APL2 IDIOMS Library

Stan Cason
IBM
Endicott, NY

APL2 is a rich and extraordinary language used by many people. The basics of the language are easy to grasp but, learning the subtleties of the language can take years.

Most people are entranced with the power of APL2, but have a hard time thinking in terms of Arrays. Even experts in APL2 sometimes have trouble remembering algorithms they have not used recently. What is needed is a library of "APL2 phrases" that can be made available to the programmer to solve a variety of common application problems.

In order to speed up the learning process of APL2, APL2 Phrases was developed. With over 650 distinct APL2 phrases, sorted into 24 general categories, APL2 Phrases represents a fairly complete list of solutions to common application problems. By having a single repository for APL2 phrases, many of us can take advantage of algorithms that others have developed.

The idioms themselves may be freely distributed.

I am indebted to many fellow IBMers for their help in generating the IDIOM list. Thanks to Ed Kellerman and Rich Hartmann for their help in coding the system. Thanks to Alan Graham, Ram Biyani, Ray Polivka, John Lindley, Dick Dunbar, Gary Logan, Dan Milch, Dick Conner, Dave Macklin, Karl Soop, Tuong Tran, and Ray Trimble for their contributions of idioms and ideas. I am also indebted to The Finnish APL Association for their Pocket Library. Their library provided insight to some of the common problems facing programmers.

To get a hardcopy of the FINNAPL Pocket Library, contact the Finnish APL Association:

FinnAPL
P.O. Box 1005
SF-00101 Helsinki 10
Finland

Note: The information in this document was extracted from IBM Technical Report 01.A845 and the IDIOMS workspace that is included in Workstation APL2. The technical report was copyrighted by IBM in 1989. The workspace was copyrighted by IBM in 2002.

CONVENTIONS

Some phrases are written in both index origins. If the algorithm is "origin dependent", the index origin 0 version is listed first.

Short names are used so they can be easily converted to be compatible with your own code.

RANK	TYPE	USE
S - Scalar or one item vector O - One item vector V - Vector M - Matrix A - Array	B - Boolean C - Character F - Floating Point I - Integer N - Numeric Z - Complex	G - Graded or Grouped L - Lengths P - Positions U - Unique

V (vector) is implied unless otherwise specified. Other characters, W, X, Y, etc., are used to differentiate between two or more variables within an idiom that are otherwise the same. Examples:

Name	Description	Name	Description
A,AX,AY	General arrays	IM	Integer matrices
BM	Boolean matrices	N,NX,NY	Numeric vectors
BS	Boolean scalars	PAV	Position array of vectors
CA	Character arrays		
C,CX,CY	Character vectors	PS	Position scalars
GAF	Graded array of floating points	UM	Unique matrices
		VM	Vector of matrices
GI	Graded integer vectors	VV	Vector of vectors
GM	Graded Matrix	V,X,Y	General vectors

There are also a few global variables that are assumed within the idioms.

NAME	CONTENTS	DEFINITION
ALP	ABCDEFGHIJKLMNOPQRSTUVWXYZ	Upper case alphabet
ALT	abcdefghijklmnopqrstuvwxyz	Alternate alphabet
LOW	abcdefghijklmnopqrstuvwxyz	Lower case alphabet
NUM	0123456789	10 numerals
SEQ	QWERTYUIOPASDFGHJKLZXCVBNM	Arb seq of alphabet
LCT	⎕AF(⍳256)+32×⎕AV∊ALP	⎕AV with ALP to LOW
UCT	⎕AF(⍳256)-32×⎕AV∊LOW	⎕AV with LOW to ALP

