either `#x` or the index of the first occurrence among the items of `x` of the corresponding `rix`-cell of `y`.

The comparison in `x i. y` is tolerant, and *fit* can be used to specify the tolerance, as in `i. !. t.`

```
(i.4);(i._4);(i.2 3 4);(i.2 _3 4);(i.'')
+-----+-----+-----+-----+
0 1 2 3	3 2 1 0	0 1 2 3	8 9 10 11	0
		4 5 6 7	4 5 6 7	
		8 9 10 11	0 1 2 3	
		12 13 14 15	20 21 22 23	
		16 17 18 19	16 17 18 19	
		20 21 22 23	12 13 14 15	
+-----+-----+-----+-----+
```

```
A=: 'abcdefghijklmnopqrstuvwxyz'
(A i. 'Now');(A i. 'now');(A {~ A i. 'now')
+-----+-----+-----+
| 26 14 22 | 13 14 22 | now |
+-----+-----+-----+
```

```
m=: 5 4 $ 12{. A
m;(m i. 'efgh');(1{m);(4{m)
+-----+-----+-----+
|abcd|1|efgh|efgh|
+-----+-----+-----+
```

|      |  |  |  |
|------|--|--|--|
| efgh |  |  |  |
| ijkl |  |  |  |
| abcd |  |  |  |
| efgh |  |  |  |

+-----+-----+-----+

**Integers****i:** 0 \_ \_**Index Of Last**

**i:** *y* is the list of integers from *-y* to *y*; in general, **i:** *a j. b* produces the list of numbers from *-a* to *a* in *b* equal steps (or in  $1+2*a$  steps if *b* is 0). Thus:

```

i: 3
_3 _2 _1 0 1 2 3

```

```

i: _2
2 1 0 _1 _2

```

```

i: _2.5j4
2.5 1.25 0 _1.25 _2.5

```

```

i: 0
0

```

**i:** is like **i.** but gives the index of the *last* occurrence. Thus:

```

1 2 3 4 1 2 3 i: 1 2 2 3 3 3
4 4 5
4 5 5 6 6 6 3 3 7

```

```

1 2 3 4 1 2 3 i. 1 2 2 3 3 3
4 4 5
0 1 1 2 2 2 3 3 7

```

For example:

```

(3 # i.3 4) i: (i.2 4)
2 5

```

```

(3 # i.3 4) i. (i.2 4)
0 3

```

```

([ #~ i.@# = i.~) 'eleemosynary' first occurrence of each letter
elmosynar

```

```

([ #~ i.@# = i:~) 'eleemosynary' last occurrence of each letter
lemosnary

```

```

([ #~ i.~ = i:~) 'eleemosynary' letters that are unique
lmosnar

```

**Imaginary****j. 0 0 0****Complex**`j. y ↔ 0j1 * y``x j. y ↔ x + j. y`

For example:

`j. 4`  
`0j4``3 j. 4`  
`3j4`

```

a=: i. 3 3
a:(j. 2*a):(a j. 2*a)
+-----+-----+-----+
0 1 2	0 0j2 0j4	0 1j2 2j4
3 4 5	0j6 0j8 0j10	3j6 4j8 5j10
6 7 8	0j12 0j14 0j16	6j12 7j14 8j16
+-----+-----+-----+

```

```

(+ a j. 2*a):(|a j. 2*a)
+-----+-----+
0 1j_2 2j_4	0 2.23607 4.47214
3j_6 4j_8 5j_10	6.7082 8.94427 11.1803
6j_12 7j_14 8j_16	13.4164 15.6525 17.8885
+-----+-----+

```

```

1 2 3 j./ 4 5 6 7
1j4 1j5 1j6 1j7
2j4 2j5 2j6 2j7
3j4 3j5 3j6 3j7

```

```

j./?. 2 3 4$1000
131j34 755j53 458j529 532j671
218j7 47j383 678j66 679j417
934j686 383j588 519j930 830j846

```

A table of random complex numbers

## Level Of

L. \_\_\_\_\_

If  $y$  is open or empty,  $L.y$  is 0; if it is boxed and non-empty, the level is one plus the maximum of the levels of the opened elements.

For example:

```

]y=: (<2 3 4),(<(5 6 ; <<i. 2 3)
+-----+-----+
+-----+	+---+-----+				
	2 3 4		5 6	+-----+	
+-----+			0 1 2		
				3 4 5	
			+-----+		
		+---+-----+			
+-----+-----+

```

L. y  
3

$$L_2 = L_3 = 0$$



## Level At

$u \ L: \ n \ \_ \_ \_$

The conjunction  $L:$  applies the verb  $u$  at the levels specified by  $n$ . The right argument  $n$  behaves like that of the rank conjunction in several respects:

- It may have three elements, specifying levels for the monadic, left, and right cases
- The full form of the levels used is  $3\$&.\ | \ .n$ . For example, a 2-element list  $p,q$  is equivalent to  $q,p,q$ .
- Negative values are complementary:  $u \ L:(-r) \ y \leftrightarrow u \ L:(0>.(L.y)-r) \ y$

For example:

```
] y=: (<2 3 4),<(5 6 ; <<i. 2 3)
```

|                 |  |  |  |         |  |  |  |         |  |  |  |
|-----------------|--|--|--|---------|--|--|--|---------|--|--|--|
| +-----+-----+   |  |  |  |         |  |  |  |         |  |  |  |
| +-----+ +-----+ |  |  |  |         |  |  |  |         |  |  |  |
| 2 3 4     5 6   |  |  |  | +-----+ |  |  |  |         |  |  |  |
| +-----+         |  |  |  |         |  |  |  | 0 1 2   |  |  |  |
|                 |  |  |  |         |  |  |  | 3 4 5   |  |  |  |
|                 |  |  |  |         |  |  |  | +-----+ |  |  |  |
|                 |  |  |  | +-----+ |  |  |  |         |  |  |  |
| +-----+-----+   |  |  |  |         |  |  |  |         |  |  |  |

$+: L: 0 \ y$

The adverb  $L:0$  may be called *leaf*

|                 |  |  |  |         |  |  |  |         |  |  |  |
|-----------------|--|--|--|---------|--|--|--|---------|--|--|--|
| +-----+-----+   |  |  |  |         |  |  |  |         |  |  |  |
| +-----+ +-----+ |  |  |  |         |  |  |  |         |  |  |  |
| 4 6 8     10 12 |  |  |  | +-----+ |  |  |  |         |  |  |  |
| +-----+         |  |  |  |         |  |  |  | 0 2 4   |  |  |  |
|                 |  |  |  |         |  |  |  | 6 8 10  |  |  |  |
|                 |  |  |  |         |  |  |  | +-----+ |  |  |  |
|                 |  |  |  | +-----+ |  |  |  |         |  |  |  |
| +-----+-----+   |  |  |  |         |  |  |  |         |  |  |  |

$2 \# L: 0 \ y$

|               |  |  |  |  |  |  |  |  |  |  |  |
|---------------|--|--|--|--|--|--|--|--|--|--|--|
| +-----+-----+ |  |  |  |  |  |  |  |  |  |  |  |
| +-----+-----+ |  |  |  |  |  |  |  |  |  |  |  |

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 6 | + | - | - | - | - | + |
| + |   |   |   |   |   |   |   |   |   | 0 | 1 | 2 |   |   |   |
|   |   |   |   |   |   |   |   |   |   | 0 | 1 | 2 |   |   |   |
|   |   |   |   |   |   |   |   |   |   | 3 | 4 | 5 |   |   |   |
|   |   |   |   |   |   |   |   |   |   | 3 | 4 | 5 |   |   |   |
|   |   |   |   |   |   |   |   |   |   | + |   |   |   |   |   |
| + |   |   |   |   |   | + |   |   |   |   |   |   |   |   |   |

2 # L: 1 y

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| + |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| + |   |   |   |   | + |   |   |   |   |   |   |   |   |   |   |
| 2 | 3 | 4 | 2 | 3 | 4 | 5 | 5 | 6 | 6 | + | - | - | - | - | + |
| + |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   | 3 |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   | + |   |   |   |   |   |   |   |   |   |   |
| + |   |   |   |   | + |   |   |   |   |   |   |   |   |   |   |

## Explicit Arguments

`m. n.`

The name `m.` denotes a *noun* left argument in an explicitly-defined adverb or conjunction, and the name `n.` denotes a *noun* right argument in an explicitly-defined conjunction. See the names `u.` `v.` `x.` `y.` and Explicit Definition (`:`).

For example:

```

pow=: 2 : 0
  i=.0
  t=.]
  while. n.>i do.
    i=.1+i
    t=.u.@t f.
  end.
)

o. pow 2
o.@(o.@])

o. pow 3
o.@(o.@(o.@])

o. pow 3 x=: 1
31.0063

o.^:3 x
31.0063

o. pow +
|value error: n.
|      n.>i

```

Uses `n.` (noun right argument)

Uses `u.` (verb left argument)

**Comment**

NB.

The rest of the line following NB. is ignored.

For example:

```

      text=: 'i. 3 4 NB. 3-by-4 table'
      ;: text
+---+---+-----+
|i.|3 4|NB. 3-by-4 table|
+---+---+-----+

      ". text
0 1  2  3
4 5  6  7
8 9 10 12

```

Execute text

**Pi Times**

o. 0 0 0

**Circle Function**

o.  $y$  yields  $\pi$  times  $y$ . Thus o. 1 is approximately 3.14159.

The function  $x \& o.$  is even or odd as  $x$  is even or odd;  $(-x) \& o.$  is its inverse.

| <b>x o. y</b>      | <b>x</b>  | <b>(-x) o. y</b>     |                                                                          |
|--------------------|-----------|----------------------|--------------------------------------------------------------------------|
| Sqrt 1-(Sqr y)     | <b>0</b>  | Sqrt 1-(Sqr y)       | $0 \& o. \text{.} @ (1 \& o. \text{.}) \leftrightarrow 2 \& o. \text{.}$ |
| Sine y             | <b>1</b>  | Arcsine y            | Radian arg/result                                                        |
| Cosine y           | <b>2</b>  | Arccos y             | "                                                                        |
| Tangent y          | <b>3</b>  | Arctan y             | "                                                                        |
| Sqrt (Sqr y)+1     | <b>4</b>  | Sqrt (Sqr y)-1       | $4 \& o. \text{.} @ (5 \& o. \text{.}) \leftrightarrow 6 \& o. \text{.}$ |
| Sinh y             | <b>5</b>  | Arcsinh y            | Sinh is hyperbolic sine                                                  |
| Cosh y             | <b>6</b>  | Arccosh y            |                                                                          |
| Tanh y             | <b>7</b>  | Arctanh y            |                                                                          |
| Sqrt - (1 + Sqr y) | <b>8</b>  | - Sqrt - (1 + Sqr y) |                                                                          |
| RealPart y         | <b>9</b>  | y                    |                                                                          |
| Magnitude y        | <b>10</b> | Conjugate (+y)       |                                                                          |
| ImaginaryPart y    | <b>11</b> | j. y                 |                                                                          |
| AngleOf y          | <b>12</b> | ^j. y                |                                                                          |

**Examples:**

```

rfd=: %&180 @ o.           Radians from degrees
sin=: 1&o.
SIN=: sin@rfd
(rfd 0 90 180);(sin 0 1.5708);(SIN 0 90)
+-----+-----+
|0 1.5708 3.14159|0 1|0 1|
+-----+-----+

```

The principal domain of the inverse of a non-monotonic function is restricted. The

limits on real domains of arcsine and arccos may be obtained as follows (as multiples of  $\pi$ ):

$$\frac{1}{-0.5} \frac{p_1}{0.5} * \frac{-1}{1} \frac{-2}{0} \circ / \frac{-1}{1}$$

**Roots**

p. 1 1 0

**Polynomial**

p. c ↔ (m;r)  
 p.p.c ↔ c

If  $e$  is a vector whose elements are all non-negative integers, then  $p.<c,.e$  gives the coefficients of the equivalent polynomial:

(p. <c,.e)&p.  
 ↔ (<c,.e)&p.

There are three cases -- coefficients; multiplier and roots; multinomial (boxed matrix of coefficients and exponents):

c p. x ↔ +/c\*x^i.#c  
 (m;r) p. x ↔ m \* \*/x-r  
 (<r)&p. ↔ (1;r)&p.  
 (<c,.e)p.<y ↔ c+/ .\*e\*/  
 .(^~)y

where  $m$  is a scalar;  $c$  and  $r$  are scalars or vectors; and  $e$  is a vector or matrix such that  $(\$(e)-: (\#c), (\#y))$ . A scalar  $y$  is extended normally.

p. 1 0 0 1  
 +-----+  
 |1|\_1 0.5j0.866025 0.5j\_0.866025|  
 +-----+

]mr=: p. c=: 0 16 \_12 2 Multiplier/Roots from Coefficients

+-----+  
 |2|4 2 0|  
 +-----+

x=: 0 1 2 3 4 5  
 (c p. x), ((<c,.i.4)p. x), (mr p. x),: 2\*(x-4)\*(x-2)\*(x-0)  
 0 6 0 \_6 0 30  
 0 6 0 \_6 0 30  
 0 6 0 \_6 0 30  
 0 6 0 \_6 0 30

c=: 1 3 3 1  
 c p. x  
 1 8 27 64 125 216

```
(x+1)^3
1 8 27 64 125 216
```

```
bc=: !~/~i.5                      Binomial coefficients
bc;(bc p./ x);((i.5) ^~/ x+1)
```

|         |   |   |   |   |  |         |    |    |     |     |      |
|---------|---|---|---|---|--|---------|----|----|-----|-----|------|
| +-----+ |   |   |   |   |  | +-----+ |    |    |     |     |      |
| 1       | 0 | 0 | 0 | 0 |  | 1       | 1  | 1  | 1   | 1   | 1    |
| 1       | 1 | 0 | 0 | 0 |  | 1       | 2  | 3  | 4   | 5   | 6    |
| 1       | 2 | 1 | 0 | 0 |  | 1       | 4  | 9  | 16  | 25  | 36   |
| 1       | 3 | 3 | 1 | 0 |  | 1       | 8  | 27 | 64  | 125 | 216  |
| 1       | 4 | 6 | 4 | 1 |  | 1       | 16 | 81 | 256 | 625 | 1296 |
| +-----+ |   |   |   |   |  | +-----+ |    |    |     |     |      |

```
c&p. d. 1 x                      First derivative of polynomial
3 12 27 48 75 108
```

```
(<1 _1 ,. 5 0) p. 3              Coefficients / Exponents
242
```

```
_1 0 0 0 0 1 p. 3
242
```

```
p. <1 _1 ,. 5 0                  Coefficients / Exponents to Coefficients
_1 0 0 0 0 1
```

```
c=: _1 1 2 3 [ e=: 4 2$2 1 1 1 1 2 0 2
c,.e                               Coefficients / Exponents
_1 2 1
1 1 1
2 1 2
3 0 2
```

```
(<c,.e) p. <y=:2.5 _1             Multinomial
11.75
```

```
c +/ .* e */ .(^~) y
11.75
```

Note that  $(<c,.e)p.<y$  is a "proper" multinomial only if the elements of  $e$  are all non-negative integers. In general the powers are not so limited, as in the weighted sum of square root and 4-th root:

```
] t=: <2 3,.1r2 1r4
+-----+
| 2 1r2|
```



$$\frac{|3 - 1r4|}{+-----+}$$

$$\frac{(t \cdot p \cdot 16), +/ 2 \cdot 3 \cdot 16 \cdot 1r2 \cdot 1r4}{14 \cdot 14}$$

The variant  $p \cdot ! \cdot s$  is a *stope* polynomial; it differs from  $p \cdot$  in that its definition is based upon the  $stope \cdot ! \cdot s$  instead of on  $\cdot ^$  (power).

## Polynomial Derivative

`p.. 1 0 1`

## Polynomial Integral

Applied to a polynomial (coefficients or boxed roots), `p..` produces the coefficients of the derivative of polynomial. For example:

```
p.. 1 2 3 4 5
2 6 12 20
p.. 5 4 3 2 1
4 6 6 4
```

```
p.. 2; 1j1 1j_1
_4 4
p.. p. 2; 1j1 1j_1
_4 4
```

`x p.. y` produces the integral of polynomial `y` with a constant term of `x`. Thus:

```
5 p.. 4 6 6 4
5 4 3 2 1
1 p.. 2 6 12 20
1 2 3 4 5
```

Further examples:

```
p.. 1&o. t. i. 11x          derivative of sine
1 0 _1r2 0 1r24 0 _1r720 0 1r40320 0
2&o. t. i.10x              cosine
1 0 _1r2 0 1r24 0 _1r720 0 1r40320 0
```

```
p.. 2&o. t. i. 11x          derivative of cosine
0 _1 0 1r6 0 _1r120 0 1r5040 0 _1r362880
-@(1&o.) t. i.10x          minus sine
0 _1 0 1r6 0 _1r120 0 1r5040 0 _1r362880
- (1&o. t. i.10x)
0 _1 0 1r6 0 _1r120 0 1r5040 0 _1r362880
```

```
p..^(i.@#) 8 $ 1
1 1 1 1 1 1 1 1
1 2 3 4 5 6 7 0
2 6 12 20 30 42 0 0
6 24 60 120 210 0 0 0
24 120 360 840 0 0 0 0
120 720 2520 0 0 0 0 0
```

|      |      |   |   |   |   |   |   |
|------|------|---|---|---|---|---|---|
| 720  | 5040 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5040 | 0    | 0 | 0 | 0 | 0 | 0 | 0 |

## Primes

$p:$  0

The result of  $p: i$  is the  $i$ -th prime.  
For example:

```
p: 0
2
```

```
p: i. 15
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

The inverse of  $p:$  is the number of primes less than the argument, often denoted by  $\pi(n)$ :

```
pi=: p: ^: _1
pi i. 15
0 0 0 1 2 2 3 3 4 4 4 4 5 5 6
```

```
(| , pi ,: p:@pi) i.15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
0 0 0 1 2 2 3 3 4 4 4 4 5 5 6
2 2 2 3 5 5 7 7 11 11 11 11 13 13 17
```

```
y=: (2^31)-1
```

```
la=: pi y
105097564
```

```
lb=: p: a
2147483647
```

```
b=y
1
```

Prime Factors

q: 0 0 0

Prime Exponents

q: y is the list of prime factors of a positive integer argument y. For example:

```
y=: 105600
q: y
2 2 2 2 2 2 2 3 5 5 11
*/q: y
105600
$ q: 1
0
*/q: 1
1
q: b. _1
*/
```

If x is positive and finite, x q: y is the list of exponents in the prime decomposition of positive integer y, based upon the first x primes; if x is \_ , a sufficient number of primes is used.

If x is negative and finite, x q: y is a 2-row table of the last |x| primes and exponents in the prime factorization of y; if x is \_\_ , a sufficient number of primes is used. For example:

```
2 q: 700
2 0
10 q: 700
2 0 2 1 0 0 0 0 0 0
_ q: 700
2 0 2 1
```

```
~.@q: 700
2 5 7

+/"1@=@q: 700
2 2 1

_ q: 700
2 5 7
2 2 1

_ q: !20x
2 3 5 7 11 13 17 19
18 8 4 2 1 1 1 1
```

Distinct prime factors

Exponents in prime factorization

|                                                                                                                                                           |                                                                              |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| <pre> y=: 100 e=: _&amp;q: (e ; +:&amp;.e ; -:&amp;.e ; %&amp;3&amp;.e)y +-----+-----+---+-----+   2 0 2 10000 10 4.64159  +-----+-----+---+-----+ </pre> | <p>Completed list of exponents</p> <p>Exponents, square, sqrt, cube root</p> |
| <pre> V=: /@,: </pre>                                                                                                                                     | <p>For vectors of disparate lengths</p>                                      |
| <pre> 12 (+V&amp;.e; -V&amp;.e; &gt;.V&amp;.e; &lt;.V&amp;.e) y +-----+-----+---+---+  1200 0.12 300 4  +-----+-----+---+---+ </pre>                      | <p>Product, quotient, LCM, GCD</p>                                           |
| <pre> totient=: * -.@%@~.&amp;.q: totient 700 240 </pre>                                                                                                  | <p>Euler's totient function</p>                                              |
| <pre> + / 1 = 700 +. i.700 240 </pre>                                                                                                                     |                                                                              |

**Angle** $r. \quad 0 \quad 0 \quad 0$ **Polar** $r.y \leftrightarrow ^j.y$  $x \ r. \ y \leftrightarrow x*r. \ y$ 

The result of  $r. \ y$  is a complex number of magnitude 1, whose real and imaginary parts are coordinates of the point on the unit circle at an angle of  $y$  radians. For example:

```
r. 2
_0.416147j0.909297
```

```
+. r. 2
_0.416147 0.909297
```

```
| r. 2
1
```

```
y=: 1r4 * o. i.7
```

Multiples of one-quarter  $\pi$ 

```
format=: 8j3&":
```

```
(format ,.y);(format +. r.y);(format +. 2 r.y)
```

```
+-----+-----+-----+
0.000	1.000 0.000	2.000 0.000
0.785	0.707 0.707	1.414 1.414
1.571	0.000 1.000	0.000 2.000
2.356	_0.707 0.707	_1.414 1.414
3.142	_1.000 0.000	_2.000 0.000
3.927	_0.707 _0.707	_1.414 _1.414
4.712	0.000 _1.000	0.000 _2.000
+-----+-----+-----+
```

```
3 r. _2
_1.24844j_2.72789
```

```
*. 3 r. _2
3 _2
```

## Symbol

`s:`    `— — —`

## Symbol

*Symbols* are a new data type and are created by the verb `s:.` Symbols provide a mechanism for searching, sorting, and comparisons more efficient than alternative mechanisms such as boxed strings. Existing structural, selection, and relational verbs are extended to work on symbols. Arithmetic verbs do not work on symbols.

The monad `s:` produces an array of symbols. Several types of arguments are acceptable:

- string with the leading character as the separator
- literal array where each row, excluding trailing blanks, is the name of a symbol
- array of boxed strings

`s:^:_1`, the inverse of `s:`, is `5&s:.`

The dyad `s:` takes a scalar integer left argument and computes a variety of functions:

| Left | Right | Function                                                              |
|------|-------|-----------------------------------------------------------------------|
| 0    | 0     | the cardinality of the set of symbols                                 |
| 0    | 1     | the string length (the number of characters used in the string table) |

0

2

the table of symbols; the columns are:

0 index in the string table

1 length in bytes



- 2 hash value
- 3 color
- 4 parent
- 5 left
- 6 right
- 7 order #
- 8 predecessor
- 9 successor
- 10 bit flags

The details of this data may change from one version of J to the next. 03 the string table 04 the hash table. \_1 indicates free entries; non-negative values are indices into the table of symbols. 05 the binary tree root 06 the binary tree fill factor 07 the binary tree gap 010 get the global symbols data, equivalent to 0 s:&. >i.8.

The details of this data may change from one version of J to the next. 011 perform an integrity check on the global symbols data 012 the number of queries required for each symbol 1array of symbols a string of the symbol names each prefaced by a leading ' ' \_1string the symbols list for a string containing symbol names each prefaced by the leading character 2array of symbols a string of the symbol names each suffixed by a trailing zero character \_2string the symbols list for a string containing symbol names each suffixed by the trailing character 3array of symbols a literal array of the symbol names padded with zero characters \_3literal array the symbols array for the literal array wherein each row, excluding trailing zero characters, is the name of a symbol 4array of symbols a literal array of the symbol names padded with blanks \_4literal array the symbols array for the literal array wherein each row, excluding trailing blanks, is the name of a symbol 5array of symbols an array of boxed strings of the symbol names \_5boxed strings the symbols array for the boxed array wherein each box is a string of a symbol name 6array of symbols an integer array of the symbol indices (indices into the table of symbols) \_6indices the symbols for the indices 7array of symbols an integer array of the order numbers for the symbols 10global symbols data set the global symbols data (as previously returned by 0 s: 10) after performing an integrity check on it. Incorrect global symbols data may cause misinterpretation of symbol arrays, or data corruption, or a system crash, or the end of civilization as we know it.

The inverse of  $k \& s:$  is  $(-k) \& s:$ , for non-zero integer  $k$  between \_6 and 6.

The remainder of this text is divided into the following sections: Display, Annotated Examples, Space and Time, and Persistence.

## Display

The display of a symbol is the character ``` (`96{a.}`) prefaced to the symbol name; the display of a symbol array is similar to that display of numeric arrays, except that columns are aligned on the *left*. See Annotated Examples below.

### Annotated Examples

```
] t=: s: ' zero one two three four five'
`zero `one `two `three `four `five
```

```
$ t
```

a list of 6 symbols

```
6
  3 5 $ t
```

a matrix of symbols

```
`zero `one `two `three `four
`five `zero `one `two `three
`four `five `zero `one `two
```

```
  1 3 5 3 1 { t
`one `three `five `three `one
|. t
`five `four `three `two `one `zero
_2 |. t
`four `five `zero `one `two `three
  1 0 2 0 4 0 # t
`zero `two `two `four `four `four `four
```

```
<"0 t
```

symbols can be boxed

```
+-----+
|`zero|`one|`two|`three|`four|`five|
+-----+
(2|i.#t) </. t
+-----+
|`zero `two `four|`one `three `five|
+-----+
```

```
<:/~ t
```

relations work on symbols

```
1 0 0 0 0 0
1 1 1 1 0 0
1 0 1 0 0 0
1 0 1 1 0 0
1 1 1 1 1 0
1 1 1 1 1 1
```

```
t + t
```

arithmetic functions *don't* work on symbols

```
|domain error
|  t    +t
```

```
/: t
5 4 1 3 2 0
```

symbols can be graded/sorted

```
5 s: t
+-----+-----+-----+-----+-----+
|zero|one|two|three|four|five|
+-----+-----+-----+-----+
  (/: t) -: /: 5 s: t
1
```

convert symbols to boxed strings

```
/:~ t
`five `four `one `three `two `zero
```

```
<:/~ /:~ t
1 1 1 1 1 1
0 1 1 1 1 1
0 0 1 1 1 1
0 0 0 1 1 1
0 0 0 0 1 1
0 0 0 0 0 1
```

```
t i. s: ' three one four one five nine'
3 1 4 1 5 6
t e.~ s: ' three one four one five nine'
1 1 1 1 1 0
```

```
10{. t
`zero `one `two `three `four `five ` ` ` `
the fill for symbols is the 0-length symbol
```

```
_10{.t
` ` ` ` `zero `one `two `three `four `five
```

```
0 s: 0
8
cardinality (current # of unique symbols)
```

```
a=: ;:'A AAPL AMAT AMD AMZN ATT BA CRA CSCO DELL F GE GM HWP
IBM INTC'
```

```
a=: a,;:'JDSU LLY LU MOT MSFT NOK NT PFE PG QCOM RMBS T XRX
YHOO'
```

```
b=: ;:'NY SF LDN TOK HK FF TOR'
```

```
c=: ;:'Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec'
```

```
d=: <;_1 ' 00 01 02 03 04 05 06 07 08 09'
```

```
e=: ;:'open high low close'
```

```

t=: }.@;.>{' ','&.>&.>a;b;c;d;i<e
$t
30 7 12 10 4
*/ $t
100800
2 4 ($,) t
+-----+-----+-----+-----+
-+
|A NY Jan 00 open|A NY Jan 00 high|A NY Jan 00 low|A NY Jan 00
close|
+-----+-----+-----+-----+
-+
|A NY Jan 01 open|A NY Jan 01 high|A NY Jan 01 low|A NY Jan 01
close|
+-----+-----+-----+-----+
-+
y=: s: t                                create a whole lot of symbols
$y
30 7 12 10 4
2 4 ($,) y
`A NY Jan 00 open `A NY Jan 00 high `A NY Jan 00 low `A NY Jan 00
close
`A NY Jan 01 open `A NY Jan 01 high `A NY Jan 01 low `A NY Jan 01
close

0 s: 11                                system integrity check
1
0 s: 0                                cardinality
100808

(+/% #) 0 s: 12                        mean # of queries per symbol
1.31213

h=: 100808 {. 2 {"1 ] 0 s: 2          hash values

(+/~: h) % #h                          fraction of distinct hash values
0.999821
(+/~: h |~ #0 s: 4) % #h              fraction with respect to hash table
0.831005

```

## Space and Time

In the current implementation, a symbol  $y$  requires 4 bytes for an index, 8 or more bytes in the hash table, 44 bytes in the table of symbols, and  $\text{len } y$  bytes (times 2 if Unicode) in the string table, where  $\text{len} =: \# \> @ (5 \& s: )$ , the length of the

symbol name. (A symbol requires a single 4-byte entry in the hash table, but for efficient hashing the system maintains at least  $2 \cdot n$  entries for  $n$  symbols.) Multiple occurrences of a symbol require just multiple indices; entries in the hash table, the table of symbols, and the string table are not duplicated.

Computations on symbols generally require linear time. Specifically:

|              |                                  |
|--------------|----------------------------------|
| query (new)  | $O((\text{len } y) * ^{0} s: 0)$ |
| query (old)  | $O(\text{len } y)$               |
| $/:y$        | $O(* / \$y)$                     |
| $i\{y$       | $O(( * / \$i) * * / \} . \$y)$   |
| $x < y$ etc. | $O(x > . \& ( * / @ \$) y)$      |
| $x i . y$    | $O(x + \& ( * / @ \$) y)$        |

## Persistence

The interpretation of symbols depend on the global symbols data  $0 s: 10$ . For this interpretation to persist across J sessions the global symbols data must be restored at the beginning of a session. Thus:

|                                      |                                    |
|--------------------------------------|------------------------------------|
| $((3!:1) 0 s: 10) 1!:2 < 'symb.dat'$ | to store the global symbols data   |
| $10 s: (3!:2) 1!:1 < 'symb.dat'$     | to restore the global symbols data |

See the cautionary statements under  $10 s: x$ .

Spread

u S: n \_ \_ \_

u S: n produces the list resulting from applying u to the argument(s) at levels n (which has the same interpretation as the right argument of L:). For example, #0:S:0 y is the number of leaves (level 0 arrays) in y.

For example:

```
fetch=: >@({&>/)@(<"0@|.@[ , <@]) " 1 _
] t=: 5!:2 <'fetch'                                An array with an interesting structure
```

|  |  |  |  |  |  |  |  |  |  |       |  |
|--|--|--|--|--|--|--|--|--|--|-------|--|
|  |  |  |  |  |  |  |  |  |  | " 1 _ |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |
|  |  |  |  |  |  |  |  |  |  |       |  |

```
+---+-----+
| / | 0 0 2 1   |
+---+-----+
| @ | 0 1       |
+---+-----+
| < | 0 2 0 0 0 0 |
+---+-----+
| " | 0 2 0 0 0 1 |
+---+-----+
| 0 | 0 2 0 0 0 2 |
+---+-----+
| @ | 0 2 0 0 1   |
+---+-----+
| | . | 0 2 0 0 2   |
+---+-----+
```

## Assign Taylor

$m \ t. \ 0 \ 0 \ 0$

The function  $h =: u \backslash v \ t.$  is equivalent to  $u$  except that the function produced by  $h \ t.$  is  $v$ . For example, an explicit definition (such as the definition of `exp` below) has no associated Taylor series, but one can be assigned.

```

y=: i. 5
^y
1 2.71828 7.38906 20.0855 54.5982

^ t. y
1 1 0.5 0.1666667 0.04166667

exp=: 3 : '^ y.'
exp y
1 2.71828 7.38906 20.0855 54.5982

exp t. y
|domain error
|      exp t.y

e=: exp`(%!) t.
e y
1 2.71828 7.38906 20.0855 54.5982

e t. y
1 1 0.5 0.1666667 0.04166667

%@e t. y
1 _1 0.5 _0.1666667 0.04166667

%@^ t. y
1 _1 0.5 _0.1666667 0.04166667

```



## Taylor Coefficient

$u\ t.\ 0\ 0\ 0$

$u\ t.\ y$  is the  $y$ th coefficient in the Taylor series approximation to the function  $u$ . The domain of the adverb  $t$ . is the same as the left domain of the derivative  $D$ . See the case  $m\ t$ .

$x\ u\ t.y$  is the product of  $(x^y)$  and  $u\ t.\ y$ .

For example:

```
f=: 1 2 1&p.
g=: 1 3 3 1&p.
x=: 10%~i=: i.8
]c=: (f*g) t. i
1 5 10 10 5 1 0 0

6.2 ":(c p. x),:(f*g) x
1.00 1.61 2.49 3.71 5.38 7.59 10.49 14.20
1.00 1.61 2.49 3.71 5.38 7.59 10.49 14.20

(c p. x)=(f*g) x
1 1 1 1 1 1 1 1

]d=: f@g t. i
4 12 21 22 15 6 1 0

(d p. x)=(f g x)
1 1 1 1 1 1 1 1

sin=: 1&o.
cos=: 2&o.
8.4":t=: (^ t. i),(sin t. i),:(cos t. i)
1.0000 1.0000 0.5000 0.1667 0.0417 0.0083 0.0014 0.0002
0.0000 1.0000 0.0000 _0.1667 0.0000 0.0083 0.0000 _0.0002
1.0000 0.0000 _0.5000 0.0000 0.0417 0.0000 _0.0014 0.0000

* t
1 1 1 1 1 1 1 1
0 1 0 _1 0 1 0 _1
1 0 _1 0 1 0 _1 0
```

```
((sin*sin)+(cos*cos)) t. i  
1 0 0 0 _2.71051e_20 0 0 0
```

```
rf=: n%d  
n=: 0 1&p.  
d=: 1 _1 _1&p.  
]fibonacci=: rf t. i. 20  
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

```
2 +/\ fibonacci  
1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
```

```
(% -. - *: ) t. i.20  
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

## Weighted Taylor

$u\ t: \quad 0\ 0\ 0$

The result of  $u\ t: k$  is  $(!k)*u\ t.\ k$ . In other words, the coefficients produced by  $t:$  are the Taylor coefficients *weighted* by the factorial. As a consequence, the coefficients produced by it when applied to functions of the exponential family show simple patterns. For this reason it is sometimes called the *exponential generating function*.

For example:

```

k=: i. 12
^ t: k
1 1 1 1 1 1 1 1 1 1 1 1 1

%t: k
1 _1 1 _1 1 _1 1 _1 1 _1 1 _1
Decaying exponential

sin =: 1&o.
cos =: 2&o.
sinh=: 5&o.
cosh=: 6&o.
exp=: ^
dec=: %t:

(exp t:,dec t:,sinh t:,cosh t:,sin t:,:cos t:) k
1 1 1 1 1 1 1 1 1 1 1 1
1 _1 1 _1 1 _1 1 _1 1 _1 1 _1
0 1 0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0 1 0
0 1 0 _1 0 1 0 _1 0 1 0 _1
1 0 _1 0 1 0 _1 0 1 0 _1 0

```

## Taylor Approximation

$u \approx T_n$

$u \approx T_n$  is the  $n$ -term Taylor approximation to the function  $u$ .

For example:

```

6.2 " : ^ T. 8 x=: 2 %~ i.8
1.00 1.65 2.72 4.48 7.38 12.13 19.85 32.23

6.2 " : ^ x
1.00 1.65 2.72 4.48 7.39 12.18 20.09 33.12

^ T. _
3 : 0"0
g=:p.&y.@:((^) t.)@i.
g +: ^:(g ~: g@+:)^:_ ] 1
)
(^ = ^T._) i. 5
1 1 1 1 1

```

Compare the *conjunction*  $T.$  with the *adverb*  $t.$

## Explicit Arguments

`u.`    `v.`

The name `u.` denotes a *verb* left argument in an explicitly-defined adverb or conjunction, and the name `v.` denotes a *verb* right argument in an explicitly-defined conjunction. See the names `m.` `n.` `x.` `y.` and Explicit Definition (`:`).

For example:

```
pow=: 2 : 0
i=.0
t=.]
while. n.>i do.
  i=.1+i
  t=.u.@t f.
end.
)
```

Uses `n.` (noun right argument)

Uses `u.` (verb left argument)

```
o. pow 2
o.@(o.@])
o. pow 3
o.@(o.@(o.@]))
```

```
o. pow 3 x=: 1
31.0063
o.^:3 x
31.0063
```

```
o. pow +
|value error: n.
|      n.>i
```

Unicode

u:    —   —   —

Unicode

Unicode or 2-byte characters are a new datatype. Unicode arrays are created by the verb `u:` . Existing verbs are extended to work on unicodes.

The monad `u:` applies to several kinds of arguments:

| Argument          | Result                        |
|-------------------|-------------------------------|
| 1-byte characters | same as <code>2&amp;u:</code> |
| 2-byte characters | copy of argument              |
| integers          | same as <code>4&amp;u:</code> |

The inverse of the monad `u:` is `3&u:`

The dyad `u:` takes a scalar integer left argument and applies to several kinds of arguments:

| Left | Right             | Result                                                            |
|------|-------------------|-------------------------------------------------------------------|
| 1    | 2-byte characters | 1-byte characters; high order bytes are discarded                 |
| 2    | 1-byte characters | 2-byte characters; high order bytes are 0                         |
| 3    | 2-byte characters | integers                                                          |
| 4    | integers          | 2-byte characters; integers must be from 0 to 65535               |
| 5    | 2-byte characters | 1-byte characters; high order bytes must be 0 (and are discarded) |

|  |                                                                                                                                                                                                                           |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <div>6</div> <div>1-byte characters</div> <div>2-byte characters; pairs of 1-byte characters are converted to 2-byte characters</div> <div>1&amp;u: and 2&amp;u: is an inverse pair, as are 3&amp;u: and 4&amp;u: .</div> |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

2-byte characters can not be entered the keyboard. The display of an array `x` of 2-byte characters is that of `1 u: x`, that is, discarding the high-order byte of each 2-byte character.

### Examples:

```
] t=: u: 'We the people'
We the people
3!:0 t
131072
131072
```

the unicode datatype numeric code is

```
u: 97 98 99 +/ 0 256 512 1024
aaaa
bbbb
cccc
```

2-byte characters have the same display as 1-byte characters

```
'a' = u: 97 + 0 256 512 1024
1 0 0 0
```

```
] t=: (2 4$'abcdefgh') , u: 'wxyz'
abcd
efgh
wxyz
3!:0 t
131072
```

1- and 2-byte characters can be catenated together. The 1-byte characters are *promoted*.

## Explicit Arguments $x.$ $y.$

The names  $x.$  and  $y.$  denote the left and right arguments in an explicit definition. See the names  $m.$   $n.$   $u.$   $v.$  and Explicit Definition  $(:)$ .

For example:

```
info=: 3 : 0
(3!:0 y.);(#$y.);($y.);5!:5 <'y.' : (info x.) ,&< (info y.) ) info
i.12 +--+---+-----+ |4|1|12|i.12| +--+---+-----+ info o.i.3 4 +--+---
--+-+-----+-----+ |8|2|3 4|3.1415926535897931*i.3 4| +--+
+---+-----+-----+ 'x' info ;:'cogito, ergo sum.' +---
-----+-----+-----+ |+-+---+---+|+-+---+---+
-----+-----+-----+ | |2|0| |'x'| | |32|1|4|<;._1 ' cogito ,
ergo sum.' | | |+-+---+---+|+-+---+---+-----+ | +---
-----+-----+-----+
```



## Extended Precision      $x:$                        Num/Denom

$x:$  applies to real numbers and produces extended precision rational numbers. It applies to integers and produces extended integers. The implied comparison in  $x:$  is tolerant. The inverse  $x:^:_1$  converts rationals, including extended integers, into finite precision numbers (floating point or integer).

1  $x: y$  is the same as  $x: y$ ; and 2  $x: y$  produces the two extended integers of the numerator and denominator of the argument.

```
x: 1.2
6r5
```

```
2 x: 1.2
6 5
```

```
x: 1.2 _1.2 0 0.07
6r5 _6r5 0 7r100
```

```
x: 3j4
|domain error
| x:3j4
```

```
] pi =: o.1
3.14159
```

```
x: pi
1285290289249r409120605684
```

```
pi - 1285290289249%409120605684
1.49214e_13
```

```
x:!.0 pi
884279719003555r281474976710656
```

```
pi - 884279719003555%281474976710656
0
```

2 x: 1r2 3r4 5r6 \_7r8  
1 2  
3 4  
5 6  
\_7 8

See also [Section II G p69](#).

## Constant Functions

`_9: to 9: _ _ _`

The results are `_9` and `_8` and `_7` and so on to `_9`.

For example:

```
x=: 1 2 3 [ y=: 4 5 6
2: y
2

x 9: y
9
```

The rank conjunction `"` applies to any noun to make a constant function of specified rank. The particular constant functions illustrated above are therefore special cases of the form `i"_`. For example:

```
2"_ y
2
```

```
2"0 y
2 2 2
```

```
2"1 i. 2 3 4
2 2 2
2 2 2
```

```
2p1"0 y
6.28319 6.28319 6.28319
```

```
1p1 1p_1 1x1"0 y
3.14159 0.3183099 2.71828
3.14159 0.3183099 2.71828
3.14159 0.3183099 2.71828
```

A constant function with a vector result

```
0 0 0"1 y
0 0 0
```

The zero vector in a space of three dimensions

```
a=: 'abcdefghijklmnopqrstuvwxyz' "_
```

```
A=: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' "_  
s=: ' ' " _
```

```
f=: { a  
f 5 8 6
```

fig

```
g=: { a,:A  
g 1 5;0 8;0 6
```

Fig

# Constants

The form of a numeric constant defined and illustrated in [Part I p61](#) is elaborated by the use of further letters, as in  $2r3$  for two-thirds,  $2p1$  for two  $\pi$ , and  $2e3p1$  for  $2000\pi$ . The complete scheme of numeric constants obeys the following hierarchy:

- .
  - 
  - e
  - ad ar j
  - p x
  - b
- The decimal point is obeyed first
- The negative sign is obeyed next
- Exponential (scientific) notation
- Complex (magnitude and angle) in degrees or radians; Complex number
- Numbers based on **pi** ( $\circ.1$ ) and on Euler's number (the exponential  $^1$ )
- Base value (using a to z for 10 to 35)

Moreover, digits with a trailing  $x$  denote an extended precision integer, and digits followed by an  $r$  followed by further digits denote a rational number. See [Section II G p69](#).

For example,  $2.3$  denotes two and three-tenths and  $-2.3$  denotes its negation; but  $-2j3$  denotes a complex number with real part  $-2$  and imaginary part  $3$ , *not* the negation of the complex number  $2j3$ . Furthermore, symbols at the same level of the hierarchy cannot be used together:  $1p2x3$  is an ill-formed number.

The following lists illustrate the main points:

$2.3e2$   $2.3e-2$   $2j3$   
 $230$   $0.023$   $2j3$

$2p1$   $1p-1$   
 $6.28319$   $0.31831$

$1x2$   $2x1$   $1x-1$   
 $7.38906$   $5.43656$   $0.367879$

$2e2j-2e2$   $2e2j2p1$   $2ad45$   $2ar0.785398$   
 $200j-200$   $628.319j6.28319$   $1.41421j1.41421$   $1.41421j1.41421$

16b1f 10b23 \_10b23 1e2b23 2b111.111  
31 23 \_17 203 7.875

Negative integers following  $p$  and  $x$  indicate the use of reciprocals. For example,  $2p_{-2}$  is two divided by  $\pi$  squared, and  $2x_{-2}$  is two divided by the square of Euler's number.

# Control Structures

Control words are used in explicit definition (:), and are punctuation that determine the sequence of execution. Matching control words and the enclosed sentences make up a control structure. The following control words and structures are available:

[assert. p201](#) T

[break. p202](#)

[continue. p203](#)

[for. p204](#) T do. B end.  
for\_xyz. T do. B end.

[goto name. p205](#)  
label\_name.

[if. p206](#) T do. B end.

if. T do. B else. B1 end.

if. T do. B  
elseif. T1 do. B1  
elseif. T2 do. B2  
end.

[return. p207](#)

[select. p208](#) T  
case. T0 do. B0  
fcase. T1 do. B1  
case. T2 do. B2  
end.

[throw. p209](#)

[try. p210](#) B catch. B1 catchd. B2 catcht. B3 end.

[while. p211](#) T do. B end.  
whilst. T do. B end.

Words beginning with **B** or **T** denote blocks, comprising zero or more simple sentences and control structures. The last sentence executed in a **T** block is tested for a non-zero value in its leading atom, and determines the block to be executed next. (An empty **T** block result or an omitted **T** block tests true.) The final result is the result of the last sentence executed that was not in a **T** block, and if there is no such last executed sentence, the final result is `i.0 0`.

These control words and control structures are further detailed in the immediately following pages.



# assert.

```
assert. T
```

An assertion failure is signalled if the single sentence `T` is not an array of all 1s. Assertions are evaluated or not according to the flag controlled by [9!:34 p223](#) and [9!:35 p223](#) .

For example:

```
cfi=: 4 : 0 " 0          The y.-th combination of i.x
assert. 0<:y.
assert. y.=<.y.
assert. y.<2^x. v=. +/\(i.x.)!x. m=. (y.<v)i. 1 (m,x.) ci (y.-
m{0,v) ) ci=: 4 : 0 " 1 0 'm n'=. x. if. 0=m do. i.0 else. v=.
+/\ (m-1)!(1-m)}i.-n k=. (v>y.) i. 1
    k,(1+k)+(x.-1,1+k)ci(y.-k{0,v)
end.
)

    5 cfi 6
0 1

    5 cfi 6+i.10
0 1
0 2
0 3
0 4
1 2
1 3
1 4
2 3
2 4
3 4

    (i.100) -: 100x cfi <:2^100x
1

    5 cfi 33
```

```
|assertion failure: cfi
|      y.<2^x.
|      5 cfi 6.2
|assertion failure: cfi
|      y.=<.y.
|      5 cfi 'a'
|domain error: cfi
|      y.=      <.y.
```

# break.

`break.` may be used within a [for. p204](#), [while. p211](#), or [whilst. p211](#) control structure, and goes to the end of the innermost such enclosing structure. See also [continue. p203](#).

For example:

```
itn=: 3 : 0                                Inverse triangular number
s=.0
for_j.
  i.2e4
do.
  if. s>:y. do. j break. end.
  s=.j+s
end.
)

x=: 10 100 1000 10000
itn"0 x
5 15 46 142

]y=: 2&!^:_1 x
5 14.651 45.2242 141.922

2!y
10 100 1000 10000
```

# continue.

continue. may be used within a [for. p204](#), [while. p211](#), or [whilst. p211](#) control structure, and goes to the top of the innermost such enclosing structure. See also [break. p202](#).

For example:

```

sumeven=: 3 : 0
s=.0
for_j. i.y. do.
  if. 2|j do. continue. end.
  s=.j+s
end.
)

sumeven 9
20

+/ (* -.@(2&|)) i.9
20

sumeven 1000
249500

+/ (* -.@(2&|)) i.1000
249500

(sumeven = 2&!@>.&.-:) 1000
1

```

# for.

```
for.      T do. B end.
for_xyz. T do. B end.
```

The `B` block is evaluated once for each item of the array `A` that results from evaluating the `T` block. In the `for_xyz.` form, the local name `xyz` is set to the value of an item on each evaluation and `xyz_index` is set to the index of the item.

[break. p202](#) goes to the end of the `for.` control structure, and [continue. p203](#) goes to the evaluation of `B` for the next item.

For example:

```
f0=: 3 : 0
s=. 0
for. i. y. do. s=.:s end.
)
```

```
f1=: 3 : 0
s=.0
for_j. i.y. do.
  if. 2|j do. continue. end.
  s=.j+s
end.
)
```

```
comb=: 4 : 0          All size x. combinations of i.y.
z=.1 0$k=.i.#c=.1,~(y.-x.)$0
for. i.x. do. z=.:k,.&.>(-c=.:+/\c){.&.><1+z end. ) (f0 = ])"0
?5$100 1 1 1 1 1 (f1 = 2&!@>.&.-:)"0 ?5$100
1 1 1 1 1
```

```
3 comb 5
0 1 2
0 1 3
0 1 4
```

```

0 2 3
0 2 4
0 3 4
1 2 3
1 2 4
1 3 4
2 3 4

```

```

queens=: 3 : 0                                solves the n queens problem
z=.i.n,*n=.y.
for. }.z do.
  b=. -. (i.n) e."1 ,. z +"1 _ ((-i.){:z) */ _1 0 1
  z=. ((+/"1 b)#z),.(,b)#(*$b)$i.n
end.
)

```

```

      queens 5
0 2 4 1 3
0 3 1 4 2
1 3 0 2 4
1 4 2 0 3
2 0 3 1 4
2 4 1 3 0
3 0 2 4 1
3 1 4 2 0
4 1 3 0 2
4 2 0 3 1

```

## goto\_name.

goto\_name. goes to the matching label\_name. . These control words are included to facilitate modelling of certain processes.

For example:

```
f=: 3 : 0
if. y. do. goto_true. else. goto_false. end.
label_true. 'true' return.
label_false. 'false' return.
)

f 0
false

f 1
true

f ''
true
```

# if.

```
if. T do. B end.
if. T do. B else. B1 end.
if. T do. B elseif. T1 do. elseif. T2 do. B2 end.
```

The last sentence executed in a `T` block is tested for a non-zero value in its leading atom, determining whether the `B` block after the `do.` or the rest of the sentence is executed. An empty `T` block result or an omitted `T` block tests true.

See also the [select. p208](#) control structure.

For example:

```
plus=: 4 : 0                                Addition on non-negative integers
if. y. do. >: x. plus <: y. else. x. end.
)

    plus"0/~ i.5
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8

sel=: 1 : 'x. # ['

quicksort=: 3 : 0
if. 1 >: #y. do. y.
else.
    (quicksort y.<sel e),(y.=sel e),quicksort y.>sel e=.y.{~?#y.
end.
)

    quicksort 15 2 9 10 4 0 13 13 18 7
0 2 4 7 9 10 13 13 15 18

test=: 3 : 0
if.      0<y. do. 'positive'
```



```
elseif. 0>y. do. 'negative'
elseif.      do. 'zero'
end.
)
```

```
test&.> 5 _2.71828 0
+-----+-----+-----+
|positive|negative|zero|
+-----+-----+-----+
```

```
test&.> '' ; 0 1 _2 ; _3 9
+-----+-----+-----+
|positive|zero|negative|
+-----+-----+-----+
```

# return.

return. exits a verb, adverb, or conjunction.

The result of an explicit definition is the result of the last executed sentence that was not in a test block. If there is no such executed sentence, the result is `i. 0 0`.

For example:

```
f=: 3 : 0
try.
  if. 0<:y. do. 'positive' return. else. t=.'negative' end.
catch.
  t=.'caught'
end.
'it is ',t
)

f 7
positive

f _12
it is negative

f 'deipnosophist'
it is caught

(i.0 0) -: 3 : 'return.' 999
1

g=: 3 : 'if. 13=|y. do. ''triskaidekaphobia'' end.'

g 13
triskaidekaphobia

g 5
(i.0 0) -: g 5
1
```

# select.