Assignment Algorithms

```
⍎(0=⎕NC 'A')/'A←1'                                ⍝ Assign value to A if not assigned.
⍎(1=≡A)/'A←⊂A'                                    ⍝ Change A to scalar if it is simple.
X←↑(' '∨.≠V)↓X V←⍞                                ⍝ Change X if new value given.
M←0 IS⍴''                                         ⍝ Initialize a matrix with no rows.
N←B←I←F←Z←⍳0                                      ⍝ Initialize variables to the empty vector.
(N B I F Z)←VV                                    ⍝ Splitting a variable into a set of variables.
VV←N B I F Z                                      ⍝ Joining a set of variables into one variable.
V←1↓⍎'0 ',V                                       ⍝ Execute which works on empty vector.
V←1↓⍎¨CSV,VV                                      ⍝ Execute each with prototype of CSV.
⍎¨⊂[1]MX,'←',⍕MY                                  ⍝ Assigns MY-values to matrix of MX-names.
⍎¨⊂[2]MX,'←',⍕MY                                  ⍝ Assigns MY-values to matrix of MX-names.
V←1 2 3 4 5 6 7 8 9 0,⎕                           ⍝ Input continuation.
⎕←A←2 10⍴⍳20                                      ⍝ Output assigned value.
VA←⍎¨I⍴'⍞'                                        ⍝ Quick input of (×/I) strings. I=⍴VA
V←(⍴V)↓↑⍞ V←⍞←'Finished? '                        ⍝ Prompt and response on same line.
FS←⌊/⍳0                                           ⍝ Largest possible number.
FS←⌈/⍳0                                           ⍝ Smallest possible number.
BA←?I⍴2                                           ⍝ I random boolean numbers.
BA←¯1+?I⍴2                                        ⍝ I random boolean numbers.
I←1+IS?IS                                         ⍝ Random numbers between 1-IS w/o repl.
I←IS?IS                                           ⍝ Random numbers between 1-IS w/o repl.
I←1+?IS⍴IS                                        ⍝ Random numbers between 1-IS w/repl.
I←?IS⍴IS                                          ⍝ Random numbers between 1-IS w/repl.
(0 0⍉M)←1                                         ⍝ Reassign main diagonal of matrix.
(1 1⍉M)←1                                         ⍝ Reassign main diagonal of matrix.
NS←V[(0≠⍴V)?⍴V]                                   ⍝ Select random item from vector. Works on ''.
Z←BS⊃X Y                                          ⍝ Select X or Y depending on BS.
Z←(BS+1)⊃X Y                                      ⍝ Select X or Y depending on BS.
Z←↑BS↓X Y                                         ⍝ Select X or Y depending on BS.
A←(⊂¯1 0 1⍳×NA)⌷SW SX SY                          ⍝ Selection depending on sign of array.
S←(¯1 0 1⍳×NS)⊃SW SX SY                           ⍝ Selection depending on sign of scalar.
A←PA⊃¨⊂A                                          ⍝ Chipmunk.  Selective picking from array.
AV←PAV⊃¨¨⊂⊂A                                      ⍝ Selective multiple subarrays from array.
(,A)[(⍴A)⊥⍉PM]←V                                  ⍝ Scatter assignment. (⍴PM)=(⍴V),⍴⍴A
(,A)[1+(⍴A)⊥⍉¯1+PM]←V                             ⍝ Scatter assignment. (⍴PM)=(⍴V),⍴⍴A
(PV⌷¨⊂A)←V                                        ⍝ Scatter assignment. (⍴PV)=⍴V. (⍴¨PV)=⍴⍴A
V←PV⌷¨⊂A                                          ⍝ Scatter indexing. (⍴PV)=⍴V. (⍴¨PV)=⍴⍴A
X←'line1',0⍴Y←'line2'                             ⍝ Pornography. Combining two lines into one.
X←↑'line1' Y←'line2'                              ⍝ Pornography. Combining two lines into one.
CM←'|',('¯',[0]CM,[0]'_'),'|'                     ⍝ Framing CM in a box.
CM←'|',('¯',[1]CM,[1]'_'),'|'                     ⍝ Framing CM in a box.
```

Boolean Selection Algorithms

```
B←¯1↓1,∨/MG≠1⊖MG                 ⍝ Boolean first item of each change in MG.
B←1↓(∨/MG≠¯1⊖MG),1               ⍝ Boolean last item of each change in MG.
B←(,(1-L),[.5]1)/1               ⍝ Boolean ending changes given # duped items.
B←(,(1-L),[1.5]1)/1              ⍝ Boolean ending changes given # duped items.
B←(1+⍳+/L)∊+\L                   ⍝ Boolean ending changes given # duped items.
B←(⍳+/L)∊+\L                     ⍝ Boolean ending changes given # duped items.
B←(,1,[.5]-L)/1                  ⍝ Boolean L[i] gaps after each one.
B←(,1,[1.5]-L)/1                 ⍝ Boolean L[i] gaps after each one.
B←(⍳+/L)∊0,+\L                   ⍝ Boolean starting changes given # duped items.
B←(⍳+/L)∊+\1,L                   ⍝ Boolean starting changes given # duped items.
B←(,1,[.5]1-L)/1                 ⍝ Boolean start changes given length vector L.
B←(,1,[1.5]1-L)/1                ⍝ Boolean start changes given length vector L.
B←(⍳+/L)∊+\0,L                   ⍝ Boolean start changes given length vector L.
B←(⍳+/L)∊+\1,L                   ⍝ Boolean start changes given length vector L.
B←(↑⍴M)↑(⍳⌈/P)∊P                 ⍝ Boolean start vector given position indices.
B←2</0,(1+B)/B                   ⍝ Boolean expand length for headers.
B←~X∊Y                           ⍝ Boolean items in X that are not in Y.
B←M∧.=S                          ⍝ Boolean rows of M all equal to scalar S.
B←∨/C⍷CM                         ⍝ Boolean rows of CM containing C.
BM←↑∨/VC⍷¨⊂CM                    ⍝ Boolean mask of CM containing VC.
B←∨/MX∧.=⍉MY                     ⍝ Boolean rows of MX containing MY.
B←∨/MVX∧.(≡¨)⍉MVY                ⍝ Boolean rows of MVX containing MVY.
B←,1↑[0]C⍷CM                     ⍝ Boolean rows of CM starting with C.
B←,1↑[1]C⍷CM                     ⍝ Boolean rows of CM starting with C.
BA←A≡¨⊂V                         ⍝ Item equals.  Find item V in array A.
(,BA)←<\,BA←C⍷CA                 ⍝ Boolean one at first occurrence of C in CA.
B←CM∧.∊C                         ⍝ Does each row contain only items from C?
B←0∧.=↑¨0⍴¨M                     ⍝ Which rows of M are all numeric?
B←((×IS)×⍴B)↑IS↓B                ⍝ Shift B forward or backward IS positions.
```