```
select. T
case.  T0 do. B0
case.  T1 do. B1
fcase. T2 do. B2
case.  T3 do. B3
end.
```

The result  $R$  of  $T$  is compared to the elements of the result  $R_i$  of  $T_i$ , and the  $B_i$  block of the first `case.` or `fcase.` with a match is evaluated. Evaluation of the `select.` control structure then terminates for a `case.`, or continues with the next  $B(i+1)$  block for an `fcase.` (and further continues if *it* is an `fcase.`).

The comparison is  $R \text{ e.}\&\text{boxifopen } R_i$  where `boxifopen=:<^(L.=0:)`, and a case with an omitted  $T_i$  is considered to match.

For example:

```
f0=: 3 : 0
select. y.
  case. 1;2 do. 'one two'
  case. 3   do. 'three'
  case. 4;5 do. 'four five'
  case. 6   do. 666
end.
)

f0&.> 1 2 3 4 5 6
+-----+-----+-----+-----+-----+-----+
|one two|one two|three|four five|four five|666|
+-----+-----+-----+-----+-----+-----+

(i.0 0) -: f0 7
1

f1=: 3 : 0
```

```

select. y.
  case. 'a' do. i.1
  case. 'b' do. i.2
  case.      do. i.3
end.
)

f1&.> 'a' ; ('a';'b') ; 'b' ; 'x'
+--+-----+
|0|0|0 1|0 1 2|
+--+-----+

```

```

f2=: 3 : 0
t=. ''
select. y.
  case. 1 do. t=.t,'one '
  fcase. 2 do. t=.t,'two '
  case. 3 do. t=.t,'three '
  fcase. 4 do. t=.t,'four '
end.
)

```

```

f2 1
one

```

```

f2 2
two three

```

```

f2 3
three

```

```

f2 4
four

```

```

'' -: f2 5
1

```

# throw.

`throw.` cuts the stack back to the `catcht.` section of a `try.` and resumes execution there; a `throw.` executed outside of a `try.` with a `catcht.` causes a return to immediate execution.

For example:

```
main=: 3 : 0
try.
  sub y.
catcht.
  select. type_jthrow_
    case. 'aaaa' do. 'throw aaaa'
    case. 'bbb'  do. 'throw bbb'
    case. 'cc'   do. 'throw cc'
    case.       do. throw.    NB. handled by higher-level catcht.
  (if any)
  end.
end.
)
```

```
sub=: 3 : 0
if. y.<0 do. type_jthrow_=: 'aaaa' throw. end. if. y. <4 do.
type_jthrow_=: 'bbb' throw. end. if. y. <8 do. type_jthrow_=:
'cc' throw. end. (":y.),' not thrown' ) main _4 throw aaaa main 1
throw bbb main 5 throw cc main 88 88 not thrown
```

A `throw.` can communicate information back to a `catcht.` through the use of a global name in a locale, as illustrated in the examples above.

# try.

```
try. B0 catch. B1 catchd. B2 catcht. B3 end.
```

The try/catch control structure may contain one or more distinct occurrences of catch. catchd. catcht. , in any order. For example:

```
try. B0 catch. B1 end.
```

```
try. B0 catcht. B1 catchd. B2 end.
```

```
try. B0 catcht. B1 catch. B2 catchd. B3 end.
```

The B0 block is executed and:

- catch. catches errors, whatever the setting of the debug flag [13!:0 p225](#)
- catchd. catches errors, but only if the debug flag is 0
- catcht. catches a [throw. p209](#)

For example:

```
f=: 4 : 0
try.
  try. 3+y. catch. *:x. end.
catch.
  'x and y are both bad'
end.
)

13 f 7
10

13 f 'primogeniture'
169

'sui' f 'generis'
x and y are both bad
```

# while.

```
while.  T do. B end.
whilst. T do. B end.
```

The last sentence executed in the `T` block is tested for a non-zero value in its leading atom. If true, the `B` block is executed. The `T` block is then retested, and so on, continuing until the `T` block tests false. (An empty `T` block result or an omitted `T` block tests true.)

`whilst.` differs from `while.` only in that it skips test the first time.

[break. p202](#) goes to the end of the `while` or `whilst.` control structure and [continue. p203](#) goes to the top.

For example:

```
exp =: 4 : 0                                Exponentiation by repeated squaring
z=.1
a=.x.
n=.y.
while. n do.
  if. 2|n do. z=.z*a end.
  a=.*:a
  n=.<.-:n
end.
z
)
```

3 exp 7  
2187

3 ^ 7  
2187

1.1 exp 0  
1

2x exp 128  
340282366920938463463374607431768211456



# References

1. Falkoff, A.D., and K.E. Iverson, *The Design of APL*, IBM Journal of Research and Development, July, 1973.
2. Falkoff, A.D., and K.E. Iverson, *The Evolution of APL*, ACM Sigplan Notices, August 1978.
3. Iverson, K.E., *A Dictionary of APL*, ACM APL Quote-Quad, September, 1987.
4. McIntyre, D.B., *Language as an Intellectual Tool: From Hieroglyphics to APL*, IBM Systems Journal, December, 1991.
5. Iverson, K.E., *A Personal View of APL*, IBM Systems Journal, December, 1991.
6. Hui, R.K.W., *An Implementation of J*, ISI, 1992.
7. Reiter, C.A., *Fractals, Visualization, and J*, ISI, 1995.
8. Iverson, K.E., *Exploring Math*, ISI, 1996.
9. Burke, C. et al., *J Phrases*, ISI, 1998.
10. McDonnell, E.E., *Complex Floor*, APL Congress 73, North-Holland/American Elsevier.
11. McDonnell, E.E., *Zero Divided by Zero*, APL76, ACM.
12. Bernecky, Robert, and R.K.W. Hui, *Gerunds and Representations*, APL91, ACM.
13. Abramowitz, M., and I.A. Stegun, *Handbook of Mathematical Functions*, National Bureau of Standards Applied Mathematics Series #55, U. S. Government Printing Office, 1964.

14. Iverson, K.E., *Concrete Math Companion*, ISI, 1995.

## Acknowledgments

We are indebted to Eric Iverson for the design and implementation of locatives, control structures, and the Windows interface.

## Appendix A. Foreign Conjunction

The conjunction `!:` applies to integer scalar left and right arguments to produce verbs, with the exception that the case `5!:0` produces an adverb. These verbs behave like any other verb: they may be assigned names, may serve as arguments to adverbs and conjunctions, and must be used with an argument even though (as in `6!:0 ''`) it may have no significance. Where these verbs take names as arguments, the names are boxed, as in `4!:55 'a';'bc'` to erase the names `a` and `bc`. A bracketed left argument indicates that it is optional.

[0!:](#) [p215](#)Scripts  
[1!:](#) [p216](#)Files  
[2!:](#) [p217](#)Host  
[3!:](#) [p218](#)Conversions  
[4!:](#) [p219](#)Names  
[5!:](#) [p220](#)Representation  
[6!:](#) [p221](#)Time  
[7!:](#) [p222](#)Space  
[9!:](#) [p223](#)Global Parameters  
[11!:](#) [p224](#)Window Driver  
[13!:](#) [p225](#)Debug  
[15!:](#) [p226](#)Dynamic Link Library  
[18!:](#) [p227](#)Locales  
[128!:](#) [p228](#)Miscellaneous

## Scripts

**0! :**

0! :k y

The script *y* is executed according to the digits (zero or one) in the 3-digit decimal representation of *k* :

|   | <b>1st digit</b>  | <b>2nd digit</b>  | <b>3rd digit</b> |
|---|-------------------|-------------------|------------------|
| 0 | From file or noun | Stop on error     | Silent           |
| 1 | From noun         | Continue on error | Display          |

For example, 0!:111 *abc* executes the *noun abc*, *completes*, and *displays*.

Sessions begin with (silent, stop) execution of  
0!:0<'profile.ijs'

0! :2 y

The script *y* is expected to be a sequence of tautologies;  
0! :2 *y* is like 0! :1 *y* but stops if any result is other than  
all 1 .

0! :3 y

Like 0! :2 *y* , but produces a 1 or 0 result according to  
whether the script passes (contains only tautologies) or  
fails.

## Files

**1! :**

Except as otherwise noted, a file may be specified by a name (such as `<'sub\abc.q'`) or by an integer file number obtained from `open` (`1! : 21` `<'sub\abc.q'`).

`1! : 0 y`

**Directory.** `y` is a string of the path search expression (a boxed string is also accepted); the result is a 5-column table of the file name, modification time, size, permission, and attributes, individually boxed. For example, try `1! : 0` `'*. *'`. The permission and attribute columns are system dependent. For example, in Windows:

```
1! : 0 'j.exe'
+-----+-----+-----+-----+-----+
|j.exe|1998 2 2 14 33 46|676864|rwx|-----a|
+-----+-----+-----+-----+-----+
```

Permission is a 3-letter string indicating the read, write, and execute permissions. Attributes is a 6-letter string indicating read-only, hidden, system, volume label, directory, and archive.

`1! : 1 y`

**Read.** `y` is a file name or a file number (produced by `1! : 21`); the result is a string of the file contents., e.g. `1! : 1` `<'abc.q'`. The following values for `y` are also permitted:

- 1 read from the keyboard (does not work within a script)
- 3 (Unix only) read from standard input (`stdin`)

`x 1! : 2 y`

**Write.** `x` is a string of the new contents of the file; `y` is a file name or file number (produced by `1! : 21`). The following values for `y` are also permitted:

- 2 screen output.
- 4 (Unix only) standard output (`stdout`)
- 5 (Unix only) standard error (`stderr`)

|                         |                                                                                                                                                                                                                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x 1!:3 y</code>   | <b>Append.</b> Like <code>x 1!:2 y</code> , but appends rather than replaces                                                                                                                                                                                                          |
| <code>1!:4 y</code>     | <b>Size</b>                                                                                                                                                                                                                                                                           |
| <code>1!:5 y</code>     | <b>Create Directory.</b> <code>y</code> is a (boxed) directory name                                                                                                                                                                                                                   |
| <code>[x] 1!:6 y</code> | <b>Query/Set Attributes.</b>                                                                                                                                                                                                                                                          |
| <code>[x] 1!:7 y</code> | <b>Query/Set Permissions.</b>                                                                                                                                                                                                                                                         |
| <code>1!:11 y</code>    | <b>Indexed Read.</b> <code>y</code> is a list of a boxed file name (or number) and a boxed index and length. The index may be negative. If the length is elided, the read goes to the end. For example:<br><br><pre>1!:11 'abc.x';1000 20 f=: 1!:21 &lt;'abc.x' 1!:11 f,1000 20</pre> |
| <code>x 1!:12 y</code>  | <b>Indexed Write.</b> <code>x</code> is the string to be written; <code>y</code> is a list of a boxed file name (or number) and a boxed index.                                                                                                                                        |
| <code>1!:20 y</code>    | <b>File Numbers and Names.</b> A 2-column table of the open file numbers and names.                                                                                                                                                                                                   |
| <code>1!:21 y</code>    | <b>Open.</b> Open file named <code>y</code> , creating it if necessary; result is a file number.                                                                                                                                                                                      |
| <code>1!:22 y</code>    | <b>Close.</b> Close file named or numbered <code>y</code> . Any locks are released.                                                                                                                                                                                                   |
| <code>1!:30 y</code>    | <b>Locks.</b> A 3-column integer table of the file number, index, and length of file locks. The argument <code>y</code> is required but ignored.                                                                                                                                      |
| <code>1!:31 y</code>    | <b>Lock.</b> <code>y</code> is a 3-element integer vector of the file number, index, and length of the file region to be locked; the result is 1 if the request succeeded, and 0 if it did not.                                                                                       |
| <code>1!:32 y</code>    | <b>Unlock.</b> <code>y</code> is a 3-element integer vector of the file number, index, and length of the file region to be unlocked.                                                                                                                                                  |
| <code>1!:43 y</code>    | <b>Query Current Working Directory.</b> Query the current working directory (Posix <code>getcwd</code> ).                                                                                                                                                                             |
| <code>1!:44 y</code>    | <b>Set Current Working Directory.</b> Set the current working directory (Posix <code>chdir</code> ).                                                                                                                                                                                  |

1!:45 y

**Profile.** The default profile.

In Windows, 1!:45 returns `profile.ijs` in the path of the J Front End (`j.exe` or `jconsole.exe`). For example:  
`c:\j501a\profile.ijs`.

In Unix, 1!:45 uses environment variables and the J version to determine the default profile:

- The version is the text in 9!:14 '' up to the first /. If 9!:14 '' returned `j501a/2002-07-05/17:50`, then the version is `j501a`.
- If `HOME/version/profile.ijs` exists, then it is the default profile. For example, if `HOME` was `/home/eric`, then `/home/eric/j501a/profile.ijs` would be the default profile if it existed.
- If that file doesn't exist, environment variable `JPATHversion` (for example, `JPATHj501a`), if it is defined, is the default profile.

1!:55 y

**Erase File/Directory.** e.g., 1!:55<'careful'



**Host****2! :**

- 2!:0** *y* **Host.** The list *y* is executed by the host system, and the result is returned. For example, `2!:0 'dir *.exe'`. Not available for Windows or Macintosh.
- 2!:1** *y* **Spawn.** Like `2!:0`, but yields `' '` without waiting for the host to finish. Any output is ignored. For example, `2!:1` can be used to invoke a text-editor. Not available for Windows or Macintosh.
- 2!:2** *y* **Host IO.** (Unix only.) The host command line *y* is passed to `/bin/sh` for processing, connecting two file numbers to the command's standard input and output. The result is a 3-element list of the process id of the task started and the file numbers associated with its standard input and output. These file numbers also appear in the result of `1!:20`. In this case, instead of appearing with a name they appear with the command line, prefixed by `>` (standard input) or `<` (standard output). The files associated with the process should be closed with `1!:22` when no longer in use. See also `2!:3` for a verb to wait for processes to complete.
- 2!:3** *y* **Wait.** (Unix only.) Wait for process id *y* to terminate. The result is the status code returned by the process.
- 2!:5** *y* **Getenv.** The value of the shell environment variable named *y*. If the named variable is undefined, the result is 0.
- 2!:55** *y* **Terminate Session.** *y* is an integer return code

Conversions

3!::

3!:0 y      **Type.** The internal type of the noun y, encoded as follows:

|     |                  |        |                       |
|-----|------------------|--------|-----------------------|
| 1   | boolean          | 1024   | sparse boolean        |
| 2   | literal          | 2048   | sparse literal        |
| 4   | integer          | 4096   | sparse integer        |
| 8   | floating point   | 8192   | sparse floating point |
| 16  | complex          | 16384  | sparse complex        |
| 32  | boxed            | 32768  | sparse boxed          |
| 64  | extended integer | 65536  | symbol                |
| 128 | rational         | 131072 | unicode               |

[x] 3!:1 y **Convert to Binary Representation.** In *standard byte order*, the bytes of a word are listed from most significant to least significant; in *reverse byte order*, the bytes are listed from least significant to most significant. For example, the 4-byte integer 265358979 is 0fd10e83 in standard byte-order and 830ed10f in reverse byte-order. The PC is a reverse byte order machine.

The dyad x 3!:1 y applies to an array y and produces its binary representation, according to the atom x:

| x  | <u>word size</u> | <u>byte order</u> |
|----|------------------|-------------------|
| 00 | 32 bits          | standard          |
| 01 | 32 bits          | reverse           |
| 10 | 64 bits          | standard          |
| 11 | 64 bits          | reverse           |

The monad 3!:1 produces the binary representation in the word size and byte order of the current machine. 3!:2 y**Convert from Binary/Hex Representation.** Inverse of 3!:1 and of 3!:3; works on an argument in either word size and in

either byte order. [x] 3!:3 y **Hex Representation.** Like 3!:1 , but the result is a literal matrix of the hexadecimal representation. For example, under Windows:

```
(3!:3 x); 3!:3 x,lp1 [ x=: 1 2 3 0 _1
+-----+-----+
04000000	08000000
00000000	00000000
05000000	06000000
01000000	01000000
05000000	06000000
01000000	00000000
02000000	0000f03f
03000000	00000000
00000000	00000040
ffffffff	00000000
	00000840
	00000000
	00000000
	00000000
	0000f0bf
	182d4454
	fb210940
+-----+-----+
```

3!:4 y

3!:5 y **Integer/Floating Conversion.** If ic=: 3!:4 and fc=: 3!:5 , then

```
2 ic y   J integers to binary long integers
_2 ic y  binary long integers to J integers
1 ic y   J integers to binary short integers
_1 ic y  binary short integers to J integers
0 ic y   binary unsigned short integers to J integers
2 fc y   J floats to binary doubles
_2 fc y  binary doubles to J floats
1 fc y   J floats to binary short floats
_1 fc y  binary short floats to J floats
```

All ranks are infinite and all inverses of k&ic and k&fc exist. 3!:6 y **Lock Script.** Converts plain script text into locked script text.

## Names

**4! :**

4!:0 y

**Name Class.** Class of (boxed) name:

|    |             |
|----|-------------|
| _2 | invalid     |
| _1 | unused      |
| 0  | noun        |
| 1  | adverb      |
| 2  | conjunction |
| 3  | verb        |
| 6  | locale      |

[x] 4!:1 y

**Name List.** Result is a vector of boxed names belonging to the classes 0 to 3 and 6, as defined under 4!:0 above. The optional left argument specifies the initial letters of names to be included.

4!:3 y

**Scripts.** List of script names that have been invoked using 0!:

4!:4 y

**Script Index.** Index (in 4!:3 ' ') of the script that defined y, or \_1 if y was not defined from a script.

4!:5 y

**Names Changed.** 4!:5]0 turns off data collection; 4!:5]1 turns it on and produces a list of global names assigned since the last execution of 4!:5.

4!:55 y

**Erase.**

4!:56 y

**Erase All.** Erase all names and all locales.

## Representation

**5!:**

$x \ 5!:0$

**Define.**  $5!:0$  is an adverb and provides a complete inverse of  $5!:1$ . That is,  $(5!:1 < 'f')$   $5!:0$  equals  $f$  for all  $f$ .

$5!:1 \ y$

**Atomic.** The atomic representation of the entity named  $y$  and is used in gerunds. The result is a single box containing a character list of the symbol (if primitive) or a two-element boxed list of the symbol and atomic representation of the arguments (if not primitive). "Symbol-less" entities are assigned the following encodings:

- 0 Noun
- 2 Hook
- 3 Fork
- 4 Bonded conjunction
- 7 Defined operator (pro-adverb or pro-conjunction)

For example:

```

plus=: +
5!:1 <'plus'
+-+
|+|
+-+
noun=: 3 1 4 1 5 9
5!:1 <'noun'
+-----+
|+-+-----+|
||0|3 1 4 1 5 9||
|+-+-----+|
+-----+
increment=: 1&+
5!:1 <'increment'
+-----+
|+-+-----+|
||&|+-----+|
```

```

				+-+--+	+					
					0	1				
				+-+--+						
				+-----+--+						
+-+-----+-----+										
+-----+

```

5!:2 y

**Boxed.**

```

nub=: (i.@# = i.~) # ]
5!:2 <'nub'

+-----+-----+-----+
+-----+-----+-----+	#	]								
	+-+--+--+	=+-+--+--+								
		i.	@	#			i.	~		
	+-+--+--+	+-+--+--+								
+-----+-----+-----+										
+-----+-----+-----+

```

5!:4 y

**Tree.** A literal matrix that represents the named entity in tree form. Thus:

```

5!:4 <'nub'
      +- i.
    +- @ -+- #
  +----+- =
  |    +- ~ --- i.
--+- #
+- ]

```

5!:5 y

**Linear.** The linear representation is a string which, when interpreted, produces the named object. For example:

```

5!:5 <'nub'
(i.@# = i.~) # ]

5!:5 <'a' [ a=: o. i. 3 4
3.14159265358979324*i.3 4

lr=: 3 : '5!:5 <'y.'''
lr 10000$x'
10000$x'

```

**Paren.** Like the linear representation, but is fully parenthesized.

**Explicit.** The left argument is 1 (monadic) or 2 (dyadic); the right argument is the boxed name of a verb, adverb, or conjunction. For example:

The result of `5! : 7` is a 3-column boxed matrix. Column 0 are the boxed integers `0 1 2 ... n-1`. Column 1 are boxed 3-element integer vectors of control information: control word code, goto line number, and source line number. Column 2 are boxed control words and sentences.

For all but the noun case, the default displays established by `9! : 3` provide convenient experimentation with all representations. For example, `9! : 3 (6 4 2)`

specifies that the paren, tree, and boxed representations are all to be displayed.



**Time****6! :****6!:0** *y***Current.** The current time in order YMDHMS (with fractional seconds):

```

6!:0 ''
1998 3 10 23 21 14.468

```

**6!:1** *y***Session.** Seconds since start of session**[x] 6!:2** *y***Execute.** Seconds to execute sentence *y* (mean of *x* times with default once). For example:

```

a=:?50 50$100
6!:2 '%.a'
0.091
10 (6!:2) '%.a'           Mean time of 10 executions
0.0771
ts=: 6!:2 , 7!:2@]        Time and space
ts '%.a'
0.08 369920

```

**6!:3** *y***Delay.** Delay execution for *y* seconds. For example, **6!:3** (2.5)**6!:8** *y***Query Clock Frequency.** Return the clock frequency, the number of clock counter values per second. (In Windows, the result of `QueryPerformanceFrequency`.) The argument *y* is ignored.

```

6!:8 ''
1.19318e6

```

6!:9 y

**Query Clock Counter.** Return the current clock counter (elapsed time) value. (In Windows, the result of QueryPerformanceCounter.) The argument y is ignored.

```
6!:9 ''
4.08486e9
_10 [\ 2 --~/\ 6!:9"1 (101 0)$0
9 7 7 6 7 5 7 6 6 7
6 7 6 6 6 6 8 6 6 7
6 6 7 6 6 7 6 6 6 6
6 7 6 6 6 6 7 6 6 8
6 7 6 6 6 7 6 6 6 7
6 6 6 7 7 6 7 6 7 6
6 6 6 7 6 6 6 6 7 6
6 6 6 7 6 7 7 6 7 6
6 6 6 6 7 6 6 6 7 6
6 6 6 6 6 6 6 6 6 8
```

6!:10 y

**Performance Monitor Data Area.** Literal (character) vector y is to hold the fixed-sized records of "packed" performance monitor data; the old PM data area (if any) is released. The PM counter (see 6!:12) is set to 0. If y is an empty vector, no PM data will be collected.

x is a 2-element vector of control parameters; 0 0 is assumed if it is elided.