Boolean Tests General Algorithms

```
BS←V∧.=↑V                        ⍝ Boolean test: Are all items in V equal? 1=≡V
BS←V∧.∊⊂↑V                       ⍝ Boolean test: Are all items in V equal?
BS←V≡1⌽V                         ⍝ Boolean test: Are all items in V equal?
BS←AX≡AY                         ⍝ Boolean test: Is AX identical to AY?
BS←(V⍳V)≡⍳⍴V                     ⍝ Boolean test: Are all items of V unique?
BS←>/⍒⎕AF⊃CX CY                  ⍝ Is CX lexically less than CY?
BS←</⍋⎕AF⊃CX CY                  ⍝ Is CX lexically less than or equal to CY?
BS←>/⍋⎕AF⊃CX CY                  ⍝ Is CX lexically greater than CY?
BS←</⍒⎕AF⊃CX CY                  ⍝ Is CX lexically greater than or equal to CY?
BS←∧/,Y∊X                        ⍝ Boolean test: Is Y a subset of X?
BO←NS>⍴V                         ⍝ Does vector V have less than NS items?
B←N>⍴A                           ⍝ Are there less than N items in each dim?
BS←A≡⍕A                          ⍝ Boolean test: Is A a simple character array?
BS←1<≡A                          ⍝ Boolean test: Is A a nested array?
BS←1≥≡A                          ⍝ Boolean test: Is A a simple array?
BS←0∊⍴⍴A                         ⍝ Boolean test: Is A a scalar?
BS←1∊⍴⍴A                         ⍝ Boolean test: Is A a vector?
BS←0∊⍴A                          ⍝ Boolean test: Is A empty?
BS←0≠⍴,A                         ⍝ Boolean test: Is A non-empty?
BS←A≡⍉A                          ⍝ Boolean test: Is A symmetric?
BS←A≡-⍉A                         ⍝ Boolean test: Is A anti-symmetric?
BS←∧/(↑V∊C),V∊NUM,'¯_',C←ALP,ALT,'∆⍙' ⍝ Boolean test: Is V a valid APL name?
BS←¯1≠⎕NC V                      ⍝ Boolean test: Is V a valid APL name?
```

Boolean Tests Numeric Algorithms

```
BS←≠/0 1∊B                       ⍝ Boolean test: All items in simple B equal?
BS←≠/0 1∊∊AB                     ⍝ Boolean test: All items in B equal?
BS←AB≡1⌽AB                       ⍝ Boolean test: All items in B equal?
BS←∧/B                           ⍝ Boolean test: Are all true?
BS←∨/B                           ⍝ Boolean test: Are any true?
BS←~∨/B                          ⍝ Boolean test: Are none true?
BS←≠/B                           ⍝ Boolean test: Parity.
BS←∧/∊B∊0 1                      ⍝ Boolean test: Is B boolean?
BS←(↑0⍴⊂A)≡(⍴A)⍴0                ⍝ Boolean test: Is A simple numeric?
BS←0=↑0⍴A                        ⍝ Boolean test: Is A numeric? (if homogeneous)
BS←~×↑⍒N                         ⍝ Boolean test: Is first item largest?
BS←|/⍋N                          ⍝ Boolean test: Is first item largest?
BS←~×↑⍋N                         ⍝ Boolean test: Is first item smallest?
BS←|/⍒N                          ⍝ Boolean test: Is first item smallest?
BS←∨/~2|N                        ⍝ Boolean test: Is any element of N even?
BS←∧/2|N                         ⍝ Boolean test: Is every element of N odd?
BS←∧/N>0                         ⍝ Boolean test: Is every element of N positive?
BS←∧⌿N=⌈⍀N                       ⍝ Boolean test: Is N in ascending column order.
BS←∧/N=⌈\N                       ⍝ Boolean test: Is N in ascending row order.
BS←N[⍋N]≡1+⍳⍴N                   ⍝ Boolean test: Is N permutation vector?
BS←N[⍋N]≡⍳⍴N                     ⍝ Boolean test: Is N permutation vector?
BS←N[⍋N]≡NX[⍋NX]                 ⍝ Boolean test: Is N permutation of NX?
BS←∧/∊~2|NA                      ⍝ Boolean test: Is every element of NA even?
BS←∧/∊NA=⌊NA                     ⍝ Boolean test: Is every element of NA integer?
BS←0=+/0=((2×IS≠2),3+2×⍳⌊.5×IS⋆.5)|IS ⍝ Boolean test: Is IS prime?
BS←0=+/0=((2×IS≠2),1+2×⍳⌊.5×IS⋆.5)|IS ⍝ Boolean test: Is IS prime?
B←</(N<⌈N),N∘.<XY                ⍝ Is N an integer in range [XY) XY ↔ lo,hi.
B←((↑XY)<N)∧N<↑⌽XY               ⍝ Is N in range (XY)  XY ↔ lo,hi.
B←</N∘.<XY                       ⍝ Is N in range [XY)  XY ↔ lo,hi.
B←>/N∘.>XY                       ⍝ Is N in range (XY]  XY ↔ lo,hi.
B←((↑XY)≤N)∧N≤↑⌽XY               ⍝ Is N in range [XY]  XY ↔ lo,hi.
BS←0≠.=N                         ⍝ Ain't Dot Is.  Test for even # of non-zeros.
BS←↑≡/X Y fn Y X                 ⍝ Test for commutativity of fn.
BS←((V f1 X)f2 Y)≡V f1 X f2 Y    ⍝ Test for associativity of f1 and f2.
BS←((V f1 X)f2 Y)≡↑f1/V X f2⊂Y   ⍝ Test for distributivity of f1 and f2.
```