```
0{x - what to record
    0 - entry and exit
    1 - entry and exit and all lines
1{x - action on running out of space in y
    0 - "wrap", discard the oldest record
    1 - "truncate", discard the newest record
```

The result is the number of records that will be held in y.

6!:11 y

**Unpack PM Data.** Unpack the PM data, resulting in a boxed vector of data columns:

- 0 name
- 1 locale where name is found
- 2 valence
- 3 line number (\_1 for entry; \_2 for exit)
- 4 space in use (bytes)
- 5 time (seconds)
- 6 list of names

Columns 0 and 1 are indices into column 6. Columns 0 to 5 have the same number of (corresponding) elements.

y are indices of the rows to be unpacked, where 0 is the oldest row (and \_1 is the newest row). All rows are selected if y is the empty vector. Rows in the result are always arranged from oldest to newest and no row is selected more than once.

6!:12 y

**Add y to the PM Counter.** PM data is recorded if the PM counter is greater than 0 and a PM data area is extant. (A non-empty PM data area must be specified before the PM counter can be changed.) The result is the value of the PM counter after the operation.

6!:13 y

**PM Statistics.** Produces a 6-element vector of the following information:

- 0 - what to record
  - 0 - entry and exit
  - 1 - entry and exit and all lines
- 1 - action on insufficient space
  - 0 - wrap
  - 1 - truncate
- 2 - the maximum number of records
- 3 - current number of records
- 4 - whether records have been lost
- 5 - current value of the PM counter

If no PM data area has been specified the result is 6\$0.  
The argument `y` is ignored.

---

### Further Examples:

```

sum=: +/

avg=: 3 : 0
n=. #y.
s=. sum y.
s % n
)

1 (6!:10) 1e5$'x'
3570
(6!:12) 1
1
avg"1 ? 3 10$100
65 64.4 64.3
(6!:12) _1
0

x=. 6!:11 ''
$&.> x
+---+---+---+---+---+---+---+
|21|21|21|21|21|21|3|
+---+---+---+---+---+---+---+
|: > 6{. x
0 2 1 _1 1152 22582.5
0 2 1 0 4224 22582.5
0 2 1 1 576 22582.5
1 2 1 _1 128 22582.5
1 2 1 _2 64 22582.5
0 2 1 2 128 22582.5
0 2 1 _2 0 22582.5
0 2 1 _1 128 22582.5
0 2 1 0 4224 22582.5
0 2 1 1 576 22582.5
1 2 1 _1 128 22582.5
1 2 1 _2 64 22582.5
0 2 1 2 128 22582.5
0 2 1 _2 0 22582.5
0 2 1 _1 0 22582.5

```

```

0 2 1 0 4224 22582.5
0 2 1 1 576 22582.5
1 2 1 _1 128 22582.5
1 2 1 _2 64 22582.5
0 2 1 2 128 22582.5
0 2 1 _2 0 22582.5

```

```

    _5[\ 2 - ~/\ > 5{x
4.52571e_5 2.09524e_5 1.50857e_5 1.67619e_5 9.21905e_6
6.28571e_5 9.21905e_6 1.59238e_5 1.59238e_5 1.34095e_5
1.00571e_5 1.00571e_5 5.61524e_5 7.54286e_6 1.59238e_5
1.50857e_5 1.42476e_5 1.00571e_5 8.38095e_6 5.61524e_5

```

```

> {: x
+---+---+---+
|avg|sum|base|
+---+---+---+

```

```

(6!:10) 1e5$'x' NB. entry and exit only
3570
(6!:12) 1
1
avg"1 ? 3 10$100
38.8 61.9 61.7
(6!:12) _1
0

```

```

x=. 6!:11 ''
|: > 6{.x
0 2 1 _1 832 22681.5
1 2 1 _1 4928 22681.5
1 2 1 _2 64 22681.5
0 2 1 _2 128 22681.5
0 2 1 _1 128 22681.5
1 2 1 _1 4928 22681.5
1 2 1 _2 64 22681.5
0 2 1 _2 128 22681.5
0 2 1 _1 0 22681.5
1 2 1 _1 4928 22681.5
1 2 1 _2 64 22681.5
0 2 1 _2 128 22681.5

```

```

    _5[\ 2 - ~/\ > 5{x
7.12381e_5 1.50857e_5 6.70476e_5 9.21905e_6 3.52e_5
1.00571e_5 0.000108952 7.54286e_6 3.93905e_5 1.08952e_5

```

6.03428e\_5                    0                    0                    0                    0

6!:13 ''  
0 0 3570 12 0 0

sum\_ab\_ =: +/

sum\_z\_ =: sum\_ab\_

avg\_q\_ =: 3 : 0  
tally=. #  
s=. sum y.  
n=. tally y.  
s % n  
)

(6!:10) 1e5\$('x'  
3570  
(6!:12) 1  
1  
avg\_q\_ i.12  
5.5

(6!:12) \_1  
0

x=: 6!:11 ''  
(>{&>/0 6{x) ,. ' ' ,. (>{&>/1 6{x) ,. ' ' ,. ": ,. >3{x  
avg\_q\_ q \_1  
sum q \_1  
sum\_ab\_ ab \_1  
sum\_ab\_ ab \_2  
sum q \_2  
tally q \_1  
tally q \_2  
avg\_q\_ q \_2

> 6{x  
+-----+-----+-----+-----+  
|avg\_q\_|sum|sum\_ab\_|tally|q|ab|  
+-----+-----+-----+-----+

## Space

**7!:**

7!:0 y

**Current.** Space currently in use.

7!:1 y

**Session.** Total space used since start of session.

7!:2 y

**Execute.** Space required to execute sentence y. For example:

```
7!:2 ' ,/x' [ x=: 3000 2 5$'kerygmatic'
33888
```

7!:3 y

**Free Space.** Information on the state of the J memory manager, currently a 2-column table of the block sizes and number of free blocks for each size. The definition of the result of 7!:3 , as well as the availability of 7!:3 itself, are subject to change. For example:

```
7!:3 ''
32 950
64 1135
128 554
256 157
512 74
1024 40
```

7!:5 y

**Space.** The space in bytes needed by the named objects y. For example:

```
Hilbert=: % @: >: @: (+/~) @: i.

7!:5 <'hilbert' 896 7!:5
<'name_without_value' |value error:
name_without_value | 7!:5
<'name_without_value' sp=: 3 : '7!:5 <'y.'''
sp 'chiaroscuro' 64 sp i.1000 4096 sp o.i.1000
8192 x=: o.y=: i.1000 7!:5 ;:'x y hilbert'
8192 4096 896
```

See also [9!:20 p223](#) and [9!:21 p223](#).

## Global Parameters

9! :

9! : (2\*n) queries a parameter and (if available) 9! : (1+2\*n) sets it.

|                       |                                                                                                                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9! :0 y               | <b>Random Seed.</b> Queries and sets the random seed used in pseudo-random number generation in the verb ? (Roll • Deal). The initial value is 7^5.                                  |
| 9! :1 y               |                                                                                                                                                                                      |
| 9! :2 y               | <b>Default Displays.</b> The representation(s) to used for default displays of non-nouns. The representations are as defined in 5! :n: 1 atomic, 2 boxed, 4 tree, 5 linear, 6 paren. |
| 9! :3 y               |                                                                                                                                                                                      |
| 9! :6 y               | <b>Box-Drawing Characters.</b> The eleven characters used to draw boxes (initially ++++++++   -).                                                                                    |
| 9! :7 y               |                                                                                                                                                                                      |
| 9! :8 y               | <b>Error Messages.</b> e.g. replace English messages (default) by French.                                                                                                            |
| 9! :9 y               |                                                                                                                                                                                      |
| 9! :10 y              | <b>Print Precision.</b> The print precision in default output (initially 6). The precision in particular cases can be set using fit, thus: " : ! .p                                  |
| 9! :11 y              |                                                                                                                                                                                      |
| 9! :12 y              | <b>System Type.</b>                                                                                                                                                                  |
| 0 PC                  | 5 Unix                                                                                                                                                                               |
| 1 PC386               | 6 Windows32 (95/98/2000/NT)                                                                                                                                                          |
| 2 Windows             | 7 Windows CE                                                                                                                                                                         |
| 3 Macintosh           | _1 Other                                                                                                                                                                             |
| 4 OS2                 |                                                                                                                                                                                      |
| 9! :14 y              | <b>J Version.</b> For example:                                                                                                                                                       |
| 9! :14 ' '            |                                                                                                                                                                                      |
| 4.01/1998-03-15/10:24 |                                                                                                                                                                                      |
| 9! :16 y              |                                                                                                                                                                                      |
| 9! :17 y              | <b>Boxed Display Positioning.</b> y is r,c specifying row and column positioning: 0 (top, left); 1 (centre); 2 (bottom, right)                                                       |
| 9! :18 y              |                                                                                                                                                                                      |



9!:19 y**Comparison Tolerance**. Queries and sets the comparison tolerance. See Equal (=). The tolerance in particular cases can be set using fit, thus: =! . t .

9!:20 y

9!:21 y**Memory Limit**. An upper bound on the size of any one memory allocation. The memory limit is initially infinity. 9!:24 y

9!:25 y**Security Level**. The security level is either 0 or 1. It is initially 0, and may be set to 1 (and can not be reset to 0). When the security level is 1, executing Window driver commands and certain foreigners (!:) that can alter the external state cause a "security violation" error to be signalled. The following foreigners are prohibited: dyads 0! : n, 1! : n except 1! : 40, 1! : 41, and 1! : 42, 2! : n, 8! : n, 14! : n, 15! : n, and 16! : n. 9!:26 y

9!:27 y**Immex Phrase**. See 9!:28 and 9!:29 . 9!:28 y

9!:29 y**Immex Flag**. If the immediate execution flag is 1, then on entry into immediate execution, the immediate execution phrase (9!:27) is executed and the flag is set to 0. 9!:32 y

9!:33 y**Execution Time Limit**. The execution time limit is a single non-negative (possibly non-integral) number of seconds. The limit is reduced for every line of immediate execution that exceeds a minimum granularity, and execution is interrupted with a "time limit error" if a non-zero limit is set and goes to 0. Current limitations:

- The resolution is milliseconds.
- Under Windows the limit is on elapsed time while J is running, instead of the task processor time as is intended.
- Under Windows the granularity is approximately 1.5 seconds.

Examples:

```

9!:32 ''                                query execution time limit; none is set
0
9!:33 ]5.25                             set limit to 5.25 seconds
9!:32 ''                                current setting
5.25

# %. ? 100 100 $ 1e6                    below granularity
100
9!:32 ''                                limit unchanged
5.25

".10#,: '# %. ?(2#100)$1e6'           above granularity
100 100 100 100 100 100 100 100 100 100

```

```

9!:32 '' limit reduced
2.226

```

```

".10#,: '# %. ?(2#100)$1e6' limit exceeded
|time limit
| # %.?100 100$1000000
9!:32 ''
0

```

```
9!:34 y
```

9!:35 y**Assertions.** 1 if and only if assertions are to be evaluated, with a default setting of 1. See the [assert. p201](#) control word. 9!:36 y

9!:37 y**Output Control.** A 4-element vector that controls session manager output:

end-of-line sequence 0 line feed; 1 carriage return; 2 carriage return line feed  
maximum line length Output lines are truncated at this length and "..." appended.  
maximum line before If the total number of output lines exceeds the sum of  
"maximum lines before" b and "maximum lines after" a ,  
then the first b lines are output, followed by a line of "...",  
followed by the last a lines.  
maximum line after See above.

The default for the end-of-line sequence is 0 for Unix, 1 for Macintosh, and 2 for others (including Windows); the defaults for the other output controls are 256 0 222 . 9!:38 y

9!:39 y**Locale Hash Table Size.** A 2-element vector that controls the default hash table sizes for named and numbered locales, with a default of 3 2 . A specified size of i indicates a table requiring approximately the same space as  $i \cdot 2^{6+i}$  . A larger hash table size improves performance; regardless of the hash table size, a locale may contain an effectively unlimited number of names.

The following table lists the initial hash table sizes:

| Category                   | Index | # Entries |
|----------------------------|-------|-----------|
| named locales              | 3     | 499       |
| numbered locales           | 2     | 241       |
| table of all named locales | 3     | 499       |
| explicit definition locals | 1     | 113       |

|      |   |      |
|------|---|------|
| base | 5 | 2029 |
| z    | 7 | 8179 |

The hash table size of a locale can be individually specified when the locale is created, using the dyad [18!:3 p227](#) . 9!:40 y

9!:41 y **Retain Comments and White Space**. Specifies whether comments and non-essential white space are retained in explicit definitions. The default is 1 (*retain* comments and white space). A setting of 1 can result in explicit definitions, even commentless ones, requiring double the amount of storage.

## Window Driver

**11! :**

$$11! : 0 \quad y$$

**Window Driver.** See help files.

## Debug

**13! :**

See [Section II.J p72](#) and the script `system\main\debug.ijs`.

The conjunction

`13!:16`

controls tracing.

`u 13!:16 n`  
`(r,c) 13!:16 n`

The right argument

`n`

specifies the maximum level of function call to be traced:

`0`

means no trace;

`1`

means immediate execution only;

`—`

means trace everything; etc. The left argument can be a verb to be used for displaying arrays in the trace (and is not itself traced during tracing). It may also be integers

`r,c`

,

whence the system default display is used, clipped to

`r`

rows and

`c`

columns. (Two numbers suffice as clipping parameters because the output of an  $n$ -dimensional array is 2-dimensional on the screen.) Finally, it may be the empty vector, whence the current trace level and display controls are shown. The result is

`i.0 0`

.

For example:

|                                      |                                                           |
|--------------------------------------|-----------------------------------------------------------|
| <code>trace=: 13!:16</code>          |                                                           |
| <code>lr =: 3 : '5!:5&lt;'y.'</code> | Linear display of an array                                |
| <code>_ _ trace _</code>             | Trace everything; display everything                      |
| <code>9 trace 1</code>               | Trace immediate execution only; display maximum of 9 rows |
| <code>" : trace n</code>             | Same as <code>_ _ trace n</code>                          |
| <code>lr trace n</code>              | Linear display of trace output                            |

Tracing provides information on results

*within*

a line; the action labels

0 monad

,

1 monad

,

9 paren

,

etc., are from the parse table in

[Section II.E p67](#)

, and reflect the activities of the interpreter with high fidelity.

|                                               |                                      |
|-----------------------------------------------|--------------------------------------|
| <code>_ _ (13!:16) _</code>                   | Trace everything; display everything |
| <code>i.4</code>                              | input sentence                       |
| <code>----- 0 monad -----</code>              | action                               |
| <code>i.</code>                               | verb                                 |
| <code>4</code>                                | argument                             |
| <code>0 1 2 3</code>                          | result                               |
| <code>=====</code>                            | end of parse                         |
| <code>0 1 2 3</code>                          | result of input sentence             |
| <br>                                          |                                      |
| <code>(i.2 4) +/ .* *: 5 * 10 20 30 40</code> | input sentence                       |
| <code>----- 2 dyad -----</code>               |                                      |
| <code>5</code>                                | left argument                        |
| <code>*</code>                                | verb                                 |
| <code>10 20 30 40</code>                      | right argument                       |
| <code>50 100 150 200</code>                   | result                               |

|                        |                                      |
|------------------------|--------------------------------------|
| ----- 0 monad -----    |                                      |
| i.                     | verb                                 |
| 2 4                    | argument                             |
| 0 1 2 3                | result                               |
| 4 5 6 7                |                                      |
| ----- 8 paren -----    |                                      |
| 0 1 2 3                |                                      |
| 4 5 6 7                |                                      |
| ----- 3 adverb -----   |                                      |
| +                      |                                      |
| /                      |                                      |
| +/                     |                                      |
| ----- 4 conj -----     |                                      |
| +/                     |                                      |
| .                      |                                      |
| *                      |                                      |
| +/ .*                  |                                      |
| ----- 1 monad -----    |                                      |
| *:                     | Words to the left of *: are parsed   |
| 50 100 150 200         | first because an operator may "grab" |
| 2500 10000 22500 40000 | a verb.                              |
| ----- 2 dyad -----     |                                      |
| 0 1 2 3                | left argument                        |
| 4 5 6 7                |                                      |
| +/ .*                  | verb                                 |
| 2500 10000 22500 40000 | right argument                       |
| 175000 475000          | result                               |
| =====                  | end of parse                         |
| 175000 475000          | result of input sentence             |

13!:0 y      **Reset.** Reset stack and disable (0) or enable (1) suspension. Nearly all the facilities in the 13!: family require that suspension be enabled; all the examples below assume that suspension is enabled: 13!:0 ]1 . Only named definitions (verb, adverb, or conjunction) can be suspended.

13!:1 y      **Display Stack.** Only named definitions (verb, adverb, or conjunction) are put on the stack. See also 13!:13 and 13!:18 .

13!:2 y      **Query Stops.**

13!:3 y

**Set Stops.** Explicit stops are requested by name and line number in the argument *y*, which contains zero or more stop specifications separated by semicolons. Each stop specification indicates a name, line numbers (if any) for the monadic case, a colon, and line numbers (if any) for the dyadic case. An asterisk indicates "all", and a tilde indicates "except for". For example:

|                          |                                                                                   |
|--------------------------|-----------------------------------------------------------------------------------|
| 13!:3 'f 0'              | f monadic line 0                                                                  |
| 13!:3 'f :2'             | f dyadic line 2                                                                   |
| 13!:3 'f 0 2:1'          | f monadic 0 2, dyadic 1                                                           |
| 13!:3 'f 0; g :*'        | f monadic 0 and g all dyadic                                                      |
| 13!:3 '* 0:0'            | monadic 0 and dyadic 0                                                            |
| 13!:3 'a* **;* ~ab* **;' | All monadic and dyadic whose names begin with a, except for any beginning with ab |

```
f=: 3 : 0
10
11
:
20
)
13!:3 'f 1:0'
1, dyad line 0
f ''
|stop: f
|      11
|f[1]
13!:0 ]1
suspension
3 f 4
|stop: f
|      20
|f[:0]
```

Stop at f monad line 1, dyad line 0

Clear stack and enable suspension



13!:4 y

**Run Again.** Resume execution at the current line. For example:

```
g=: 3 : ('t=. 2*y.'; '1+t')
3 4,g 'abc'
|domain error: g
| t=.2      *y.
|g[0]
    y.
indicates suspension
abc
    y.=. 25
y.
    13!:4 ''
3 4 51
```

six-space indent  
Local value of y.  
Redefine local value of  
Run again

13!:5 y

**Run Next.** Resume execution at the next line. For example:

```
h=: 3 : ('t=. 2 3*y.'; '1+t')
3 4,h 5 6 7
|length error: h
| t=.2 3      *y.
|h[0]
    t=. 99
indicates suspension
    13!:5 ''
3 4 100
```

six-space indent  
Run next

13!:6 y

**Exit and Return.** Exit the verb/adverb/conjunction at the top of the stack, returning result y. For example:

```
g=: 3 : ('t=. 2*y.'; '1+t')
3 4,g 'abc'
|domain error: g
| t=.2      *y.
|g[0]
    13!:6 [9
3 4 9
    h=: 2&*
    3 4,h 'abc'
|domain error: h
|h[0]
    13!:6 [97
3 4 97
```

Exit g with result 9  
Exit h with result 97

13!:7 y

**Continue.** Resume execution at line number y

[x] 13!:8 y

**Signal.** Signal error number {.,y (an integer between 1 and 255) with optional text x

[x] 13!:9 y

**Rerun.** Resume execution by rerunning the tacit verb at the top of the stack with the specified arguments. Thus:

```
plus=: +
plus/'abc'
|domain error: plus
|plus[:0] *
13!:13 ''
```

See below re

interpretation of stack

```
+-----+---+---+---+---+---+---+---+---+
plus	3	0	3	+	+---+		*		
						b	c		
					+---+				
+-----+---+---+---+---+---+---+---+---+
```

```
2 (13!:9) 3
```

Rerun, getting another

error

```
|domain error: plus
|plus[:0] !
13!:13 ''
```

Note left and right args

('a' and 5)

```
+-----+---+---+---+---+---+---+---+---+
plus	3	0	3	+	+---+		*		
						a	5		
					+---+				
+-----+---+---+---+---+---+---+---+---+
```

```
1 (13!:9) 5
```

Rerun

6

13!:11 y

**Error Number.** Last error number

13!:12 y

**Error Message.** Last error message

13! : 13 y

**Stack.** Produces a 9-column matrix of information on the (named) verbs/adverbs/conjunctions on the stack:

- 0 Name
- 1 Error number or 0 if not in error
- 2 Line number
- 3 Name class: 3, 1, or 2, denoting verb, adverb, or conjunction
- 4 Linear representation of the entity
- 5 The name of the defining script
- 6 Argument(s) individually boxed
- 7 Locals as a 2-column matrix of name and value
- 8 \* if begins a suspension; a blank otherwise

In columns 6 and 7, nouns are included *per se*, and verb, adverb, and conjunction are represented by their linear representation. For example:

```

mean=: sum % #
sum=: plus/
plus=: 4 : 'x.+y.'
mean 'abcd'
|domain error: plus
|   x.      +y.
|plus[:0]
```

13! : 13 ' '

### Note tacit definitions

have no locals

|      |   |   |   |             |        |         |   |
|------|---|---|---|-------------|--------|---------|---|
| plus | 3 | 0 | 3 | 4 : 'x.+y.' | +-+--+ | +--+--+ | * |
|      |   |   |   |             | c d    | x. c    |   |
|      |   |   |   |             | +-+--+ | +--+--+ |   |
|      |   |   |   |             |        | y. d    |   |
|      |   |   |   |             |        | +--+--+ |   |
| sum  | 0 | 0 | 3 | plus/       | +----+ |         |   |
|      |   |   |   |             | abcd   |         |   |
|      |   |   |   |             | +----+ |         |   |
| mean | 0 | 0 | 3 | sum % #     | +----+ |         |   |
|      |   |   |   |             | abcd   |         |   |

```

|      | | | |      | | +----+ |      | |
+----+ +

```

13!:14 y

**Query Latent Expression.**

13!:15 y

**Set Latent Expression.** The latent expression is executed when execution is about to be suspended; error messages are suppressed; any continuation must be programmed in the latent expression.

13!:16 y

**Trace.** See below.

13!:17 y

**Query.** Is suspension enabled? (Set by 13!:0)

13!:18 y

**Stack Text.** Like 13!:1, but gives the stack as a literal matrix.

A *debug suspension* is an immediate execution state with a non-empty execution stack. Another debug suspension is created when a named object (verb, adverb, or conjunction) is invoked in a debug suspension and it too suspends.

Four verbs are provided. They are part of the debug facility in the J development environment and usually are not invoked directly. Their behaviour and availability are subject to change without notice.

13!:19 y

**Cut Back.** Cut back one stack level, stopping at the line at the next stack level.

[x] 13!:20 y

**Step Over.** Run the current line (or line x if specified) to completion, stopping at the next line.

[x] 13!:21 y

**Step Into.** Run the current line (or line x if specified), stopping at the next line.

[x] 13!:22 y

**Step Out.** Run the current object to completion, starting with the current line (or line x if specified), stopping at the next line.

## Dynamic Link Library

**15! :**

See help files and the script `system\main\dll.ijs`.

|           |                          |
|-----------|--------------------------|
| x 15!:0 y | <b>Call DLL Function</b> |
| 15!:1 y   | <b>Memory Read</b>       |
| x 15!:2 y | <b>Memory Write</b>      |
| 15!:3 y   | <b>Allocate Memory</b>   |
| 15!:4 y   | <b>Release Memory</b>    |

## Locales

**18! :**

See also [Section II.I p71](#) and the "Locales" lab under menu item `Studio|Labs...|Locales`.

`18!:0 y`

**Name Class.** Give the name class of the locale named `y`, with `0` for named, `1` for numbered, `_1` for non-existent, and `_2` for illegal name. Thus:

```
18!:0 ;:'base j z 45bad asdf 0'
0 0 0 _2 _1 1
```

`[x] 18!:1 y`

**Name List.** Give the names of named (0) or numbered (1) locales. The optional left argument specifies the initial letters of names. Thus:

```
18!:1 [0           All named locales
+---+---+---+---+---+---+
|base|j|jcfg|jnewuser|newuser|z|
+---+---+---+---+---+---+
```

```
asdf_bb_=: 'sesquipedalian'
```

```
'jb' 18!:1 [0           All named locales beginning in j or
b
```

```
+---+---+---+---+---+---+
|base|bb|j|jcfg|jnewuser|
+---+---+---+---+---+---+
```

```
18!:3 ''           Create a numbered locale
+-+
|0|
+-+
```

```
18!:1 i.2           All named and numbered locales
+---+---+---+---+---+---+
|0|base|j|jcfg|jnewuser|newuser|z|
```

```

+-+-----+-+-----+-----+-----+-----+

```

```
[x] 18!:2 y
```

**Path.** The monad gives the (search) path for locale  $y$ ; the dyad sets the path for locale  $y$  to  $x$ . The path of a locale is initially  $, <, 'z'$ , except that the path of locale  $z$  is empty initially. If a name sought in locale  $f$  is not found in  $f$ , then it is sought in the locales in the path of  $f$  (but not searching *their* paths). For example:

```

( ; : 'a cd b' ) 18!:2 < 'f'
18!:2 < 'f'
+-+-----+-+
|a|cd|b|
+-+-----+-+

```

The path of locale  $f$  is set to  $a$ ,  $cd$ , and  $b$ .

```
[x] 18!:3 y
```

**Create.** If  $y$  is the empty string, then create a previously-unused numbered locale. If  $y$  is a name, then (re-)create the named locale; an error is signalled if the named locale already exists and is non-empty. The result is the name of the created locale.

$x$  specifies the size of the hash table for the locale, requiring approximately the same space as  $i.2^{6+x}$ . If  $x$  is elided, then the defaults specified in [9!:39 p223](#) are used. A larger hash table size improves performance; regardless of the hash table size, a locale may contain an effectively unlimited number of names.

```

18!:3 ''          Create a numbered locale
+-+
|0|
+-+
18!:3 ''          Create another one
+-+
|1|
+-+
18!:1 [1          Names of numbered locales
+-+-----+-+
|0|1|
+-+-----+-+

```

18!:4 y

**Switch Current.** Switch the current locale to *y* at the end of the currently executing named verb. Initially the current locale is *base*.

18!:5 y

**Current.** The name of the current locale. For example:

```
18!:5 ''
+-----+
|base|
+-----+
```

18!:55 y

**Erase.** Erase locale *y* (once it finishes execution). A numbered locale, once erased, may not be reused; a named locale may be reused at will.



**Miscellaneous****128! :**

128!:0 y

**QR.** Produces the QR decomposition of a complex matrix  $y$  (in the domain of matrix inverse %.), an Hermitian matrix and a square upper triangular matrix, individually boxed.

```

x=: +/ . *
A=: j./?. 2 7 4$10
matrix
$A
7 4
'Q R'=: 128!:0 A
$Q
7 4
$R
4 4
>./|,(=i.4) - (+|:Q) x Q
6.33846e_16
0~:R
1 1 1 1
0 1 1 1
0 0 1 1
0 0 0 1
A -: Q x R
1

```

Matrix product  
A random complex matrix

Q is Hermitian

R is upper triangular

128!:1 y

**R Inv.** Invert square upper triangular matrix.

`x 128!:2 y`

**Apply.** `x 128!:2 y` applies the verb in string `x` to `y` .  
For example:

```
'+' 128!:2 i.2 5
5 7 9 11 13
'+' 128!:2"1 i.2 5
10 35
'+' "1 128!:2 i.2 5
10 35
('+' ; '|.' ; '|.'"1') 128!:2&.><i.2 5
+-----+-----+-----+
|5 7 9 11 13|5 6 7 8 9|4 3 2 1 0|
|           |0 1 2 3 4|9 8 7 6 5|
+-----+-----+-----+

'2 3' 128!:2 i.2 5
|syntax error
|  '2 3'      128!:2 i.2 5

'@' 128!:2 i.2 5
|syntax error
|  '@'        128!:2 i.2 5
```

The ranks of `128!:2` are `1 _` , that is, apply the lists in the left argument to the right argument *in toto*.

## Appendix B. Special Code

Many primitives contain special code for certain arguments to effect time and/or space savings not available to general arguments. Moreover, some phrases are "recognized" and are supported by special code. For example, the dyad of the hook ( $\$,$ ) is exactly the reshape of APL (denoted by  $\rho$ ); its implementation avoids actually ravelling the right argument, and in so doing saves both time and space:

```
ts=: 6!:2 , 7!:2@]
x=: 11 13 17 19 23
y=: 29 7 23 11 19$'sesquipedalian'