Computational Algorithms

```
NS←+/N                           ⍝ Sum of N.
NS←-/N                           ⍝ Alternating sum of N.
NS←×/N                           ⍝ Product of N.
NS←÷/N                           ⍝ Alternating product of N.
NS←+/|N                          ⍝ Sum of magnitude of N.
NS←-/|N                          ⍝ Alternating sum of magnitude of N.
N←+\N                            ⍝ Cumulative sums.
N←NS+/N                          ⍝ Running sum of NS consecutive elements of N.
N←¯2-/0,N                        ⍝ Inverse of +\.  Difference of adjacent pairs.
N←↑-//NS 1/¯2 N                  ⍝ NS differences of differences of adjacents.
N←-\⍳IS                          ⍝ Alternating series of length IS(1,-1,2,-2..).
I←+\1+⍳IS                        ⍝ First IS triangular numbers.
I←+\⍳IS                          ⍝ First IS triangular numbers.
I←+\+\1+⍳IS                      ⍝ First IS figurative numbers.
I←+\+\⍳IS                        ⍝ First IS figurative numbers.
NA←AX÷AY+AY=0                    ⍝ Division.  Avoid DOMAIN ERROR for N÷0.
NA←AX×÷AY                        ⍝ Division.  Force DOMAIN ERROR for 0÷0.
N←100×NM÷[1]+⌿NM                 ⍝ Col-wise percentage per column.
N←100×NM÷[2]+⌿NM                 ⍝ Col-wise percentage per column.
N←100×NM÷[0]+/NM                 ⍝ Row-wise percentage per row.
N←100×NM÷[1]+/NM                 ⍝ Row-wise percentage per row.
NA←NA×|NA                        ⍝ Square without changing sign.
NM←NM+[1]N                       ⍝ Add vector N to each column of NM.
NM←NM+[2]N                       ⍝ Add vector N to each column of NM.
NM←NM×[0]N                       ⍝ Multiply each row of NM by vector N.
NM←NM×[1]N                       ⍝ Multiply each row of NM by vector N.
NS←÷+/÷N                         ⍝ Ohm's Law - resistance of parallel resistors.
NS←-/×⌿0 1⌽M                     ⍝ Evaluating a two row determinant.
NS←-/+/¨×⌿¨(1 ¯1×⊂0 1 2)⌽¨⊂M     ⍝ Evaluating a three row determinant.
M←IS IS⍴1,IS⍴0                   ⍝ Identity matrix: IS by IS.
M←↑⌹/0⍴⊂IS⍴0                     ⍝ Identity matrix: IS by IS.
M←(⍳IS)∘.=⍳IS                    ⍝ Identity matrix: IS by IS.
M←(1+⍳IS)∘.×1+⍳IS                ⍝ Multiplication table: IS by IS.
M←(⍳IS)∘.×⍳IS                    ⍝ Multiplication table: IS by IS.
M←(⍳IS)∘.>⍳IS                    ⍝ Lower triangular matrix: IS by IS.
M←X∘.×Y                          ⍝ Outer product.
M←MX+.×MY                        ⍝ Matrix product.
A←AX,.×AY                        ⍝ Mid product of AX and AY.
I←(2=+⌿0=I∘.|I)/I←1+⍳IS          ⍝ Prime numbers from 1...IS.
I←(2=+⌿0=I∘.|I)/I←⍳IS            ⍝ Prime numbers from 1...IS.
IS←⌈/(∧/0=V∘.|I)/V←1+⍳⌊/I        ⍝ Greatest common divisor of vector I.
IS←⌈/(∧/0=V∘.|I)/V←⍳⌊/I          ⍝ Greatest common divisor of vector I.
M←0=(1+⍳⌈/I)∘.|I                 ⍝ Table of divisibility.
M←0=(⍳⌈/I)∘.|I                   ⍝ Table of divisibility.
NS←+/∊NA                         ⍝ Sum of all elements in NA.
I←(0=I|IS)/I←2+⍳⌊IS÷2            ⍝ All factors of IS.
I←(0=I|IS)/I←1+⍳⌊IS÷2            ⍝ All factors of IS.
FS←+/F[⍒|F]                      ⍝ Accurately sum a vector of floating numbers.
FA←NA⍟A                          ⍝ Find the exponent of NA such that NA⋆FA = A.
```