(x ($,) y) -: x $, y
1

ts 'x ($,) y'
0.00773981 2.09818e6
ts 'x $, y'
0.0170125 3.14662e6
```

Instances of such special code are listed below:

|      |      |                                                                                                                              |
|------|------|------------------------------------------------------------------------------------------------------------------------------|
| =    | dyad | word-parallel operation on Boolean arguments for the following verbs:<br>= < <. <: > >. >: +. +: * *. *: ^ ~:  <br>!         |
| <.@f | both | avoids non-integer intermediate results on extended precision integers                                                       |
| >.@f | both | avoids non-integer intermediate results on extended precision integers                                                       |
| +    | dyad | also * and - ; on Windows, assembly code for integer arguments for the vector-vector, vector-scalar, and scalar-vector cases |
| ^    | dyad | $x^y$ works by repeated multiplication if $x$ is real and $y$ is integral                                                    |

|                                     |       |                                                                                                                                                                                                                                                                                           |
|-------------------------------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>m&amp;   @^</code>            | dyad  | avoids exponentiation for extended precision arguments                                                                                                                                                                                                                                    |
| <code>m&amp;   @ ( n&amp;^ )</code> | monad | avoids exponentiation for extended precision arguments                                                                                                                                                                                                                                    |
| <code>+ / . *</code>                | dyad  | special code                                                                                                                                                                                                                                                                              |
| <code>- / . *</code>                | monad | special code in general; special code for square matrices; special code for arrays of 2-by-2 matrices; see the J 4.05 release notes                                                                                                                                                       |
| <code>\$ ,</code>                   | dyad  | also <code>( \$ , ) "r</code> ; also <code>{ , { . , } . , e . ,</code> ; avoids ravel; see the J 4.06 release notes                                                                                                                                                                      |
| <code>[ ; . 0</code>                | both  | also <code>]</code> ; special code for vector or matrix right arguments; see the J 5.01 release notes                                                                                                                                                                                     |
| <code>f ; . 1</code>                | both  | also <code>f ; . _1 f ; . 2 f ; . _2</code> ; avoids building argument cells for several verbs: <code>&lt; \$ , # [ ] { . { : &lt;@ } . &lt;@ } :</code> ; also <code>&lt; &amp; } . &lt;@ : } .</code> etc.; special code for sparse Boolean left argument; see the J 4.05 release notes |
| <code>f ; . 3</code>                | both  | also <code>f ; . _3</code> ; special code for matrix right arguments                                                                                                                                                                                                                      |
| <code>#</code>                      | dyad  | special code for Boolean left arguments                                                                                                                                                                                                                                                   |
| <code># i . @ #</code>              | monad | also <code>( # i . &amp; # )</code> , etc.; avoids <code>i .</code> on Boolean arguments; see the J 4.06 release notes                                                                                                                                                                    |
| <code># : i . @ ( * / )</code>      | monad | also <code>( # : i . &amp; ( * / ) )</code> , etc.; special code for non-negative integer vectors; see the J 4.05 release notes                                                                                                                                                           |
| <code>= /</code>                    | monad | also <code>&lt; &lt; : &gt; &gt; : + . + : * * . * : ~ :</code> ; word-parallel operations on Boolean arguments                                                                                                                                                                           |
| <code>+ /</code>                    | monad | also <code>*</code> and <code>-</code> ; on Windows, assembly code for integer arguments                                                                                                                                                                                                  |
| <code>, /</code>                    | monad | also <code>, . , . &amp; . &gt; ;</code> ; linear time; see the J 4.05 release notes                                                                                                                                                                                                      |
| <code>m b . /</code>                | both  | special code for bitwise Boolean functions; see the J 5.01 release notes                                                                                                                                                                                                                  |

|                           |       |                                                                                                                                                                                                                                                                                                         |
|---------------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>f/@,</code>         | monad | also <code>f/@:</code> , <code>f/∧</code> , <code>f/∧:</code> , <code>;</code> ; avoids ravel; see the J 4.05 release notes                                                                                                                                                                             |
| <code>#/.</code>          | dyad  | avoids building argument cells                                                                                                                                                                                                                                                                          |
| <code>/:</code>           | both  | also <code>\:</code> ; special code for several data types; special code for arguments with 5 items or less; see the J 4.05 release notes                                                                                                                                                               |
| <code>/:</code>           | dyad  | special code when the left and right arguments are the same boolean, literal, integer, or floating point vector; also for <code>/:"1</code> when the left and right arguments are the same boolean, literal, integer, or floating point arrays; also for <code>\:</code> ; see the J 5.01 release notes |
| <code>/:~</code>          | monad | special code for boolean, literal, integer, or floating point vectors; also for <code>/:~"1</code> and <code>/:"1~</code> ; also for <code>\:</code> ; see the J 5.01 release notes                                                                                                                     |
| <code>=/\</code>          | monad | also <code>+</code> , <code>*</code> , <code>~:</code> ; word-parallel operations on Boolean arguments                                                                                                                                                                                                  |
| <code>+/\</code>          | monad | also <code>*</code> and <code>-</code> ; on Windows, assembly code for integer arguments                                                                                                                                                                                                                |
| <code>[ \</code>          | dyad  | also <code>]</code> and <code>,</code> ; see the J 5.01 release notes                                                                                                                                                                                                                                   |
| <code>2 f/\y</code>       | dyad  | also <code>2 f~/\y</code> ; special code for atomic <code>f</code> and vector <code>y</code> ; see the J 4.06 release notes                                                                                                                                                                             |
| <code>m b./\</code>       | monad | special code for bitwise Boolean functions; see the J 5.01 release notes                                                                                                                                                                                                                                |
| <code>=/\.</code>         | monad | also <code>&lt;</code> , <code>&lt;:</code> , <code>&gt;</code> , <code>&gt;:</code> , <code>+</code> , <code>+:</code> , <code>*</code> , <code>*:</code> , <code>~:</code> ; word-parallel operations on Boolean arguments                                                                            |
| <code>+/\.</code>         | monad | also <code>*</code> and <code>-</code> ; on Windows, assembly code for integer arguments                                                                                                                                                                                                                |
| <code>m b./\.</code>      | monad | special code for bitwise Boolean functions; see the J 5.01 release notes                                                                                                                                                                                                                                |
| <code>{</code>            | dyad  | special code for right arguments of several data types; special code for integer left arguments; special code for indexing first two axes                                                                                                                                                               |
| <code>&lt;"1@[ { ]</code> | dyad  | avoids <code>&lt;"1</code> if left argument is integer array                                                                                                                                                                                                                                            |

|                             |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>a=: c}x,y,:z</code>   | -     | avoids catenation and lamination; in-place if <code>c</code> is Boolean and <code>a</code> is <code>x</code> or <code>y</code> ; see the J 4.05 release notes                                                                                                                                                                                                                                                                                           |
| <code>y=: x i}y</code>      | -     | in-place                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>f"r</code>            | both  | <p>numerous verbs have integrated rank support:</p> <p>= &lt; &lt;. &lt;: &gt; &gt;. &gt;: + +. +: * *. *: - -: % ^ ~:    .  : \$ , ,. ,: # ! [ ] { { . { : } . } : / /: \ \. \: b. e. i. i: o. p. p:</p>                                                                                                                                                                                                                                               |
| <code>i.&amp;1@:&lt;</code> | dyad  | <p>special code for Boolean, integer, floating point, or literal vectors or scalars, for the following functions, where <code>comp</code> is one of <code>= ~: &lt; &lt;: &gt;: &gt;</code> (<code>=</code> and <code>~:</code> only for literal). See the J 5.01 release notes.</p> <pre> i.&amp;0@:comp      comp i. 0: i.&amp;1@:comp      comp i. 1: i:&amp;0@:comp      comp i: 0: i:&amp;1@:comp      comp i: 1: +/@:comp       [: +/ comp </pre> |
| <code>?</code>              | monad | also <code>?.</code> ; special code if argument is identically 2                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>?</code>              | dyad  | also <code>?.</code> ; special code if left argument is much smaller than right argument                                                                                                                                                                                                                                                                                                                                                                |
| <code>E.</code>             | monad | special code for Boolean and literal vector arguments                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>i.</code>             | monad | also <code>i:</code> ; special case for length-1 arguments                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>i.</code>             | dyad  | <p>also <code>e.</code> and <code>i:</code>; special code for several data types; special code for <code>i.!0</code>; special code for the monad <code>i.~</code> or <code>x i. x</code>; special code for arguments with many identical columns (see the J 4.05 release notes)</p>                                                                                                                                                                     |

## Appendix C. System Limits

error number    (right argument of `13! : 8`) less than 256

print precision less than or equal to 20

tolerance        less than or equal to  $2^{\_34}$

# Index

## A

- a. [Sample Topics p31](#) • [Sample 1. Spelling p32](#) • [Sample 2. Alphabet and Numbers p33](#) • [Sample 12. Sorting p43](#) • [H. Frets and Scripts p70](#) • [J. Errors and Suspensions p72](#) • [Vocabulary p74](#) • [< Box - Less Than p77](#) • [^: Power p100](#) • [|. Reverse - Rotate \(Shift\) p110](#) • [|: Transpose p111](#) • [;. Cut p122](#) • [/: Grade Up - Sort p132](#) • [\: Grade Down - Sort p135](#) • [p140](#) • [} Item Amend - Amend p142](#) • [} Item Amend • Amend p143](#) • [}. Behead - Drop p144](#) • [}: Curtail - p145](#) • [" Rank p147](#) • [" Rank p148](#) • [a. Alphabet p162](#) • [s: Symbol p188](#)
- A. [Intro 12. Reading and Writing p13](#) • [Intro 22. Recursion p23](#) • [Intro 25. Permutations p26](#) • [Intro 26. Linear Functions p27](#) • [A. Nouns p63](#) • [J. Errors and Suspensions p72](#) • [Vocabulary p74](#) • [:: Adverse p117](#) • [A. Anagram Index - Anagram p163](#) • [C. Cycle-Direct - Permute p167](#) • [Foreign conjunction p214](#)
- a: [I. Alphabet and Words p61](#) • [Vocabulary p74](#) • [{. Head - Take p139](#) • [{:: Map - Fetch p141](#) • [a. Alphabet p162](#)
- abbreviated [Intro 25. Permutations p26](#)
- abbreviation(s) [Intro 9. Vocabulary p10](#) • [III. Definitions p73](#)
- Abramowitz [H. Hypergeometric p174](#) • [References p212](#)
- abs [Monad-Dyad p115](#) • [\[: Cap p137](#)
- ace [I. Alphabet and Words p61](#) • [Vocabulary p74](#) • [a. Alphabet p162](#)
- acknowledgments [Acknowledgments p213](#)



|              |                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACM          | <a href="#">References p212</a>                                                                                                                                                                                                                                                                                                                                                      |
| actuarial    | <a href="#">Sample 28. Polynomials: Stopes p59</a>                                                                                                                                                                                                                                                                                                                                   |
| adverbial    | <a href="#">E. Parsing and Execution p67</a>                                                                                                                                                                                                                                                                                                                                         |
| adverse      | <a href="#">Vocabulary p74</a> • <a href="#">:: Adverse p117</a>                                                                                                                                                                                                                                                                                                                     |
| agenda(s)    | <a href="#">Intro 21. Gerund and Agenda p22</a> • <a href="#">Intro 23. Iteration p24</a> • <a href="#">Vocabulary p74</a> • <a href="#">* Signum - Times p89</a> • <a href="#">\$: Self-Reference p104</a> • <a href="#">  Magnitude - Residue p109</a> • <a href="#">` Tie (Gerund) p151</a> • <a href="#">@. Agenda p154</a>                                                      |
| agree(s)     | <a href="#">Sample 13. Compositions I p44</a> • <a href="#">B. Verbs p64</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">*: Square - Not-And p91</a> • <a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">~: Nub Sieve - Not-Equal p108</a> • <a href="#">. Determinant • Dot Product p112</a>       |
| agreement(s) | <a href="#">B. Verbs p64</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">* Signum - Times p89</a> • <a href="#">%: Square Root - Root p97</a>                                                                                                                                                                                                                        |
| algebra      | <a href="#">II. Grammar p62</a> • <a href="#">\$. Sparse p103</a>                                                                                                                                                                                                                                                                                                                    |
| algebraic    | <a href="#">\$. Sparse p103</a>                                                                                                                                                                                                                                                                                                                                                      |
| alphabet     | <a href="#">Sample Topics p31</a> • <a href="#">Sample 2. Alphabet and Numbers p33</a> • <a href="#">Sample 8. Classification p39</a> • <a href="#">Sample 12. Sorting p43</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">Vocabulary p74</a> • <a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a> • <a href="#">a. Alphabet p162</a> |
| alphabetized | <a href="#">Sample 12. Sorting p43</a>                                                                                                                                                                                                                                                                                                                                               |
| ambivalence  | <a href="#">Intro 2. Ambivalence p3</a>                                                                                                                                                                                                                                                                                                                                              |
| ambivalent   | <a href="#">Introduction p1</a> • <a href="#">Intro 2. Ambivalence p3</a>                                                                                                                                                                                                                                                                                                            |
| amend(s)     | <a href="#">Vocabulary p74</a> • <a href="#">\$. Sparse p103</a> • <a href="#">} Item Amend - Amend p142</a> • <a href="#">} Item Amend • Amend p143</a>                                                                                                                                                                                                                             |
| amendment(s) | <a href="#">\$. Sparse p103</a>                                                                                                                                                                                                                                                                                                                                                      |
| anagram(s)   | <a href="#">Intro 25. Permutations p26</a> • <a href="#">Vocabulary p74</a> • <a href="#">A. Anagram Index - Anagram p163</a>                                                                                                                                                                                                                                                        |
| antibase     | <a href="#">Vocabulary p74</a> • <a href="#">#: Antibase 2 - Antibase p126</a>                                                                                                                                                                                                                                                                                                       |
| APL          | <a href="#">Dictionary p60</a> • <a href="#">References p212</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                 |

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| appose           | <a href="#">Vocabulary p74</a> • <a href="#">&amp;: Appose p160</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| approx           | <a href="#">Intro 29. Secondaries p30</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| approximate(s)   | <a href="#">Sample 23. Polynomials p54</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| approximated     | <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">%. Matrix Inverse - Matrix Divide p96</a>                                                                                                                                                                                                                                                                                                                                                                                                                         |
| approximately    | <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">^ Exponential - Power p98</a> • <a href="#">o. Pi Times - Circle Function p182</a> • <a href="#">9!: Global Parameters p223</a> • <a href="#">18!: Locales p227</a>                                                                                                                                                                                                                                                                                               |
| approximating    | <a href="#">H. Hypergeometric p174</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| approximation(s) | <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">^: Power v p101</a> • <a href="#">D. Derivative p169</a> • <a href="#">D: Secant Slope p170</a> • <a href="#">t. Taylor Coefficient p191</a> • <a href="#">T. Taylor Approximation p193</a> |
| arc(s)           | <a href="#">Sample 20. Directed Graphs p51</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| arccos           | <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| arccosh          | <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| arcsine          | <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| arcsinh          | <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| arctan           | <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| arctanh          | <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| arithmetic       | <a href="#">Introduction p1</a> • <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 5. Forks p6</a> • <a href="#">Sample 16. Partitions I p47</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> • <a href="#">&gt;. Ceiling - Larger of (Max) p81</a> • <a href="#">+ Conjugate - Plus p86</a> • <a href="#">- Negate - Minus p92</a> • <a href="#">.. Even p113</a> • <a href="#">&amp;. p158</a> • <a href="#">s: Symbol p188</a>                      |
| ascii            | <a href="#">I. Alphabet and Words p61</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| assert(s)        | <a href="#">: Explicit / p114</a> • <a href="#">Control structures p200</a> • <a href="#">assert. p201</a> • <a href="#">9!: Global Parameters p223</a>                                                                                                                                                                                                                                                                                                                                                                           |

assertion(s) [Intro 28. Identity Functions and Neutrals p29](#) • [\\$. Sparse p103](#) • [: Explicit / p114](#) • [assert. p201](#) • [9!: Global Parameters p223](#)

assignment(s) [Introduction p1](#) • [Intro 1. Mnemonics p2](#) • [=. Is \(Local\) p76](#)

atom(s) [Intro 9. Vocabulary p10](#) • [Intro 20. Rank p21](#) • [A. Nouns p63](#) • [F. Trains p68](#) • [G. Extended and Rational Arithmeti p69](#) • [> Open - Larger Than p80](#) • [^: Power p100](#) • [: Explicit / p114](#) • [. Ravel - Append p118](#) • [. Ravel Items - Stitch p119](#) • [# Tally - Copy p124](#) • [#: Antibase 2 - Antibase p126](#) • [{ Catalogue - From p138](#) • [{. Head - Take p139](#) • [p140](#) • [{:: Map - Fetch p141](#) • [} Item Amend - Amend p142](#) • [}. Behead - Drop p144](#) • [b. Boolean / p164](#) • [c. Characteristic Values p166](#) • [C. Cycle-Direct - Permute p167](#) • [D. Derivative p169](#) • [e. Raze In - Member \(In\) p171](#) • [i. Integers - Index Of p175](#) • [Control structures p200](#) • [if. p206](#) • [while. p211](#) • [3!: Conversions p218](#)

atomic [< Box - Less Than p77](#) • [\\$. Sparse p103](#) • [:: Adverse p117](#) • [. Ravel - Append p118](#) • [.: Itemize - Laminate p120](#) • [#. Base 2 - Base p125](#) • [` Tie \(Gerund\) p151](#) • [5!: Representation p220](#) • [9!: Global Parameters p223](#) • [Special Code p229](#)

atop [Intro 8. Atop Conjunction p9](#) • [Intro 20. Rank p21](#) • [Vocabulary p74](#) • [@ Atop p153](#)

axe(s) [Intro 20. Rank p21](#) • [A. Nouns p63](#) • [B. Verbs p64](#) • [\\$. Sparse p103](#) • [|. Reverse - Rotate \(Shift\) p110](#) • [|: Transpose p111](#) • [{ Catalogue - From p138](#) • [Special Code p229](#)

## B

b. [Intro 20. Rank p21](#) • [Intro 28. Identity Functions and Neutrals p29](#) • [Vocabulary p74](#) • [/ Insert - Table p130](#) • [" Rank p148](#) • [`: Evoke Gerund p152](#) • [@: At p155](#) • [&: Appose p160](#) • [b. Boolean / p164](#) • [Basic p165](#) • [q: Prime Factors - Prime Exponents p186](#) • [Special Code p229](#)

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| backspace   | <a href="#">Intro 10. Housekeeping p11</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| barchart(s) | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Sample 8. Classification p39</a> • <a href="#">Sample 9. Disjoint Classification (Graphs) p40</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| bident(s)   | <a href="#">E. Parsing and Execution p67</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| binary      | <a href="#">#: Antibase 2 - Antibase p126</a> • <a href="#">s: Symbol p188</a> • <a href="#">3!:</a> <a href="#">Conversions p218</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| binomial(s) | <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Intro 15. Defined Adverbs p16</a> • <a href="#">Intro 22. Recursion p23</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Intro 29. Secondaries p30</a> • <a href="#">! Factorial - Out Of p127</a> • <a href="#">p. Polynomial p183</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| bits        | <a href="#">b. Boolean / p164</a> • <a href="#">3!:</a> <a href="#">Conversions p218</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| bitwise     | <a href="#">b. Boolean / p164</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| bold        | <a href="#">E. Parsing and Execution p67</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| boole       | <a href="#">Intro 1. Mnemonics p2</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| boolean(s)  | <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">Sample 9. Disjoint Classification (Graphs) p40</a> • <a href="#">Sample 11. Classification II p42</a> • <a href="#">Sample 20. Directed Graphs p51</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">+. Real / Imaginary - GCD (Or) p87</a> • <a href="#">*. Length/Angle - LCM (And) p90</a> • <a href="#">-. Not - Less p93</a> • <a href="#">^: Power p100</a> • <a href="#">\$. Sparse p103</a> • <a href="#">~: Nub Sieve - Not-Equal p108</a> • <a href="#">;. Cut p122</a> • <a href="#">/ Insert - Table p130</a> • <a href="#">&amp;. p158</a> • <a href="#">b. Boolean / p164</a> • <a href="#">e. Raze In - Member (In) p171</a> • <a href="#">3!:</a> <a href="#">Conversions p218</a> • <a href="#">Special Code p229</a> |
| box         | <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">Sample 10. Classification I p41</a> • <a href="#">Sample 16. Partitions I p47</a> • <a href="#">Sample 17. Partitions II p48</a> • <a href="#">A. Nouns p63</a> • <a href="#">F. Trains p68</a> • <a href="#">Vocabulary p74</a> • <a href="#">&lt; Box - Less Than p77</a> • <a href="#">&gt; Open - Larger Than p80</a> • <a href="#"> : Transpose p111</a> • <a href="#">": Default Format - Format p150</a> • <a href="#">s: Symbol p188</a> • <a href="#">5!:</a> <a href="#">Representation p220</a> • <a href="#">9!:</a> <a href="#">Global Parameters p223</a>                                                                                                                                                                                                                                                              |

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| boxed    | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 6. Programs p7</a> • <a href="#">Intro 10. Housekeeping p11</a> • <a href="#">Intro 16. Word Formation p17</a> • <a href="#">Intro 19. Tacit Equivalents p20</a> • <a href="#">Intro 22. Recursion p23</a> • <a href="#">Intro 25. Permutations p26</a> • <a href="#">Sample 2. Alphabet and Numbers p33</a> • <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">A. Nouns p63</a> • <a href="#">B. Verbs p64</a> • <a href="#">H. Frets and Scripts p70</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">=. Is (Local) p76</a> • <a href="#">&lt; Box - Less Than p77</a> • <a href="#">&gt; Open - Larger Than p80</a> • <a href="#">\$. Sparse p103</a> • <a href="#"> : Transpose p111</a> • <a href="#">: Explicit / p114</a> • <a href="#">; Raze - Link p121</a> • <a href="#">:: Word Formation - p123</a> • <a href="#">!: Foreign p129</a> • <a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a> • <a href="#">{ Catalogue - From p138</a> • <a href="#">}. Head - Take p139</a> • <a href="#">{:: Map - Fetch p141</a> • <a href="#">": Default Format - Format p150</a> • <a href="#">` Tie (Gerund) p151</a> • <a href="#">@. Agenda p154</a> • <a href="#">a. Alphabet p162</a> • <a href="#">C. Cycle-Direct - Permute p167</a> • <a href="#">L. Level Of p178</a> • <a href="#">p. Polynomial p183</a> • <a href="#">p.. Poly. Deriv. - Poly. Integral p184</a> • <a href="#">s: Symbol p188</a> • <a href="#">S: Spread p189</a> • <a href="#">Foreign conjunction p214</a> (only first 40 listed) |
| boxes    | <a href="#">Intro 14. Partitions p15</a> • <a href="#">A. Nouns p63</a> • <a href="#">C. Cycle-Direct - Permute p167</a> • <a href="#">9!: Global Parameters p223</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| break(s) | <a href="#">_ Negative Sign / Infinity p83</a> • <a href="#">: Explicit / p114</a> • <a href="#">Control structures p200</a> • <a href="#">break. p202</a> • <a href="#">continue. p203</a> • <a href="#">for. p204</a> • <a href="#">while. p211</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| byte(s)  | <a href="#">\$. Sparse p103</a> • <a href="#">s: Symbol p188</a> • <a href="#">u: Unicode p195</a> • <a href="#">3!: Conversions p218</a> • <a href="#">6!: Time p221</a> • <a href="#">7!: Space p222</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

## C

|    |                                                                                                                                                                                                                                                                                                                                                                           |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| C. | <a href="#">Intro 25. Permutations p26</a> • <a href="#">C. Adverbs and Conjunctions p65</a> • <a href="#">Vocabulary p74</a> • <a href="#">^: Power p100</a> • <a href="#">/ Insert - Table p130</a> • <a href="#">} Item Amend - Amend p142</a> • <a href="#">C. Cycle-Direct - Permute p167</a> • <a href="#">References p212</a> • <a href="#">System Limits p230</a> |
| c. | <a href="#">Vocabulary p74</a> • <a href="#">c. Characteristic Values p166</a>                                                                                                                                                                                                                                                                                            |

|                   |                                                                                                                                                                                                                                                                      |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| calculus          | <a href="#">Sample 28. Polynomials: Stopes p59</a> • <a href="#">^ Exponential - Power p98</a> • <a href="#">D. Derivative p169</a>                                                                                                                                  |
| cardinality       | <a href="#">s: Symbol p188</a>                                                                                                                                                                                                                                       |
| caret             | <a href="#">Intro 9. Vocabulary p10</a>                                                                                                                                                                                                                              |
| cartesian         | <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">{ Catalogue - From p138</a>                                                                                                                                                              |
| category          | <a href="#">9!: Global Parameters p223</a>                                                                                                                                                                                                                           |
| catenated         | <a href="#">B. Verbs p64</a> • <a href="#">.: Itemize - Laminate p120</a> • <a href="#">u: Unicode p195</a>                                                                                                                                                          |
| catenation        | <a href="#">Special Code p229</a>                                                                                                                                                                                                                                    |
| ceiling           | <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">D. Comparatives p66</a> • <a href="#">Vocabulary p74</a> • <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> • <a href="#">&gt;. Ceiling - Larger of (Max) p81</a>                       |
| celsius           | <a href="#">Intro 12. Reading and Writing p13</a>                                                                                                                                                                                                                    |
| characteristic(s) | <a href="#">Intro 20. Rank p21</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Vocabulary p74</a> • <a href="#">" Rank p148</a> • <a href="#">Basic p165</a> • <a href="#">c. Characteristic Values p166</a>                       |
| circle            | <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Vocabulary p74</a> • <a href="#">* Signum - Times p89</a> • <a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">o. Pi Times - Circle Function p182</a> • <a href="#">r. Angle - Polar p187</a> |
| circular          | <a href="#">Intro 9. Vocabulary p10</a>                                                                                                                                                                                                                              |
| circumference     | <a href="#">Intro 9. Vocabulary p10</a>                                                                                                                                                                                                                              |
| clipped           | <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                      |
| coeff(s)          | <a href="#">Intro 22. Recursion p23</a> • <a href="#">Intro 29. Secondaries p30</a>                                                                                                                                                                                  |

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| coefficient(s) | <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Intro 4. Punctuation p5</a> • <a href="#">Intro 15. Defined Adverbs p16</a> • <a href="#">Intro 21. Gerund and Agenda p22</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Intro 29. Secondaries p30</a> • <a href="#">Sample 23. Polynomials p54</a> • <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">Sample 28. Polynomials: Stopes p59</a> • <a href="#">Vocabulary p74</a> • <a href="#">#. Base 2 - Base p125</a> • <a href="#">! Factorial - Out Of p127</a> • <a href="#">/. Oblique - Key p131</a> • <a href="#">p. Polynomial p183</a> • <a href="#">p.. Poly. Deriv. - Poly. Integral p184</a> • <a href="#">t. Taylor Coefficient p191</a> • <a href="#">t: Weighted Taylor p192</a> |
| commutative    | <a href="#">Intro 3. Verbs and Adverbs p4</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| commute(s)     | <a href="#">~ Reflex - Passive p105</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| commuted       | <a href="#">F. Trains p68</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| comparative(s) | <a href="#">D. Comparatives p66</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| compared       | <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">select. p208</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| comparisons    | <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">s: Symbol p188</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| component(s)   | <a href="#">III. Definitions p73</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| compose        | <a href="#">Vocabulary p74</a> • <a href="#">Compose p157</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| composition(s) | <a href="#">Sample 13. Compositions I p44</a> • <a href="#">Sample 14. Compositions II p45</a> • <a href="#">Sample 23. Polynomials p54</a> • <a href="#">C. Adverbs and Conjunctions p65</a> • <a href="#">Compose p157</a> • <a href="#">&amp;. p158</a> • <a href="#">D. Derivative p169</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| conj           | <a href="#">E. Parsing and Execution p67</a> • <a href="#">: Explicit / p114</a> • <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| conjugate(s)   | <a href="#">Sample 2. Alphabet and Numbers p33</a> • <a href="#">Vocabulary p74</a> • <a href="#">+ Conjugate - Plus p86</a> • <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| conjunction(s) | <a href="#">Introduction p1</a> • <a href="#">Intro 6. Programs p7</a> • <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 8. Atop Conjunction p9</a> • <a href="#">Intro 10. Housekeeping p11</a> • <a href="#">Intro 11. Power and Inverse p12</a> • <a href="#">Intro 15. Defined Adverbs p16</a> • <a href="#">Intro 18. Explicit Definition p19</a> • <a href="#">Intro 20. Rank p21</a> • <a href="#">Intro 21. Gerund and Agenda p22</a> • <a href="#">Intro 23. Iteration p24</a> • <a href="#">Intro 24. Trains p25</a> • <a href="#">Sample Topics p31</a> • <a href="#">Sample 3. Grammar p34</a> • <a href="#">Sample 13. Compositions I p44</a> • <a href="#">Sample 17. Partitions II p48</a> • <a href="#">Sample 28. Polynomials: Stopes p59</a> • <a href="#">Dictionary p60</a> • <a href="#">II. Grammar p62</a> • <a href="#">B. Verbs p64</a> • <a href="#">C. Adverbs and Conjunctions p65</a> • <a href="#">D. Comparatives p66</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">F. Trains p68</a> • <a href="#">H. Frets and Scripts p70</a> • <a href="#">J. Errors and Suspensions p72</a> • <a href="#">III. Definitions p73</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">&lt;: Decrement - Less Or Equal p79</a> • <a href="#">* Signum - Times p89</a> • <a href="#">^ Exponential - Power p98</a> • <a href="#">^: Power p100</a> • <a href="#">\$ Shape Of - Shape p102</a> • <a href="#">~. Nub - p107</a> • <a href="#">~: Nub Sieve - Not-Equal p108</a> • <a href="#">  Magnitude - Residue p109</a> • <a href="#">: Explicit / p114</a> • <a href="#">:. Obverse p116</a> • <a href="#">, Ravel - Append p118</a><br>(only first 40 listed) |
| conjunctival   | <a href="#">E. Parsing and Execution p67</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| constants      | <a href="#">I. Alphabet and Words p61</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">". Do - Numbers p149</a> • <a href="#">Constants p199</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| control(s)     | <a href="#">Intro 10. Housekeeping p11</a> • <a href="#">Intro 18. Explicit Definition p19</a> • <a href="#">Intro 23. Iteration p24</a> • <a href="#">Sample 2. Alphabet and Numbers p33</a> • <a href="#">II. Grammar p62</a> • <a href="#">Vocabulary p74</a> • <a href="#">&lt; Box - Less Than p77</a> • <a href="#">  Magnitude - Residue p109</a> • <a href="#">: Explicit / p114</a> • <a href="#">a. Alphabet p162</a> • <a href="#">Control structures p200</a> • <a href="#">break. p202</a> • <a href="#">continue. p203</a> • <a href="#">for. p204</a> • <a href="#">goto_name. p205</a> • <a href="#">if. p206</a> • <a href="#">select. p208</a> • <a href="#">try. p210</a> • <a href="#">while. p211</a> • <a href="#">Acknowledgments p213</a> • <a href="#">5!: Representation p220</a> • <a href="#">6!: Time p221</a> • <a href="#">9!: Global Parameters p223</a> • <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |



## D

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| D.             | <a href="#">Intro 29. Secondaries p30</a> • <a href="#">D. Comparatives p66</a> • <a href="#">Vocabulary p74</a> • <a href="#">d. Derivative p168</a> • <a href="#">D. Derivative p169</a> • <a href="#">D: Secant Slope p170</a> • <a href="#">t. Taylor Coefficient p191</a>                                                                                                                                                                                                                                                                                                                               |
| d.             | <a href="#">Sample 23. Polynomials p54</a> • <a href="#">Vocabulary p74</a> • <a href="#">^: Natural Log - Logarithm p99</a> • <a href="#">d. Derivative p168</a> • <a href="#">p. Polynomial p183</a>                                                                                                                                                                                                                                                                                                                                                                                                       |
| D:             | <a href="#">Vocabulary p74</a> • <a href="#">^: Power p100</a> • <a href="#">D. Derivative p169</a> • <a href="#">D: Secant Slope p170</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| date(s)        | <a href="#">A. Nouns p63</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| deal(s)        | <a href="#">Vocabulary p74</a> • <a href="#">? Roll - Deal p161</a> • <a href="#">9!: Global Parameters p223</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| debug          | <a href="#">try. p210</a> • <a href="#">Foreign conjunction p214</a> • <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| debugging      | <a href="#">J. Errors and Suspensions p72</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| decoded        | <a href="#">&lt; Box - Less Than p77</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| denominator(s) | <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">x: Extended Precision p197</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| derivative(s)  | <a href="#">Intro 29. Secondaries p30</a> • <a href="#">Sample 23. Polynomials p54</a> • <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">Vocabulary p74</a> • <a href="#">^: Natural Log - Logarithm p99</a> • <a href="#">d. Derivative p168</a> • <a href="#">D. Derivative p169</a> • <a href="#">D: Secant Slope p170</a> • <a href="#">p. Polynomial p183</a> • <a href="#">p.. Poly. Deriv. - Poly. Integral p184</a> • <a href="#">t. Taylor Coefficient p191</a> |
| derive(s)      | <a href="#">III. Definitions p73</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| derived        | <a href="#">II. Grammar p62</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">D. Derivative p169</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| determinant(s)  | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Intro 29. Secondaries p30</a> • <a href="#">Sample 18. Geometry p49</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">. Determinant • Dot Product p112</a> • <a href="#">? Roll - Deal p161</a> • <a href="#">C. Cycle-Direct - Permute p167</a> • <a href="#">D. Derivative p169</a> |
| diagonal(s)     | <a href="#">Intro 14. Partitions p15</a> • <a href="#">Intro 29. Secondaries p30</a> • <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> • <a href="#">\$. Sparse p103</a> • <a href="#">/. Oblique - Key p131</a> • <a href="#">c. Characteristic Values p166</a>                                                                                                                                                                                                              |
| dimension(s)    | <a href="#">Sample 18. Geometry p49</a> • <a href="#">\$. Sparse p103</a> • <a href="#">;. Cut p122</a> • <a href="#">_9: to 9: Constant Functions p198</a>                                                                                                                                                                                                                                                                                                                     |
| dimensional     | <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| dir(s)          | <a href="#">2!: Host p217</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| distributed     | <a href="#">I. Locatives p71</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| distributing    | <a href="#">~. Nub - p107</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| distribution(s) | <a href="#">~. Nub - p107</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| div             | <a href="#">Sample 7. Tables p38</a>                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| divide(s)       | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">Vocabulary p74</a> • <a href="#">+. Real / Imaginary - GCD (Or) p87</a> • <a href="#">% Reciprocal - Divide p95</a> • <a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">D. Derivative p169</a>                                                                                                       |
| divided         | <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">_ Negative Sign / Infinity p83</a> • <a href="#">*. Length/Angle - LCM (And) p90</a> • <a href="#">% Reciprocal - Divide p95</a> • <a href="#">\$. Sparse p103</a> • <a href="#">[: Cap p137</a> • <a href="#">s: Symbol p188</a> • <a href="#">Constants p199</a> • <a href="#">References p212</a>                                                                                                       |
| dividing        | <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">  Magnitude - Residue p109</a>                                                                                                                                                                                                                                                                                                                                                                                 |
| divisibility    | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Sample 7. Tables p38</a>                                                                                                                                                                                                                                                                                                                                                                                        |
| divisible       | <a href="#">G. Extended and Rational Arithmeti p69</a>                                                                                                                                                                                                                                                                                                                                                                                                                          |
| division(s)     | <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">H. Frets and Scripts p70</a> • <a href="#">% Reciprocal - Divide p95</a> • <a href="#">D. Derivative p169</a>                                                                                                                                                                                                                                                                                                               |

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| divisor(s) | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Sample 7. Tables p38</a> • <a href="#">+. Real / Imaginary - GCD (Or) p87</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| dot(s)     | <a href="#">Vocabulary p74</a> • <a href="#">. Determinant • Dot Product p112</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| doubled    | <a href="#">@ Atop p153</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| doubles    | <a href="#">3!: Conversions p218</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| draw(s)    | <a href="#">9!: Global Parameters p223</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| drop(s)    | <a href="#">Intro 10. Housekeeping p11</a> • <a href="#">Vocabulary p74</a> • <a href="#">}. Behead - Drop p144</a> • <a href="#">}: Curtail - p145</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| dual(s)    | <a href="#">Vocabulary p74</a> • <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> • <a href="#">&gt;. Ceiling - Larger of (Max) p81</a> • <a href="#">+: Double • Not-Or p88</a> • <a href="#">*: Square - Not-And p91</a> • <a href="#">~: Nub Sieve - Not-Equal p108</a> • <a href="#">&amp;. p158</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| duality    | <a href="#">+. Real / Imaginary - GCD (Or) p87</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| uplicated  | <a href="#">s: Symbol p188</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| dyad(s)    | <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Intro 5. Forks p6</a> • <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 8. Atop Conjunction p9</a> • <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Intro 20. Rank p21</a> • <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">B. Verbs p64</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">Vocabulary p74</a> • <a href="#">&lt;: Decrement - Less Or Equal p79</a> • <a href="#">&gt; Open - Larger Than p80</a> • <a href="#">&gt;: Increment - Larger Or Equal p82</a> • <a href="#">* Signum - Times p89</a> • <a href="#">^: Power p100</a> • <a href="#">\$. Sparse p103</a> • <a href="#">~: Nub Sieve - Not-Equal p108</a> • <a href="#">  Magnitude - Residue p109</a> • <a href="#">Monad-Dyad p115</a> • <a href="#">! Factorial - Out Of p127</a> • <a href="#">/ Insert - Table p130</a> • <a href="#">\ Prefix - Infix p133</a> • <a href="#">{. Head - Take p139</a> • <a href="#">&amp; Bond / p156</a> • <a href="#">b. Boolean / p164</a> • <a href="#">s: Symbol p188</a> • <a href="#">u: Unicode p195</a> • <a href="#">3!: Conversions p218</a> • <a href="#">9!: Global Parameters p223</a> • <a href="#">13!: Debug p225</a> • <a href="#">18!: Locales p227</a> • <a href="#">Special Code p229</a> |

dyadic [Intro 2. Ambivalence p3](#) • [Intro 3. Verbs and Adverbs p4](#) • [Intro 5. Forks p6](#) • [Intro 6. Programs p7](#) • [Intro 19. Tacit Equivalents p20](#) • [Intro 28. Identity Functions and Neutrals p29](#) • [Sample 13. Compositions I p44](#) • [B. Verbs p64](#) • [C. Adverbs and Conjunctions p65](#) • [G. Extended and Rational Arithmetic p69](#) • [H. Frets and Scripts p70](#) • [III. Definitions p73](#) • [= Self-Classify - Equal p75](#) • [%. Matrix Inverse - Matrix Divide p96](#) • [|. Reverse - Rotate \(Shift\) p110](#) • [. Determinant • Dot Product p112](#) • [: Explicit / p114](#) • [Monad-Dyad p115](#) • [, Ravel - Append p118](#) • [,. Ravel Items - Stitch p119](#) • [,: Itemize - Laminate p120](#) • [;. Cut p122](#) • [!. Fit \(Customize\) p128](#) • [\[: Cap p137](#) • [{ Catalogue - From p138](#) • [Compose p157](#) • [b. Boolean / p164](#) • [H. Hypergeometric p174](#) • [5!: Representation p220](#) • [13!: Debug p225](#)

dyadically [Intro 7. Bond Conjunction p8](#) • [Intro 8. Atop Conjunction p9](#) • [Intro 9. Vocabulary p10](#) • [Sample 16. Partitions I p47](#) • [E. Parsing and Execution p67](#)

## E

e. [Intro 12. Reading and Writing p13](#) • [Sample 20. Directed Graphs p51](#) • [I. Alphabet and Words p61](#) • [Vocabulary p74](#) • [\\$. Sparse p103](#) • [!. Fit \(Customize\) p128](#) • [/ Insert - Table p130](#) • [e. Raze In - Member \(In\) p171](#) • [s: Symbol p188](#) • [for. p204](#) • [select. p208](#) • [Special Code p229](#)

E. [E. Parsing and Execution p67](#) • [Vocabulary p74](#) • [!. Fit \(Customize\) p128](#) • [E. - Member of Interval p172](#) • [Special Code p229](#)

eigen [c. Characteristic Values p166](#)

eigenvalue(s) [c. Characteristic Values p166](#)

eigenvector(s) [c. Characteristic Values p166](#)

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| enclose      | <a href="#">Intro 16. Word Formation p17</a>                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| enclosed     | <a href="#">I. Alphabet and Words p61</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">Control structures p200</a>                                                                                                                                                                                                                                                                                                                                                     |
| enclosing    | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">break. p202</a> • <a href="#">continue. p203</a>                                                                                                                                                                                                                                                                                                                                  |
| encoded      | <a href="#">3!: Conversions p218</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| encoding(s)  | <a href="#">&lt; Box - Less Than p77</a> • <a href="#">5!: Representation p220</a>                                                                                                                                                                                                                                                                                                                                                                                                     |
| equality     | <a href="#">D. Comparatives p66</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">!. Fit (Customize) p128</a>                                                                                                                                                                                                                                                                                                                                                            |
| equals       | <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 26. Linear Functions p27</a> • <a href="#">Sample 23. Polynomials p54</a> • <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">&gt;: Increment - Larger Or Equal p82</a> • <a href="#">+: Double</a> • <a href="#">Not-Or p88</a> • <a href="#">\$. Sparse p103</a> • <a href="#">}_ Item Amend - Amend p142</a> • <a href="#">5!: Representation p220</a> |
| equation(s)  | <a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">^: Power p100</a>                                                                                                                                                                                                                                                                                                                                                                                                  |
| equivalence  | <a href="#">Intro 6. Programs p7</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| equivalently | <a href="#">Intro 26. Linear Functions p27</a> • <a href="#">/. Oblique - Key p131</a>                                                                                                                                                                                                                                                                                                                                                                                                 |
| equivalents  | <a href="#">Intro 19. Tacit Equivalents p20</a>                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| euler(s)     | <a href="#">^ Exponential - Power p98</a> • <a href="#">q: Prime Factors - Prime Exponents p186</a> • <a href="#">Constants p199</a>                                                                                                                                                                                                                                                                                                                                                   |
| evaluate(s)  | <a href="#">Intro 4. Punctuation p5</a> • <a href="#">Sample 19. Symbolic Functions p50</a> • <a href="#">~. Nub - p107</a>                                                                                                                                                                                                                                                                                                                                                            |
| evaluated    | <a href="#">E. Parsing and Execution p67</a> • <a href="#">=. Is (Local) p76</a> • <a href="#">assert. p201</a> • <a href="#">for. p204</a> • <a href="#">select. p208</a> • <a href="#">9!: Global Parameters p223</a>                                                                                                                                                                                                                                                                |
| evaluating   | <a href="#">~. Nub - p107</a> • <a href="#">for. p204</a>                                                                                                                                                                                                                                                                                                                                                                                                                              |
| evaluation   | <a href="#">Intro 21. Gerund and Agenda p22</a> • <a href="#">for. p204</a> • <a href="#">select. p208</a>                                                                                                                                                                                                                                                                                                                                                                             |
| evoke        | <a href="#">Vocabulary p74</a> • <a href="#">EVOKE p106</a> • <a href="#">`: Evoke Gerund p152</a>                                                                                                                                                                                                                                                                                                                                                                                     |
| evoked       | <a href="#">=. Is (Local) p76</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| executes     | <a href="#">". Do - Numbers p149</a> • <a href="#">0!: Scripts p215</a>                                                                                                                                                                                                                                                                                                                                                                                                                |

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| executing      | <a href="#">Intro 10. Housekeeping p11</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">I. Locatives p71</a> • <a href="#">9!: Global Parameters p223</a> • <a href="#">18!: Locales p227</a>                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| executions     | <a href="#">E. Parsing and Execution p67</a> • <a href="#">6!: Time p221</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| expand(s)      | <a href="#">Intro 29. Secondaries p30</a> • <a href="#">^: Power p100</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| exponent(s)    | <a href="#">Intro 4. Punctuation p5</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">p. Polynomial p183</a> • <a href="#">q: Prime Factors - Prime Exponents p186</a>                                                                                                                                                                                                                                                                                                                                                                                                                         |
| exponential(s) | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">Vocabulary p74</a> • <a href="#">^: Exponential - Power p98</a> • <a href="#">^: Natural Log - Logarithm p99</a> • <a href="#">": Default Format - Format p150</a> • <a href="#">&amp;. p158</a> • <a href="#">D. Derivative p169</a> • <a href="#">t: Weighted Taylor p192</a> • <a href="#">Constants p199</a> |
| exponentiation | <a href="#">while. p211</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

## F

|           |                                                                                                                                                                                                                                                                                                                                                  |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| f.        | <a href="#">Intro 6. Programs p7</a> • <a href="#">Sample 21. Closure p52</a> • <a href="#">Vocabulary p74</a> • <a href="#">\$: Self-Reference p104</a> • <a href="#">f. Fix p173</a> • <a href="#">m. n. Explicit Noun Args p180</a> • <a href="#">u. v. Explicit Verb Args p194</a>                                                           |
| factor(s) | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Sample 28. Polynomials: Stopes p59</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">+. Real / Imaginary - GCD (Or) p87</a> • <a href="#">q: Prime Factors - Prime Exponents p186</a> • <a href="#">s: Symbol p188</a> |

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| factorial(s)     | <a href="#">Intro 18. Explicit Definition p19</a> • <a href="#">Intro 22. Recursion p23</a> • <a href="#">Sample 13. Compositions I p44</a> • <a href="#">Sample 28. Polynomials: Stopes p59</a> • <a href="#">Vocabulary p74</a> • <a href="#">^ Exponential - Power p98</a> • <a href="#">! Factorial - Out Of p127</a> • <a href="#">!. Fit (Customize) p128</a> • <a href="#">@. Agenda p154</a> • <a href="#">Compose p157</a> • <a href="#">H. Hypergeometric p174</a> • <a href="#">t: Weighted Taylor p192</a> |
| factorization(s) | <a href="#">q: Prime Factors - Prime Exponents p186</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| float(s)         | <a href="#">3!: Conversions p218</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| floor(s)         | <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">D. Comparatives p66</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> • <a href="#">&gt;. Ceiling - Larger of (Max) p81</a> • <a href="#">[: Cap p137</a> • <a href="#">References p212</a>                                                                                                            |
| fork(s)          | <a href="#">Intro 4. Punctuation p5</a> • <a href="#">Intro 5. Forks p6</a> • <a href="#">Intro 6. Programs p7</a> • <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 24. Trains p25</a> • <a href="#">Sample 13. Compositions I p44</a> • <a href="#">Sample 14. Compositions II p45</a> • <a href="#">F. Trains p68</a> • <a href="#">&gt;. Ceiling - Larger of (Max) p81</a> • <a href="#">[: Cap p137</a> • <a href="#">5!: Representation p220</a>                                                |
| fractal(s)       | <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">References p212</a>                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| fraction(s)      | <a href="#">Sample 16. Partitions I p47</a> • <a href="#">+. Real / Imaginary - GCD (Or) p87</a> • <a href="#">". Do - Numbers p149</a> • <a href="#">s: Symbol p188</a>                                                                                                                                                                                                                                                                                                                                               |
| fractional       | <a href="#">  Magnitude - Residue p109</a> • <a href="#">6!: Time p221</a>                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| functional(s)    | <a href="#">Introduction p1</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## G

|          |                                                                                                                                                                                                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| gamma    | <a href="#">! Factorial - Out Of p127</a>                                                                                                                                                       |
| gaussian | <a href="#">&lt;. Floor - Lesser Of (Min) p78</a>                                                                                                                                               |
| gcd      | <a href="#">Vocabulary p74</a> • <a href="#">+. Real / Imaginary - GCD (Or) p87</a> • <a href="#">*. Length/Angle - LCM (And) p90</a> • <a href="#">q: Prime Factors - Prime Exponents p186</a> |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geometric     | <a href="#">Intro 5. Forks p6</a> • <a href="#">Intro 8. Atop Conjunction p9</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| geometrically | <a href="#">%. Matrix Inverse - Matrix Divide p96</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| geometry      | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Sample 18. Geometry p49</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| gerund(s)     | <a href="#">Intro 21. Gerund and Agenda p22</a> • <a href="#">Vocabulary p74</a> • <a href="#">=. Is (Local) p76</a> • <a href="#">^: Power p100</a> • <a href="#">;. Cut p122</a> • <a href="#">/ Insert - Table p130</a> • <a href="#">/. Oblique - Key p131</a> • <a href="#">\ Prefix - Infix p133</a> • <a href="#">\ . Suffix - Outfix p134</a> • <a href="#">} Item Amend - Amend p142</a> • <a href="#">} Item Amend • Amend p143</a> • <a href="#">` Tie (Gerund) p151</a> • <a href="#">`: Evoke Gerund p152</a> • <a href="#">@. Agenda p154</a> • <a href="#">References p212</a> • <a href="#">5!: Representation p220</a> |
| goto          | <a href="#">: Explicit / p114</a> • <a href="#">Control structures p200</a> • <a href="#">goto_name. p205</a> • <a href="#">5!: Representation p220</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| grade(s)      | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Sample 12. Sorting p43</a> • <a href="#">A. Nouns p63</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a>                                                                                                                                                                                                                                                                                                                                                    |
| graded        | <a href="#">s: Symbol p188</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| grading       | <a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a> • <a href="#">a. Alphabet p162</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| grammar(s)    | <a href="#">Introduction p1</a> • <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Sample 3. Grammar p34</a> • <a href="#">II. Grammar p62</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

## H

|           |                                                                                                                                                  |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| H.        | <a href="#">H. Frets and Scripts p70</a> • <a href="#">Vocabulary p74</a> • <a href="#">H. Hypergeometric p174</a>                               |
| halve(s)  | <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Vocabulary p74</a> • <a href="#">-: Halve - Match p94</a> • <a href="#">@. Agenda p154</a> |
| halved    | <a href="#">@ Atop p153</a>                                                                                                                      |
| handbook  | <a href="#">References p212</a>                                                                                                                  |
| head(s)   | <a href="#">Vocabulary p74</a> • <a href="#">{. Head - Take p139</a>                                                                             |
| hermitian | <a href="#">128!: Miscellaneous p228</a>                                                                                                         |



hex [3!: Conversions p218](#)

hexadecimal [3!: Conversions p218](#)

hilbert [G. Extended and Rational Arithmeti p69](#) • [7!: Space p222](#)

hyperbolic(s) [Intro 28. Identity Functions and Neutrals p29](#) • [o. Pi Times - Circle Function p182](#)

hypergeometric(s) [Vocabulary p74](#) • [H. Hypergeometric p174](#)

## I

i. [Intro 2. Ambivalence p3](#) • [Intro 3. Verbs and Adverbs p4](#) • [Intro 5. Forks p6](#) • [Intro 6. Programs p7](#) • [Intro 7. Bond Conjunction p8](#) • [Intro 8. Atop Conjunction p9](#) • [Intro 10. Housekeeping p11](#) • [Intro 11. Power and Inverse p12](#) • [Intro 12. Reading and Writing p13](#) • [Intro 13. Format p14](#) • [Intro 14. Partitions p15](#) • [Intro 15. Defined Adverbs p16](#) • [Intro 18. Explicit Definition p19](#) • [Intro 20. Rank p21](#) • [Intro 21. Gerund and Agenda p22](#) • [Intro 22. Recursion p23](#) • [Intro 23. Iteration p24](#) • [Intro 25. Permutations p26](#) • [Intro 26. Linear Functions p27](#) • [Intro 27. Obverse and Under p28](#) • [Intro 28. Identity Functions and Neutrals p29](#) • [Intro 29. Secondaries p30](#) • [Sample Topics p31](#) • [Sample 1. Spelling p32](#) • [Sample 2. Alphabet and Numbers p33](#) • [Sample 3. Grammar p34](#) • [Sample 4. Function Tables p35](#) • [Sample 7. Tables p38](#) • [Sample 8. Classification p39](#) • [Sample 9. Disjoint Classification \(Graphs\) p40](#) • [Sample 10. Classification I p41](#) • [Sample 11. Classification II p42](#) • [Sample 12. Sorting p43](#) • [Sample 13. Compositions I p44](#) • [Sample 17. Partitions II p48](#) • [Sample 18. Geometry p49](#) • [Sample 20. Directed Graphs p51](#) • [Sample 21. Closure p52](#) • [Sample 22. Distance p53](#) • [Sample 23. Polynomials p54](#) (only first 40 listed)

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| i:            | <a href="#">Vocabulary p74</a> • <a href="#">\$. Sparse p103</a> • <a href="#">!. Fit (Customize) p128</a> • <a href="#">i: Integers - Index Of Last p176</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                                                                          |
| image(s)      | <a href="#">Intro 5. Forks p6</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| imaginary     | <a href="#">Intro 13. Format p14</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">Vocabulary p74</a> • <a href="#">+. Real / Imaginary - GCD (Or) p87</a> • <a href="#">*. Length/Angle - LCM (And) p90</a> • <a href="#">%: Square Root - Root p97</a> • <a href="#">/: Grade Up - Sort p132</a> • <a href="#">c. Characteristic Values p166</a> • <a href="#">j. Imaginary - Complex p177</a> • <a href="#">r. Angle - Polar p187</a> • <a href="#">Constants p199</a> |
| imaginarypart | <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| increment(s)  | <a href="#">Intro 5. Forks p6</a> • <a href="#">Vocabulary p74</a> • <a href="#">&lt;: Decrement - Less Or Equal p79</a> • <a href="#">&gt;: Increment - Larger Or Equal p82</a> • <a href="#">D: Secant Slope p170</a> • <a href="#">5!: Representation p220</a>                                                                                                                                                                                                                                                                          |
| indexed       | <a href="#">\$. Sparse p103</a> • <a href="#">;. Cut p122</a> • <a href="#">1!: Files p216</a>                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| indexing      | <a href="#">Sample 12. Sorting p43</a> • <a href="#">* Signum - Times p89</a> • <a href="#">\$. Sparse p103</a> • <a href="#">b. Boolean / p164</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                                                                                    |
| inductively   | <a href="#">Sample Topics p31</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| inexact       | <a href="#">G. Extended and Rational Arithmeti p69</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| infinity      | <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">Vocabulary p74</a> • <a href="#">_ Negative Sign / Infinity p83</a> • <a href="#">. Indeterminate p84</a> • <a href="#">_: Infinity p85</a> • <a href="#">9!: Global Parameters p223</a>                                                                                                                                                                                                         |
| infix         | <a href="#">Sample 16. Partitions I p47</a> • <a href="#">Vocabulary p74</a> • <a href="#">\ Prefix - Infix p133</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                                                                                                                   |
| infixes       | <a href="#">\ Prefix - Infix p133</a> • <a href="#">\. Suffix - Outfix p134</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| inflected     | <a href="#">I. Alphabet and Words p61</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| inflection(s) | <a href="#">I. Alphabet and Words p61</a> • <a href="#">III. Definitions p73</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| integral(s)   | <a href="#">Sample 23. Polynomials p54</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">\$. Sparse p103</a> • <a href="#">D. Derivative p169</a> • <a href="#">p.. Poly. Deriv. - Poly. Integral p184</a> • <a href="#">9!: Global Parameters p223</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                      |

integrate [Intro 29. Secondaries p30](#)

integrated [Special Code p229](#)

inverses [Intro 27. Obverse and Under p28](#) • [3!: Conversions p218](#)

invert [128!: Miscellaneous p228](#)

invertible [Intro 26. Linear Functions p27](#) • [^: Power p100](#)

invoke(s) [\\$. Sparse p103](#) • [\[: Cap p137](#) • [2!: Host p217](#)

invoked [Intro 10. Housekeeping p11](#) • [Intro 17. Names and Displays p18](#) • [: Explicit / p114](#) • [4!: Names p219](#) • [13!: Debug p225](#)

iota [III. Definitions p73](#)

## J

j. [Intro 18. Explicit Definition p19](#) • [Vocabulary p74](#) • [<. Floor - Lesser Of \(Min\) p78](#) • [+ Conjugate - Plus p86](#) • [+. Real / Imaginary - GCD \(Or\) p87](#) • [\\* Signum - Times p89](#) • [^: Power p100](#) • [\\$. Sparse p103](#) • [# Tally - Copy p124](#) • [": Default Format - Format p150](#) • [D. Derivative p169](#) • [i: Integers - Index Of Last p176](#) • [j. Imaginary - Complex p177](#) • [o. Pi Times - Circle Function p182](#) • [break. p202](#) • [continue. p203](#) • [for. p204](#) • [128!: Miscellaneous p228](#)

jsoftware [s: Symbol p188](#)

## K

keyboard [Sample Topics p31](#) • [: Explicit / p114](#) • [!: Foreign p129](#) • [u: Unicode p195](#) • [1!: Files p216](#)

keystrokes [H. Frets and Scripts p70](#)

## L

|            |                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| L.         | <a href="#">Vocabulary p74</a> · <a href="#">: Explicit / p114</a> · <a href="#">{:: Map - Fetch p141</a> · <a href="#">L. Level Of p178</a> · <a href="#">select. p208</a>                                                                                                                                                                                                                                             |
| L:         | <a href="#">Vocabulary p74</a> · <a href="#">: Explicit / p114</a> · <a href="#">{:: Map - Fetch p141</a> · <a href="#">L: Level At p179</a> · <a href="#">S: Spread p189</a>                                                                                                                                                                                                                                           |
| laminate   | <a href="#">Vocabulary p74</a> · <a href="#">.: Itemize - Laminate p120</a>                                                                                                                                                                                                                                                                                                                                             |
| lamination | <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                                                                                                                       |
| languages  | <a href="#">Introduction p1</a>                                                                                                                                                                                                                                                                                                                                                                                         |
| largest    | <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> · <a href="#">\$. Sparse p103</a> · <a href="#">#: Antibase 2 - Antibase p126</a> · <a href="#">C. Cycle-Direct - Permute p167</a>                                                                                                                                                                                                                                    |
| latent     | <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                                                                                                                                                                         |
| lesser     | <a href="#">Intro 1. Mnemonics p2</a> · <a href="#">Intro 3. Verbs and Adverbs p4</a> · <a href="#">Intro 28. Identity Functions and Neutrals p29</a> · <a href="#">Vocabulary p74</a> · <a href="#">&lt;. Floor - Lesser Of (Min) p78</a>                                                                                                                                                                              |
| lessor     | <a href="#">Intro 2. Ambivalence p3</a>                                                                                                                                                                                                                                                                                                                                                                                 |
| letter(s)  | <a href="#">Intro 29. Secondaries p30</a> · <a href="#">Sample 6. Tables (Letter Frequency) p37</a> · <a href="#">Sample 8. Classification p39</a> · <a href="#">I. Alphabet and Words p61</a> · <a href="#">A. Nouns p63</a> · <a href="#">i: Integers - Index Of Last p176</a> · <a href="#">Constants p199</a> · <a href="#">1!: Files p216</a> · <a href="#">4!: Names p219</a> · <a href="#">18!: Locales p227</a> |
| lexical    | <a href="#">A. Nouns p63</a> · <a href="#">A. Anagram Index - Anagram p163</a>                                                                                                                                                                                                                                                                                                                                          |
| library    | <a href="#">Foreign conjunction p214</a> · <a href="#">15!: Dynamic Link Library p226</a>                                                                                                                                                                                                                                                                                                                               |
| limit(s)   | <a href="#">Intro 11. Power and Inverse p12</a> · <a href="#">B. Verbs p64</a> · <a href="#">^: Power p100</a> · <a href="#">\$. Sparse p103</a> · <a href="#">H. Hypergeometric p174</a> · <a href="#">o. Pi Times - Circle Function p182</a> · <a href="#">9!: Global Parameters p223</a> · <a href="#">System Limits p230</a>                                                                                        |
| linearly   | <a href="#">%. Matrix Inverse - Matrix Divide p96</a>                                                                                                                                                                                                                                                                                                                                                                   |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| locale(s)    | <a href="#">Intro 17. Names and Displays p18</a> • <a href="#">I. Locatives p71</a> • <a href="#">throw. p209</a> • <a href="#">Foreign conjunction p214</a> • <a href="#">4!: Names p219</a> • <a href="#">6!: Time p221</a> • <a href="#">9!: Global Parameters p223</a> • <a href="#">18!: Locales p227</a>                                                                                                                                                                                                          |
| locals       | <a href="#">9!: Global Parameters p223</a> • <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| locative(s)  | <a href="#">Intro 17. Names and Displays p18</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">I. Locatives p71</a> • <a href="#">: Explicit / p114</a> • <a href="#">Acknowledgments p213</a>                                                                                                                                                                                                                                                                                                              |
| log(s)       | <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 15. Defined Adverbs p16</a> • <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">Vocabulary p74</a> • <a href="#">^ Natural Log - Logarithm p99</a> • <a href="#">Monad-Dyad p115</a> • <a href="#">&amp; Bond / p156</a> • <a href="#">Compose p157</a> • <a href="#">&amp;. p158</a> • <a href="#">D: Secant Slope p170</a>                             |
| logarithm(s) | <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">Sample 13. Compositions I p44</a> • <a href="#">Vocabulary p74</a> • <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> • <a href="#">_ Negative Sign / Infinity p83</a> • <a href="#">^ Exponential - Power p98</a> • <a href="#">^ Natural Log - Logarithm p99</a> • <a href="#">Monad-Dyad p115</a> • <a href="#">Compose p157</a> • <a href="#">&amp;. p158</a> |

## M

|           |                                                                                                                                                                                                                                                             |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| m.        | <a href="#">Vocabulary p74</a> • <a href="#">: Explicit / p114</a> • <a href="#">H. Hypergeometric p174</a> • <a href="#">m. n. Explicit Noun Args p180</a> • <a href="#">u. v. Explicit Verb Args p194</a> • <a href="#">x. y. Explicit Arguments p196</a> |
| macintosh | <a href="#">Intro 10. Housekeeping p11</a> • <a href="#">Intro 16. Word Formation p17</a> • <a href="#">2!: Host p217</a> • <a href="#">9!: Global Parameters p223</a>                                                                                      |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| magnitude(s) | <a href="#">Sample 22. Distance p53</a> • <a href="#">Sample 27. Polynomial Roots II p58</a><br>• <a href="#">A. Nouns p63</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a><br>• <a href="#">+ Conjugate - Plus p86</a> • <a href="#">*. Length/Angle - LCM (And) p90</a> •<br><a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">  Magnitude - Residue p109</a> •<br><a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a> • <a href="#">c. Characteristic Values p166</a> •<br><a href="#">D: Secant Slope p170</a> • <a href="#">o. Pi Times - Circle Function p182</a> •<br><a href="#">r. Angle - Polar p187</a> • <a href="#">Constants p199</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| math         | <a href="#">Intro 4. Punctuation p5</a> • <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Sample 13. Compositions I p44</a> •<br><a href="#">A. Nouns p63</a> • <a href="#">% Reciprocal - Divide p95</a> • <a href="#">^: Power p100</a> • <a href="#">. Determinant • Dot Product p112</a> •<br><a href="#">References p212</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| mathematical | <a href="#">Introduction p1</a> • <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Dictionary p60</a> •<br><a href="#">References p212</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| mathematics  | <a href="#">Introduction p1</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> •<br><a href="#">* Signum - Times p89</a> • <a href="#">Compose p157</a> • <a href="#">&amp;. p158</a> • <a href="#">References p212</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| matrices     | <a href="#">Introduction p1</a> • <a href="#">Sample 28. Polynomials: Stopes p59</a> • <a href="#">\$. Sparse p103</a> •<br><a href="#">. Determinant • Dot Product p112</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| matrix       | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 26. Linear Functions p27</a> • <a href="#">Intro 29. Secondaries p30</a> •<br><a href="#">Sample 10. Classification I p41</a> • <a href="#">Sample 15. Junctions p46</a> • <a href="#">Sample 18. Geometry p49</a> •<br><a href="#">Sample 20. Directed Graphs p51</a> • <a href="#">Sample 21. Closure p52</a> • <a href="#">Sample 28. Polynomials: Stopes p59</a> •<br><a href="#">A. Nouns p63</a> • <a href="#">B. Verbs p64</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> •<br><a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">\$ Shape Of - Shape p102</a> •<br><a href="#">\$. Sparse p103</a> • <a href="#">. Determinant • Dot Product p112</a> • <a href="#">.. Even p113</a> •<br><a href="#">;. Cut p122</a> • <a href="#">} Item Amend - Amend p142</a> • <a href="#">? Roll - Deal p161</a> •<br><a href="#">c. Characteristic Values p166</a> • <a href="#">D. Derivative p169</a> • <a href="#">p. Polynomial p183</a> •<br><a href="#">s: Symbol p188</a> • <a href="#">3!: Conversions p218</a> • <a href="#">5!: Representation p220</a> •<br><a href="#">13!: Debug p225</a> • <a href="#">128!: Miscellaneous p228</a> • <a href="#">Special Code p229</a> |

|               |                                                                                                                                                                                                                                                                                                                                         |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| member(s)     | <a href="#">Vocabulary p74</a> • <a href="#">e. Raze In - Member (In) p171</a> • <a href="#">E. - Member of Interval p172</a>                                                                                                                                                                                                           |
| membership(s) | <a href="#">I. Alphabet and Words p61</a>                                                                                                                                                                                                                                                                                               |