Conversion Algorithms

```
CA←⎕AF(⎕AF CA)-64×CA∊ALF         ⍝ Convert to lower case for EBCDIC.
CA←⎕AF(⎕AF CA)+32×CA∊ALF         ⍝ Convert to lower case for ASCII.
CA←⎕AF(⎕AF CA)+64×CA∊LOW         ⍝ Convert to upper case for EBCDIC.
CA←⎕AF(⎕AF CA)-32×CA∊LOW         ⍝ Convert to upper case for ASCII.
CA←LCT[⎕AF CA]                   ⍝ Convert to lower case. LCT ↔ ⎕AV w/low/up.
CA←LCT[1+⎕AF CA]                 ⍝ Convert to lower case. LCT ↔ ⎕AV w/low/up.
CA←UCT[⎕AF CA]                   ⍝ Convert to upper case. UCT ↔ ⎕AV w/up/low.
CA←UCT[1+⎕AF CA]                 ⍝ Convert to upper case. UCT ↔ ⎕AV w/up/low.
CA←'FDCBA'[+/IA∘.≥10×6 7 8 9]    ⍝ Students grades given score IA.
CA←'FDCBA'[+/IA∘.≥10×0 6 7 8 9]  ⍝ Students grades given score IA.
I←+/I×¯1⋆I<1⌽I←0,(,(10⋆⍳4)∘.×1 5)['IVXLCDM'⍳C]  ⍝ Roman numerals to Arabic.
I←+/I×¯1⋆I<1⌽I←0,(,(1,10⋆⍳3)∘.×1 5)['IVXLCDM'⍳C]⍝ Roman numerals to Arabic.
IA←(¯1⌽⍳1+⍴⍴NA)⍉((1+⌊IS⍟1⌈⌈/,NA)⍴IS)⊤NA ⍝ Base IS representation of a number.
IA←(¯1⌽⍳1+⍴⍴NA)⍉((1+⌊10⍟1⌈⌈/,NA)⍴10)⊤NA ⍝ Base 10 representation of a number.
NA←(¯1⌽⍳1+⍴⍴FA)⍉⌊(IS⍴NS)⊤(NS⋆IS)×1|FA ⍝ IS place-base NS rep. of a fraction.
H←'0123456789ABCDEF'[,⍉16 16⊤⎕AF C] ⍝ REXX C2X.  Convert character to hex.
H←'0123456789ABCDEF'[1+,⍉16 16⊤⎕AF C] ⍝ REXX C2X.  Convert character to hex.
H←'0123456789ABCDEF'[,⍉((1+⌊16⍟1⌈⌈/,N)⍴16)⊤N] ⍝ REXX D2X.  Decimal to hex.
H←'0123456789ABCDEF'[1+,⍉((1+⌊16⍟1⌈⌈/,N)⍴16)⊤N] ⍝ REXX D2X.  Decimal to hex.
NA←⊃⍎¨⊂[¯1+⍴⍴CA]',',CA           ⍝ Convert non-empty CA to NA - rank ≥1.
NA←⊃⍎¨⊂[⍴⍴CA]',',CA              ⍝ Convert non-empty CA to NA - rank ≥1.
NM←⊃⍎¨⊂[1]',',CM                 ⍝ Convert non-empty CM to numeric vector.
NM←⊃⍎¨⊂[2]',',CM                 ⍝ Convert non-empty CM to numeric vector.
N←1↓⍎'0',',',CM                  ⍝ Convert character matrix to numeric vector.
I←1↓⍎'0 ',(C∊' 0123456789')/C    ⍝ Convert to numeric, throw out characters.
I←10⊥'0123456789'⍳C              ⍝ Convert character to numeric.
I←10⊥¯1+'0123456789'⍳C           ⍝ Convert character to numeric.
I←10⊥⍉M                          ⍝ Convert rows of digits to base 10.
I←⍎¨C                            ⍝ Convert character vector to vector of digits.
I←16⊥¨(⌈.5×(2|⍴H)+1+⍳⍴H)⊂'0123456789ABCDEF'⍳H ⍝ REXX X2D routine.  Hex to Dec.
I←16⊥¨(⌈.5×(2|⍴H)+⍳⍴H)⊂16|'123456789ABCDEF'⍳H ⍝ REXX X2D routine.  Hex to Dec.
C←∊4↑¨('FEC80124936DA5B7'⍳H)⌽¨⊂'1111000010011010' ⍝ Convert hex to binary char.
C←∊4↑¨('FEC80124936DA5B7'⍳H)⌽¨⊂'0111100001001101' ⍝ Convert hex to binary char.
BM←⍉((1+⌊2⍟1⌈⌈/I)⍴2)⊤I           ⍝ Convert integer to binary.
IS←2⊥B                           ⍝ Convert binary to integer.
C←⎕AF 2⊥¨((⌈.125×⍳⍴B)⊂B          ⍝ Convert binary to character.
B←,⍉(8⍴2)⊤⎕AF C                  ⍝ Convert character to binary.
I←⎕AF C                          ⍝ Convert character to EBCDIC/ASCII positions.
C←16 16⍴⎕AV                      ⍝ EBCDIC/ASCII sequence in HEX table.
C←⎕AF⍉(4⍴256)⊤I+(256⋆4)×I<0      ⍝ Convert integers to double words.
I←(256⊥⍉I)-(256⋆4)×128≤,1↑[1]I←⎕AF C⍝ Convert double words to integer.
I←(256⊥⍉I)-(256⋆4)×128≤,1↑[2]I←⎕AF C⍝ Convert double words to integer.
M←(256⊥1↓[0]N)×(×128-M)×16⋆¯63⌈¯70+128|M←1↑[0]N ⍝ Convert halfword to float.
M←(256⊥1↓[1]N)×(×128-M)×16⋆¯63⌈¯70+128|M←1↑[1]N ⍝ Convert halfword to float.
C←⎕AF 16⊥¨(⌈.5×(2|⍴H)+1+⍳⍴H)⊂'0123456789ABCDEF'⍳H ⍝ REXX X2C routine.  Hex/Char.
C←⎕AF 16⊥¨(⌈.5×(2|⍴H)+⍳⍴H)⊂16|'123456789ABCDEF'⍳H ⍝ REXX X2C routine.  Hex/Char.
```