| memory        | <a href="#">7!: Space p222</a> • <a href="#">9!: Global Parameters p223</a> • <a href="#">15!: Dynamic Link Library p226</a>                                                                                                                                                                                                            |
| mnemonic(s)   | <a href="#">Introduction p1</a> • <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">Intro 9. Vocabulary p10</a>                                                                                                                                                                                                                       |
| modulo        | <a href="#">#: Antibase 2 - Antibase p126</a> • <a href="#">C. Cycle-Direct - Permute p167</a>                                                                                                                                                                                                                                          |
| monadically   | <a href="#">Intro 8. Atop Conjunction p9</a> • <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Sample 16. Partitions I p47</a>                                                                                                                                                                                                    |
| monads        | <a href="#">Intro 20. Rank p21</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">B. Verbs p64</a> • <a href="#">^: Power p100</a> • <a href="#">^: Power v p101</a> • <a href="#">\$. Sparse p103</a> • <a href="#">:. Cut p122</a> • <a href="#">[ Same - Left p136</a> • <a href="#">Compose p157</a> |
| monomial(s)   | <a href="#">Sample 23. Polynomials p54</a> • <a href="#">Sample 25. Polynomials in Terms of Roots p56</a>                                                                                                                                                                                                                               |
| multinomial   | <a href="#">p. Polynomial p183</a>                                                                                                                                                                                                                                                                                                      |
| multiplied    | <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">+ Conjugate - Plus p86</a>                                                                                                                                                                                                                                              |
| multiplier(s) | <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">p. Polynomial p183</a>                                                                                                                                                                                                                                       |
| multiplies    | <a href="#">+ Conjugate - Plus p86</a>                                                                                                                                                                                                                                                                                                  |
| multiply      | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">} Item Amend - Amend p142</a>                                                                                                                                                                                         |

## N

|      |                                                                                                                                                                                                                                                             |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n.   | <a href="#">Vocabulary p74</a> • <a href="#">: Explicit / p114</a> • <a href="#">H. Hypergeometric p174</a> • <a href="#">m. n. Explicit Noun Args p180</a> • <a href="#">u. v. Explicit Verb Args p194</a> • <a href="#">x. y. Explicit Arguments p196</a> |
| nand | <a href="#">+: Double</a> • <a href="#">Not-Or p88</a> • <a href="#">*: Square - Not-And p91</a>                                                                                                                                                            |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NB.          | <a href="#">Vocabulary p74</a> • <a href="#">NB. Comment p181</a> • <a href="#">throw. p209</a> • <a href="#">6!: Time p221</a>                                                                                                                                                                                                                                                                                                                                                                                           |
| neg          | <a href="#">Intro 28. Identity Functions and Neutrals p29</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| negate(s)    | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Vocabulary p74</a> • <a href="#">- Negate - Minus p92</a>                                                                                                                                                                                                                                                                                                                                                                                                           |
| negation     | <a href="#">Introduction p1</a> • <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> • <a href="#">&gt;. Ceiling - Larger of (Max) p81</a> • <a href="#">+.: Double • Not-Or p88</a> • <a href="#">*: Square - Not-And p91</a> • <a href="#">.. Even p113</a> • <a href="#">": Default Format - Format p150</a> • <a href="#">@ Atop p153</a> • <a href="#">Compose p157</a> • <a href="#">&amp;. p158</a> • <a href="#">D. Derivative p169</a> • <a href="#">Constants p199</a> |
| nested       | <a href="#">: Explicit / p114</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| nub          | <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">~. Nub - p107</a> • <a href="#">~: Nub Sieve - Not-Equal p108</a> • <a href="#">5!: Representation p220</a>                                                                                                                                                                                                                                                    |
| nubindex     | <a href="#">~. Nub - p107</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| numerator(s) | <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">x: Extended Precision p197</a>                                                                                                                                                                                                                                                                                                                                                                                                                       |

## O

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| o. | <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Intro 11. Power and Inverse p12</a> • <a href="#">Intro 18. Explicit Definition p19</a> • <a href="#">Intro 23. Iteration p24</a> • <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">Vocabulary p74</a> • <a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">^: Power p100</a> • <a href="#">\$. Sparse p103</a> • <a href="#">.. Even p113</a> • <a href="#">&amp; Bond / p156</a> • <a href="#">D. Derivative p169</a> • <a href="#">m. n. Explicit Noun Args p180</a> • <a href="#">o. Pi Times - Circle Function p182</a> • <a href="#">p.. Poly. Deriv. - Poly. Integral p184</a> • <a href="#">r. Angle - Polar p187</a> • <a href="#">t. Taylor Coefficient p191</a> • <a href="#">t: Weighted Taylor p192</a> • <a href="#">u. v. Explicit Verb Args p194</a> • <a href="#">x: Extended Precision p197</a> • <a href="#">Constants p199</a> • <a href="#">5!: Representation p220</a> • <a href="#">Special Code</a> |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



[p229](#)

obverse(s) [Intro 27. Obverse and Under p28](#) • [Vocabulary p74](#) • [^: Power p100](#) • [.: Obverse p116](#) • [&. p158](#) • [Basic p165](#)

operator(s) [G. Extended and Rational Arithmeti p69](#) • [5!: Representation p220](#) • [13!: Debug p225](#)

## P

p. [Sample 23. Polynomials p54](#) • [Sample 24. Polynomials \(Continued\) p55](#) • [Sample 25. Polynomials in Terms of Roots p56](#) • [Sample 26. Polynomial Roots I p57](#) • [Sample 27. Polynomial Roots II p58](#) • [Sample 28. Polynomials: Stopes p59](#) • [Vocabulary p74](#) • [%. Matrix Inverse - Matrix Divide p96](#) • [^: Power p100](#) • [^: Power v p101](#) • [.. Even p113](#) • [#. Base 2 - Base p125](#) • [!. Fit \(Customize\) p128](#) • [d. Derivative p168](#) • [p. Polynomial p183](#) • [p.. Poly. Deriv. - Poly. Integral p184](#) • [t. Taylor Coefficient p191](#) • [Special Code p229](#)

p: [Intro 12. Reading and Writing p13](#) • [G. Extended and Rational Arithmeti p69](#) • [Vocabulary p74](#) • [^: Power p100](#) • [p: Primes - p185](#) • [Special Code p229](#)

parentheses [Intro 4. Punctuation p5](#) • [Intro 5. Forks p6](#) • [Intro 8. Atop Conjunction p9](#) • [II. Grammar p62](#) • [E. Parsing and Execution p67](#) • [=. Is \(Local\) p76](#) • [@ Atop p153](#)

parenthesis [E. Parsing and Execution p67](#) • [H. Frets and Scripts p70](#)

parenthesization [@ Atop p153](#)

parenthesize(s) [E. Parsing and Execution p67](#)

parenthesized [@. Agenda p154](#) • [5!: Representation p220](#)

parse(s) [E. Parsing and Execution p67](#) • [13!: Debug p225](#)

parsed [E. Parsing and Execution p67](#) • [13!: Debug p225](#)

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| parsing       | <a href="#">E. Parsing and Execution p67</a> • <a href="#">F. Trains p68</a> • <a href="#">J. Errors and Suspensions p72</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| partition(s)  | <a href="#">Intro 14. Partitions p15</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Sample 16. Partitions I p47</a> • <a href="#">Sample 17. Partitions II p48</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| partitioning  | <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">/ Insert - Table p130</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| polynomial(s) | <a href="#">Intro 4. Punctuation p5</a> • <a href="#">Intro 14. Partitions p15</a> • <a href="#">Intro 21. Gerund and Agenda p22</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Intro 29. Secondaries p30</a> • <a href="#">Sample 23. Polynomials p54</a> • <a href="#">Sample 24. Polynomials (Continued) p55</a> • <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">Sample 28. Polynomials: Stopes p59</a> • <a href="#">Vocabulary p74</a> • <a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">^ Exponential - Power p98</a> • <a href="#">.. Even p113</a> • <a href="#">#. Base 2 - Base p125</a> • <a href="#">!. Fit (Customize) p128</a> • <a href="#">/. Oblique - Key p131</a> • <a href="#">D. Derivative p169</a> • <a href="#">p. Polynomial p183</a> • <a href="#">p.. Poly. Deriv. - Poly. Integral p184</a>                                                                         |
| posix         | <a href="#">1!: Files p216</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| power(s)      | <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Intro 4. Punctuation p5</a> • <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 11. Power and Inverse p12</a> • <a href="#">Intro 21. Gerund and Agenda p22</a> • <a href="#">Intro 23. Iteration p24</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Sample 4. Function Tables p35</a> • <a href="#">Sample 10. Classification I p41</a> • <a href="#">Sample 21. Closure p52</a> • <a href="#">Sample 23. Polynomials p54</a> • <a href="#">Sample 28. Polynomials: Stopes p59</a> • <a href="#">C. Adverbs and Conjunctions p65</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">^ Exponential - Power p98</a> • <a href="#">^.. Natural Log - Logarithm p99</a> • <a href="#">^: Power p100</a> • <a href="#">^: Power v p101</a> • <a href="#">&amp;. p158</a> • <a href="#">D. Derivative p169</a> • <a href="#">p. Polynomial p183</a> |
| precedence(s) | <a href="#">Sample 20. Directed Graphs p51</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| prime(s)       | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Sample 5. Bordering a Table p36</a> • <a href="#">Sample 7. Tables p38</a> • <a href="#">Sample 8. Classification p39</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">^: Power p100</a> • <a href="#">@. Agenda p154</a> • <a href="#">p: Primes - p185</a> • <a href="#">q: Prime Factors - Prime Exponents p186</a>                                                                                                                                                                                                                                                                                                                                                                                             |
| primitive(s)   | <a href="#">Introduction p1</a> • <a href="#">Intro 1. Mnemonics p2</a> • <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Intro 10. Housekeeping p11</a> • <a href="#">Intro 11. Power and Inverse p12</a> • <a href="#">Intro 18. Explicit Definition p19</a> • <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Intro 29. Secondaries p30</a> • <a href="#">Sample Topics p31</a> • <a href="#">Sample 23. Polynomials p54</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">\$. Sparse p103</a> • <a href="#">5!: Representation p220</a> • <a href="#">Special Code p229</a> |
| probability    | <a href="#">-. Not - Less p93</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| pronoun(s)     | <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Sample 3. Grammar p34</a> • <a href="#">I. Alphabet and Words p61</a> • <a href="#">II. Grammar p62</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">f. Fix p173</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| proposition(s) | <a href="#">Intro 6. Programs p7</a> • <a href="#">Sample 11. Classification II p42</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| proverb(s)     | <a href="#">Sample 3. Grammar p34</a> • <a href="#">II. Grammar p62</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">f. Fix p173</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| punctuate      | <a href="#">Intro 4. Punctuation p5</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

## Q

|    |                                                                                                                                                                                                                                       |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| q: | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">^: Power p100</a> • <a href="#">q: Prime Factors - Prime Exponents p186</a> |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## R

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| r.          | <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">Vocabulary p74</a> • <a href="#">*. Length/Angle - LCM (And) p90</a> • <a href="#">^: Power p100</a> • <a href="#">\$. Sparse p103</a> • <a href="#">r. Angle - Polar p187</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| radian(s)   | <a href="#">*. Length/Angle - LCM (And) p90</a> • <a href="#">o. Pi Times - Circle Function p182</a> • <a href="#">r. Angle - Polar p187</a> • <a href="#">Constants p199</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| radix       | <a href="#">#: Antibase 2 - Antibase p126</a> • <a href="#">/. Oblique - Key p131</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| random      | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">? Roll - Deal p161</a> • <a href="#">C. Cycle-Direct - Permute p167</a> • <a href="#">j. Imaginary - Complex p177</a> • <a href="#">9!: Global Parameters p223</a> • <a href="#">128!: Miscellaneous p228</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| randomly    | <a href="#">? Roll - Deal p161</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| rank(s)     | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">Intro 14. Partitions p15</a> • <a href="#">Intro 20. Rank p21</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Sample 13. Compositions I p44</a> • <a href="#">Sample 23. Polynomials p54</a> • <a href="#">A. Nouns p63</a> • <a href="#">B. Verbs p64</a> • <a href="#">F. Trains p68</a> • <a href="#">III. Definitions p73</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">&lt; Box - Less Than p77</a> • <a href="#">&gt; Open - Larger Than p80</a> • <a href="#">_: Infinity p85</a> • <a href="#">\$ Shape Of - Shape p102</a> • <a href="#">\$. Sparse p103</a> • <a href="#">. Determinant • Dot Product p112</a> • <a href="#">. Ravel - Append p118</a> • <a href="#">;. Cut p122</a> • <a href="#">/ Insert - Table p130</a> • <a href="#">/. Oblique - Key p131</a> • <a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a> • <a href="#">" Rank p146</a> • <a href="#">" Rank p147</a> • <a href="#">" Rank p148</a> • <a href="#">`: Evoke Gerund p152</a> • <a href="#">@ Atop p153</a> • <a href="#">@: At p155</a> • <a href="#">Compose p157</a> • <a href="#">&amp;: Appose p160</a> • <a href="#">Basic p165</a> • <a href="#">d. Derivative p168</a> • <a href="#">D. Derivative p169</a> • <a href="#">D: Secant Slope p170</a> • <a href="#">i. Integers - Index Of p175</a> • <a href="#">L: Level At p179</a> (only first 40 listed) |
| rational(s) | <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">x: Extended Precision p197</a> • <a href="#">Constants p199</a> • <a href="#">3!: Conversions p218</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

|               |                                                                                                                                                                                                                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ravel(s)      | <a href="#">B. Verbs p64</a> • <a href="#">Vocabulary p74</a> • <a href="#">\$ Shape Of - Shape p102</a> • <a href="#">\$. Sparse p103</a> • <a href="#">, Ravel - Append p118</a> • <a href="#">,. Ravel Items - Stitch p119</a> • <a href="#">; Raze - Link p121</a> • <a href="#">" Rank p148</a> • <a href="#">Special Code p229</a> |
| ravelled      | <a href="#">Intro 12. Reading and Writing p13</a> • <a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a> • <a href="#">b. Boolean / p164</a>                                                                                                                                                              |
| ravelling     | <a href="#">,. Ravel Items - Stitch p119</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                         |
| raze          | <a href="#">Vocabulary p74</a> • <a href="#">; Raze - Link p121</a> • <a href="#">e. Raze In - Member (In) p171</a>                                                                                                                                                                                                                      |
| reciprocal(s) | <a href="#">Intro 2. Ambivalence p3</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">% Reciprocal - Divide p95</a> • <a href="#">%. Matrix Inverse - Matrix Divide p96</a> • <a href="#">^ . Natural Log - Logarithm p99</a> • <a href="#">Constants p199</a>                 |
| recursion     | <a href="#">Intro 22. Recursion p23</a> • <a href="#">@. Agenda p154</a>                                                                                                                                                                                                                                                                 |
| recursive     | <a href="#">Intro 22. Recursion p23</a> • <a href="#">. Determinant • Dot Product p112</a>                                                                                                                                                                                                                                               |
| recursively   | <a href="#">Intro 17. Names and Displays p18</a> • <a href="#">f. Fix p173</a>                                                                                                                                                                                                                                                           |
| relational(s) | <a href="#">s: Symbol p188</a>                                                                                                                                                                                                                                                                                                           |
| rhematic      | <a href="#">Intro 16. Word Formation p17</a> • <a href="#">:: Word Formation - p123</a>                                                                                                                                                                                                                                                  |
| rotated       | <a href="#">C. Cycle-Direct - Permute p167</a>                                                                                                                                                                                                                                                                                           |
| rotates       | <a href="#"> . Reverse - Rotate (Shift) p110</a>                                                                                                                                                                                                                                                                                         |
| rotation(s)   | <a href="#"> . Reverse - Rotate (Shift) p110</a>                                                                                                                                                                                                                                                                                         |
| rounded       | <a href="#">Intro 2. Ambivalence p3</a>                                                                                                                                                                                                                                                                                                  |
| rounding      | <a href="#">\$. Sparse p103</a>                                                                                                                                                                                                                                                                                                          |

## S

|           |                                                                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------|
| s:        | <a href="#">Vocabulary p74</a> • <a href="#">^: Power p100</a> • <a href="#">{. Head - Take p139</a> • <a href="#">s: Symbol p188</a> |
| S:        | <a href="#">Vocabulary p74</a> • <a href="#">{:: Map - Fetch p141</a> • <a href="#">S: Spread p189</a>                                |
| school(s) | <a href="#">Sample 4. Function Tables p35</a>                                                                                         |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| script(s)    | <a href="#">Intro 10. Housekeeping p11</a> • <a href="#">Intro 18. Explicit Definition p19</a> • <a href="#">H. Frets and Scripts p70</a> • <a href="#">I. Locatives p71</a> • <a href="#">J. Errors and Suspensions p72</a> • <a href="#">: Explicit / p114</a> • <a href="#">Foreign conjunction p214</a> • <a href="#">0!: Scripts p215</a> • <a href="#">1!: Files p216</a> • <a href="#">3!: Conversions p218</a> • <a href="#">4!: Names p219</a> • <a href="#">13!: Debug p225</a> • <a href="#">15!: Dynamic Link Library p226</a> |
| self         | <a href="#">Intro 17. Names and Displays p18</a> • <a href="#">Intro 22. Recursion p23</a> • <a href="#">Intro 23. Iteration p24</a> • <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">- Negate - Minus p92</a> • <a href="#">^: Power p100</a> • <a href="#">\$: Self-Reference p104</a> • <a href="#">~. Nub - p107</a> • <a href="#">Monad-Dyad p115</a> • <a href="#">@. Agenda p154</a> • <a href="#">C. Cycle-Direct - Permute p167</a>                                                   |
| semicolon(s) | <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| shaped       | <a href="#">. Determinant • Dot Product p112</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| shapes       | <a href="#">B. Verbs p64</a> • <a href="#">&gt; Open - Larger Than p80</a> • <a href="#">-: Halve - Match p94</a> • <a href="#">. Determinant • Dot Product p112</a> • <a href="#">{ Catalogue - From p138</a>                                                                                                                                                                                                                                                                                                                             |
| significance | <a href="#">Intro 5. Forks p6</a> • <a href="#">Intro 15. Defined Adverbs p16</a> • <a href="#">* Signum - Times p89</a> • <a href="#">Foreign conjunction p214</a>                                                                                                                                                                                                                                                                                                                                                                        |
| signum       | <a href="#">Vocabulary p74</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">* Signum - Times p89</a>                                                                                                                                                                                                                                                                                                                                                                                                                        |
| sinh         | <a href="#">Intro 18. Explicit Definition p19</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">o. Pi Times - Circle Function p182</a> • <a href="#">t: Weighted Taylor p192</a>                                                                                                                                                                                                                                                                                                                           |
| slope(s)     | <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">Vocabulary p74</a> • <a href="#">D. Derivative p169</a> • <a href="#">D: Secant Slope p170</a>                                                                                                                                                                                                                                                                                                                                                                             |
| sorted       | <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">\$. Sparse p103</a> • <a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a> • <a href="#">s: Symbol p188</a>                                                                                                                                                                                                                                                                                                                                |
| sorting      | <a href="#">Sample 12. Sorting p43</a> • <a href="#">a. Alphabet p162</a> • <a href="#">s: Symbol p188</a>                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| sorts        | <a href="#">/: Grade Up - Sort p132</a> • <a href="#">\: Grade Down - Sort p135</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| sparse       | <a href="#">Vocabulary p74</a> • <a href="#">\$. Sparse p103</a> • <a href="#">C. Cycle-Direct - Permute p167</a> • <a href="#">3!: Conversions p218</a> • <a href="#">Special Code p229</a>                                                                                                                                                                                                                                                                                                                                               |

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqrt      | <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">=. Is (Local) p76</a> • <a href="#">@ Atop p153</a> • <a href="#">o. Pi Times - Circle Function p182</a> • <a href="#">q: Prime Factors - Prime Exponents p186</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| square(s) | <a href="#">Intro 5. Forks p6</a> • <a href="#">Intro 7. Bond Conjunction p8</a> • <a href="#">Intro 8. Atop Conjunction p9</a> • <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">Intro 11. Power and Inverse p12</a> • <a href="#">Intro 26. Linear Functions p27</a> • <a href="#">Intro 27. Obverse and Under p28</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">Sample 2. Alphabet and Numbers p33</a> • <a href="#">Sample 18. Geometry p49</a> • <a href="#">C. Adverbs and Conjunctions p65</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">Vocabulary p74</a> • <a href="#">&gt;: Increment - Larger Or Equal p82</a> • <a href="#">+ Conjugate - Plus p86</a> • <a href="#">+: Double • Not-Or p88</a> • <a href="#">*: Square - Not-And p91</a> • <a href="#">%: Square Root - Root p97</a> • <a href="#">^ Exponential - Power p98</a> • <a href="#">. Determinant • Dot Product p112</a> • <a href="#">@ Atop p153</a> • <a href="#">D. Derivative p169</a> • <a href="#">p. Polynomial p183</a> • <a href="#">q: Prime Factors - Prime Exponents p186</a> • <a href="#">Constants p199</a> • <a href="#">128!: Miscellaneous p228</a> • <a href="#">Special Code p229</a> |
| squared   | <a href="#">Constants p199</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| squaring  | <a href="#">while. p211</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| stderr    | <a href="#">1!: Files p216</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| stdin     | <a href="#">1!: Files p216</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| stdout    | <a href="#">1!: Files p216</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

## T

|    |                                                                                                                                                                                                                                                                                                                                                                                                |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| t. | <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">Vocabulary p74</a> • <a href="#">p.. Poly. Deriv. - Poly. Integral p184</a> • <a href="#">t. Taylor Coefficient p190</a> • <a href="#">t. Taylor Coefficient p191</a> • <a href="#">t: Weighted Taylor p192</a> • <a href="#">T. Taylor Approximation p193</a> |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                |                                                                                                                                                                                                                                                                                                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T.             | <a href="#">Vocabulary p74</a> • <a href="#">^: Power v p101</a> • <a href="#">T. Taylor Approximation p193</a>                                                                                                                                                                                                                                                         |
| t:             | <a href="#">Vocabulary p74</a> • <a href="#">t: Weighted Taylor p192</a>                                                                                                                                                                                                                                                                                                |
| tacit          | <a href="#">Introduction p1</a> • <a href="#">Intro 19. Tacit Equivalents p20</a> • <a href="#">: Explicit / p114</a> • <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                 |
| tacitly        | <a href="#">Intro 19. Tacit Equivalents p20</a>                                                                                                                                                                                                                                                                                                                         |
| tally          | <a href="#">Intro 20. Rank p21</a> • <a href="#">Vocabulary p74</a> • <a href="#"># Tally - Copy p124</a> • <a href="#">6!: Time p221</a>                                                                                                                                                                                                                               |
| tangent        | <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                  |
| tanh           | <a href="#">o. Pi Times - Circle Function p182</a>                                                                                                                                                                                                                                                                                                                      |
| tautologies    | <a href="#">Intro 28. Identity Functions and Neutrals p29</a> • <a href="#">0!: Scripts p215</a>                                                                                                                                                                                                                                                                        |
| tautology      | <a href="#">Intro 5. Forks p6</a> • <a href="#">Intro 28. Identity Functions and Neutrals p29</a>                                                                                                                                                                                                                                                                       |
| taylor         | <a href="#">Sample 25. Polynomials in Terms of Roots p56</a> • <a href="#">Sample 27. Polynomial Roots II p58</a> • <a href="#">Vocabulary p74</a> • <a href="#">^: Power v p101</a> • <a href="#">t. Taylor Coefficient p190</a> • <a href="#">t. Taylor Coefficient p191</a> • <a href="#">t: Weighted Taylor p192</a> • <a href="#">T. Taylor Approximation p193</a> |
| teacher(s)     | <a href="#">Intro 4. Punctuation p5</a> • <a href="#">Intro 9. Vocabulary p10</a>                                                                                                                                                                                                                                                                                       |
| teaching       | <a href="#">Dictionary p60</a>                                                                                                                                                                                                                                                                                                                                          |
| technique(s)   | <a href="#">Intro 9. Vocabulary p10</a>                                                                                                                                                                                                                                                                                                                                 |
| tensor         | <a href="#">/ Insert - Table p130</a> • <a href="#">C. Cycle-Direct - Permute p167</a>                                                                                                                                                                                                                                                                                  |
| tesselation(s) | <a href="#">:. Cut p122</a>                                                                                                                                                                                                                                                                                                                                             |
| theorem(s)     | <a href="#">Intro 28. Identity Functions and Neutrals p29</a>                                                                                                                                                                                                                                                                                                           |
| tie(s)         | <a href="#">Intro 21. Gerund and Agenda p22</a> • <a href="#">Vocabulary p74</a> • <a href="#">` Tie (Gerund) p151</a>                                                                                                                                                                                                                                                  |
| tilde          | <a href="#">Intro 9. Vocabulary p10</a> • <a href="#">13!: Debug p225</a>                                                                                                                                                                                                                                                                                               |
| tiling         | <a href="#">&lt;. Floor - Lesser Of (Min) p78</a>                                                                                                                                                                                                                                                                                                                       |



|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tolerant          | <a href="#">D. Comparatives p66</a> • <a href="#">G. Extended and Rational Arithmeti p69</a> • <a href="#">= Self-Classify - Equal p75</a> • <a href="#">&lt;. Floor - Lesser Of (Min) p78</a> • <a href="#">&gt; Open - Larger Than p80</a> • <a href="#">&gt;. Ceiling - Larger of (Max) p81</a> • <a href="#">* Signum - Times p89</a> • <a href="#">-: Halve - Match p94</a> • <a href="#">  Magnitude - Residue p109</a> • <a href="#">e. Raze In - Member (In) p171</a> • <a href="#">i. Integers - Index Of p175</a> • <a href="#">x: Extended Precision p197</a> |
| tolerantly        | <a href="#">&lt; Box - Less Than p77</a> • <a href="#">&gt; Open - Larger Than p80</a> • <a href="#">&gt;: Increment - Larger Or Equal p82</a> • <a href="#">~. Nub - p107</a> • <a href="#">~: Nub Sieve - Not-Equal p108</a>                                                                                                                                                                                                                                                                                                                                           |
| train(s)          | <a href="#">Intro 15. Defined Adverbs p16</a> • <a href="#">Intro 24. Trains p25</a> • <a href="#">Sample 13. Compositions I p44</a> • <a href="#">Sample 14. Compositions II p45</a> • <a href="#">E. Parsing and Execution p67</a> • <a href="#">F. Trains p68</a> • <a href="#">[: Cap p137</a> • <a href="#">`: Evoke Gerund p152</a> • <a href="#">@ Atop p153</a> • <a href="#">@. Agenda p154</a>                                                                                                                                                                 |
| transform(s)      | <a href="#">Sample 26. Polynomial Roots I p57</a> • <a href="#">^ Exponential - Power p98</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| transformation(s) | <a href="#">Sample 28. Polynomials: Stopes p59</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| translatability   | <a href="#">&lt;. Floor - Lesser Of (Min) p78</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| translate(s)      | <a href="#">Intro 12. Reading and Writing p13</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| translating       | <a href="#">Intro 12. Reading and Writing p13</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| translation(s)    | <a href="#">Intro 19. Tacit Equivalents p20</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| transpose(s)      | <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Vocabulary p74</a> • <a href="#">\$. Sparse p103</a> • <a href="#"> : Transpose p111</a> • <a href="#">.. Even p113</a> • <a href="#">A. Anagram Index - Anagram p163</a>                                                                                                                                                                                                                                                                                                                                    |
| transposed        | <a href="#">Intro 3. Verbs and Adverbs p4</a> • <a href="#">Sample 20. Directed Graphs p51</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| trident(s)        | <a href="#">Intro 4. Punctuation p5</a> • <a href="#">E. Parsing and Execution p67</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## U

u. [Vocabulary p74](#) • [. Determinant](#) • [Dot Product p112](#) • [: Explicit / p114](#) • [m. n. Explicit Noun Args p180](#) • [u. v. Explicit Verb Args p194](#) • [x. y. Explicit Arguments p196](#)

u: [Vocabulary p74](#) • [^: Power p100](#) • [u: Unicode p195](#)

## V

v. [Vocabulary p74](#) • [. Determinant](#) • [Dot Product p112](#) • [: Explicit / p114](#) • [m. n. Explicit Noun Args p180](#) • [u. v. Explicit Verb Args p194](#) • [x. y. Explicit Arguments p196](#)

valence(s) [Intro 2. Ambivalence p3](#) • [C. Adverbs and Conjunctions p65](#) • [Monad-Dyad p115](#) • [\[: Cap p137](#) • [5!: Representation p220](#) • [6!: Time p221](#)

verb(s) [Introduction p1](#) • [Intro 3. Verbs and Adverbs p4](#) • [Intro 4. Punctuation p5](#) • [Intro 5. Forks p6](#) • [Intro 6. Programs p7](#) • [Intro 7. Bond Conjunction p8](#) • [Intro 8. Atop Conjunction p9](#) • [Intro 10. Housekeeping p11](#) • [Intro 12. Reading and Writing p13](#) • [Intro 13. Format p14](#) • [Intro 14. Partitions p15](#) • [Intro 15. Defined Adverbs p16](#) • [Intro 17. Names and Displays p18](#) • [Intro 18. Explicit Definition p19](#) • [Intro 19. Tacit Equivalents p20](#) • [Intro 21. Gerund and Agenda p22](#) • [Intro 22. Recursion p23](#) • [Intro 24. Trains p25](#) • [Intro 27. Obverse and Under p28](#) • [Sample 3. Grammar p34](#) • [Sample 4. Function Tables p35](#) • [Sample 13. Compositions I p44](#) • [Sample 14. Compositions II p45](#) • [II. Grammar p62](#) • [A. Nouns p63](#) • [B. Verbs p64](#) • [C. Adverbs and Conjunctions p65](#) • [E. Parsing and Execution p67](#) • [F. Trains p68](#) • [G. Extended and Rational Arithmeti p69](#) • [H. Frets and Scripts p70](#) • [I. Locatives p71](#) • [J. Errors and Suspensions p72](#) • [III. Definitions p73](#) • [Vocabulary p74](#) • [≡ Self-Classify - Equal p75](#) • [< Box - Less Than p77](#) • [<: Decrement - Less Or Equal p79](#) • [-: Halve - Match p94](#) (only first 40 listed)

## W

weighted [Vocabulary p74](#) • [#. Base 2 - Base p125](#) • [p. Polynomial p183](#) •  
[t: Weighted Taylor p192](#)

whilst [: Explicit / p114](#) • [Control structures p200](#) • [break. p202](#) •  
[continue. p203](#) • [while. p211](#)

## X

x. [Intro 16. Word Formation p17](#) • [Intro 17. Names and Displays p18](#) • [Intro 18. Explicit Definition p19](#) • [Intro 24. Trains p25](#) •  
[Intro 29. Secondaries p30](#) • [Sample 5. Bordering a Table p36](#) •  
[Sample 11. Classification II p42](#) • [Sample 26. Polynomial Roots I p57](#) • [Sample 27. Polynomial Roots II p58](#) • [Sample 28. Polynomials: Stopes p59](#) • [H. Frets and Scripts p70](#) • [J. Errors and Suspensions p72](#) • [Vocabulary p74](#) • [~. Nub - p107](#) • [: Explicit / p114](#) • [{:: Map - Fetch p141](#) • [D. Derivative p169](#) • [m. n. Explicit Noun Args p180](#) • [u. v. Explicit Verb Args p194](#) • [x. y. Explicit Arguments p196](#) • [assert. p201](#) • [for. p204](#) • [if. p206](#) • [13!: Debug p225](#)

x: [G. Extended and Rational Arithmeti p69](#) • [Vocabulary p74](#) • [% Reciprocal - Divide p95](#) • [^ Exponential - Power p98](#) • [^: Power p100](#) • [x: Extended Precision p197](#)

## Y

y. [Intro 16. Word Formation p17](#) • [Intro 17. Names and Displays p18](#) • [Intro 18. Explicit Definition p19](#) • [Intro 19. Tacit Equivalents p20](#) • [Intro 23. Iteration p24](#) • [Intro 24. Trains p25](#) • [Intro 29. Secondaries p30](#) • [Sample 28. Polynomials: Stopes p59](#) • [H. Frets and Scripts p70](#) • [I. Locatives p71](#) • [J. Errors and Suspensions p72](#) • [Vocabulary p74](#) • [^: Power p100](#) • [^: Power v p101](#) • [: Explicit / p114](#) • [{:: Map - Fetch p141](#) • [C. Cycle-Direct - Permute p167](#) • [m. n. Explicit Noun Args p180](#) • [t. Taylor Coefficient p190](#) • [T. Taylor Approximation p193](#) • [u. v. Explicit Verb Args p194](#) • [x. y. Explicit Arguments p196](#) • [assert. p201](#) • [for. p204](#) • [goto\\_name. p205](#) • [if. p206](#) • [return. p207](#) • [select. p208](#) • [throw. p209](#) • [try. p210](#) • [5!:](#) [Representation p220](#) • [6!:](#) [Time p221](#) • [7!:](#) [Space p222](#) • [13!:](#) [Debug p225](#)

year(s) [Intro 2. Ambivalence p3](#) • [Intro 20. Rank p21](#) • [\\$. Sparse p103](#)

yearly [Intro 20. Rank p21](#)

yield [Sample 11. Classification II p42](#) • [^ Exponential - Power p98](#) • [:. Cut p122](#)

## Z

zeros [Sample 25. Polynomials in Terms of Roots p56](#) • [Sample 26. Polynomial Roots I p57](#) • [\\$ Shape Of - Shape p102](#) • [\\$. Sparse p103](#) • [{. Head - Take p139](#)