Date and Time Algorithms

```
IS←0 100 100⊥3↑⎕TS              ⍝ Joining current date.
(IW IX IY)←0 100 100⊤IS          ⍝ Separating date IS - YYYYMMDD format.
B←0≠.=400 100 4∘.|IS             ⍝ Is IS (YYYY) a leap year?
C←'06:06:05'⍕3↑3↓⎕TS             ⍝ Current time - HH:MM:SS.
C←'56/06/0005'⍕⎕TS[1ϕι3]         ⍝ Current US date - MM/DD/YYYY.
C←'56/06/0005 06:06:05'⍕⎕TS[(1ϕι3),3+ι3] ⍝ Current US date and time.
C←'56/06/0005'⍕⎕TS[ϕι3]          ⍝ Current European date - DD/MM/YYYY.
C←'56/06/0005 06:06:05'⍕⎕TS[(ϕι3),3+ι3] ⍝ Current European date and time.
IS←D+(M⊃0 0,+\30+1 ¯2,∊5 4ρ¨⊂1 0)+(M>2)∧0=4|Y ⍝ Julian day (DDD) given Y M D.
IS←D+(M⊃0,+\30+1 ¯2,∊5 4ρ¨⊂1_0)+(M>2)∧0=4|Y ⍝ Julian day (DDD) given Y M D.
IS←(1000×Y)+D+(M⊃0 0,+\30+1 ¯2,∊5 4ρ¨⊂1 0)+(M>2)∧0=4|Y ⍝ Julian date (YYYYDDD).
IS←(1000×Y)+D+(M⊃0,+\30+1 ¯2,∊5 4ρ¨⊂1 0)+(M>2)∧0=4|Y ⍝ Julian date (YYYYDDD).
IS←7|6+IS+-/⌈IS÷4 100 400        ⍝ Weekday (S-S:0-6) of first of year IS (YYYY).
IS←7|+/D(⍋M⊃'0032503514624'),⌊5 1.25×4 100⊤Y-3>M⍝ Weekday (S-S:0-6) given Y M D.
IS←7|+/D(⍋M⊃'032503514624'),⌊5 1.25×4 100⊤Y-3>M ⍝ Weekday (S-S:0-6) given Y M D.
I←1+0 100⊥0 12⊤(0 12⊥0 100⊤IS)-2+ϕιNS ⍝ NS months before date IS (YYYYMM).
I←1+0 100⊥0 12⊤(0 12⊥0 100⊤IS)-1+ϕιNS ⍝ NS months before date IS (YYYYMM).
I←1+0 100⊥0 12⊤(0 12⊥0 100⊤IS)+ιNS ⍝ NS months after date IS (YYYYMM).
I←1+0 100⊥0 12⊤(0 12⊥0 100⊤IS)-1-ιNS ⍝ NS months after date IS (YYYYMM).
IS←0 12⊥0 100⊤IS                 ⍝ IS months from "0" given IS (YYYYMM) date.
IS←1+0 100⊥0 12⊤IS-1             ⍝ Date IS (YYYYMM) given IS months from "0".
```

External Name Routine Algorithms

```
V←('W'=↑¨7⊃¨8↑¨∆F¨(⊂'⋆ ⋆ '),¨ALP)/ALP ⍝ File modes of R/W disks.
V←('W'=↑¨8⊃¨8↑¨∆F¨(⊂'⋆ ⋆ '),¨ALP)/ALP ⍝ File modes of R/W disks.
V←('R'=↑¨7⊃¨8↑¨∆F¨(⊂'⋆ ⋆ '),¨ALP)/ALP ⍝ File modes of R/O disks.
V←('R'=↑¨8⊃¨8↑¨∆F¨(⊂'⋆ ⋆ '),¨ALP)/ALP ⍝ File modes of R/O disks.
V←('E'=↑¨7⊃¨8↑¨∆F¨(⊂'⋆ ⋆ '),¨ALP)/ALP ⍝ File modes of disk extensions.
V←('E'=↑¨8⊃¨8↑¨∆F¨(⊂'⋆ ⋆ '),¨ALP)/ALP ⍝ File modes of disk extensions.
M←∆FM FILEID                     ⍝ Read data from fixed file into matrix M.
S←M ∆FM FILEID                   ⍝ Write M or VV to a fixed file.
VV←∆FV FILEID                    ⍝ Read data from variable file into VV.
S←VV ∆FV FILEID                  ⍝ Write M or VV to a variable file.
R←('EXIT ',V,'(ARG(1))')∆EXEC C  ⍝ Perform REXX built-in function V(C).
VV←∆FV 'A A A3',0ρM ∆FV 'A A A3' ⍝ Reversing disclose.
VV←(C≠' ')SAN C                  ⍝ Sentence to vector of words, keep blanks.
VV←' ' DAN C                     ⍝ Sentence to vector of words, drop blanks.
VV←(C≠' ')CAN C                  ⍝ Sentence to vector of words, drop blanks.
VV←' ' DAN∊' ',PR AA             ⍝ Nested AA to vector of words. (See 11-34)
CM←⊃CS DAN V                     ⍝ Vector to matrix at selected character.
C←('(B1 1 ',(⍕ρB),')')ATR B      ⍝ Convert binary to character.
B←('(B1 1 ',(⍕8×ρC),')')RTA C    ⍝ Convert character to binary.
I←('(B8 1 ',(⍕ρC),')')RTA C      ⍝ Convert character to 1 byte integer.
I←('(I2 1 ',(⍕.5×ρC),')')RTA C   ⍝ Convert character to 2 byte integer.
I←('(I4 1 ',(⍕.25×ρC),')')RTA C  ⍝ Convert character to 4 byte integer.
C←('(B8 1 ',(⍕ρI),')')ATR I      ⍝ Convert 1 byte integer to character.
C←('(I2 1 ',(⍕ρI),')')ATR I      ⍝ Convert 2 byte integer to character.
C←('(I4 1 ',(⍕ρI),')')ATR I      ⍝ Convert 4 byte integer to character.
F←('(E4 1 ',(⍕.25×ρC),')')RTA C  ⍝ Convert character to floating point.
C←('(E4 1 ',(⍕ρF),')')ATR F      ⍝ Convert floating point to character.
NA←CTN CA                        ⍝ Convert character array to numeric array.
IS←2⊥'AIBJE' 'C'∨.∊¨⊂PFA A       ⍝ Type of A.  1-char, 2-num, 3-mixed.
```

Financial Algorithms

```
A←(÷1+FS)⊥⌽NA                    ⍝ Present value of cash flows NA at int FS.
A←(1+FS)⊥NA                      ⍝ Future value of cash flows NA at int FS.
A←FA÷⍉1-(1+FA)∘.×-IA             ⍝ Annuity coefficient: IA periods at int FA.
A←NA∘.×(1+FA)∘.⋆IA               ⍝ Compound interest: IA prds, FA int, NA prn.
```

Formatting Algorithms

```
IA←1+⌊10⍟1⌈NA                     ⍝ Field width for integral part of number.
IA←+⌿NA≠⌊NA←(10⋆⍳NS)∘.×NA         ⍝ Field width ≤NS of fractional part of number.
IA←+⌿NA≠⌊NA←(10⋆-1-⍳NS)∘.×NA      ⍝ Field width ≤NS of fractional part of number.
CM←1 0⍕10 10⊤⍳NS                  ⍝ Create col header CM for NS wide text.
⎕FC[3]←'⋆'                        ⍝ Fills format overflow with '⋆'.
⎕FC[4]←'⋆'                        ⍝ Fills format overflow with '⋆'.
CM←⍕⊂1/AA                         ⍝ From nested to simple char image.
CM←I↓(-I←(-2-2>⍴⍴AA)↑≡AA)↓⍕⊂1/AA ⍝ Nested to simple char image w/o extra blanks.
CM←⍕⊃(C≠' ')⊂¨C←⊂[1]CM            ⍝ Columnize rows of data separated by blanks.
CM←⍕⊃(C≠' ')⊂¨C←⊂[2]CM            ⍝ Columnize rows of data separated by blanks.
CM←1↓[0]⍕0,[0]AA                  ⍝ Format and right justify columns of report.
CM←1↓[1]⍕0,[1]AA                  ⍝ Format and right justify columns of report.
CM←1↓[0](NS,0)⍕0,[0]AA            ⍝ Format and right justify NS wide columns.
CM←1↓[1](NS,0)⍕0,[1]AA            ⍝ Format and right justify NS wide columns.
M←(⍳↑⍴M),M                        ⍝ Attach row numbers to a matrix.
,['']VV                           ⍝ Display vector of vectors vertically.
CM←('∧-',B),[0]B,'|',B∘.∧B←0 1   ⍝ Truth table: All possibilities of and(∧).
CM←('∧-',B),[1]B,'|',B∘.∧B←0 1   ⍝ Truth table: All possibilities of and(∧).
CM←('∨-',B),[0]B,'|',B∘.∨B←0 1   ⍝ Truth table: All possibilities of or(∨).
CM←('∨-',B),[1]B,'|',B∘.∨B←0 1   ⍝ Truth table: All possibilities of or(∨).
CM←('⍲-',B),[0]B,'|',B∘.⍲B←0 1   ⍝ Truth table: All possibilities of nand(⍲).
CM←('⍲-',B),[1]B,'|',B∘.⍲B←0 1   ⍝ Truth table: All possibilities of nand(⍲).
CM←('⍱-',B),[0]B,'|',B∘.⍱B←0 1   ⍝ Truth table: All possibilities of nor(⍱).
CM←('⍱-',B),[1]B,'|',B∘.⍱B←0 1   ⍝ Truth table: All possibilities of nor(⍱).
```

Function Algorithms

```
C←(¯1≠⎕NC,[' ']⎕AV)/⎕AV          ⍝ All valid one character APL2 names.
CM←(¯1≠⎕NC C)⌿C←'⎕',,[' ']ALP    ⍝ All valid two character ⎕ names.
CM←(¯1≠⎕NC C)⌿C←'⎕',ALP[⍉(NSρ26)⊤⍳26⋆NS] ⍝ All valid NS-character ⎕ names.
CM←(¯1≠⎕NC C)⌿C←'⎕',ALP[1+⍉(NSρ26)⊤⍳26⋆NS] ⍝ All valid NS-character ⎕ names.
A0←⎕IO+A      ⍝ A1=A+1   A0=A      ⍝ Change ⎕IO dependant argument.
A1←A+⎕IO-1    ⍝  A1=A   A0=A-1     ⍝ Change ⎕IO dependant argument.
A0←-⎕IO-A     ⍝ A1=A-1   A0=A      ⍝ Change ⎕IO dependant result.
A1←A+~⎕IO     ⍝  A1=A   A0=A+1     ⍝ Change ⎕IO dependant result.
((~,∧\('⍝'≠CM)∨≠\CM='''')/,CM)←' '⍝ Decommenting the ⎕CR of a function.
PFK←12+¯12|PFK                   ⍝ Keep PFK within range 1-12.
VV←(⊂[1]⎕NL 3 4)~¨' '            ⍝ List of functions and operators without ' '.
VV←(⊂[2]⎕NL 3 4)~¨' '            ⍝ List of functions and operators without ' '.
M←,[' ']⎕CR¨⊂[1]⎕NL 3 4          ⍝ Quick list of all functions and operators.
M←,[' ']⎕CR¨⊂[2]⎕NL 3 4          ⍝ Quick list of all functions and operators.
IS←+/↑¨ρ¨⎕CR¨⊂[1]⎕NL 3 4         ⍝ The number of code lines in a workspace.
IS←+/↑¨ρ¨⎕CR¨⊂[2]⎕NL 3 4         ⍝ The number of code lines in a workspace.
IS←(+/↑¨ρ¨⎕CR¨⊂[1]M)÷↑ρM←⎕NL 3 4 ⍝ The average # of lines per pgm. in a WS.
IS←(+/↑¨ρ¨⎕CR¨⊂[2]M)÷↑ρM←⎕NL 3 4 ⍝ The average # of lines per pgm. in a WS.
CM←(1∊¨(⊂C)⍷¨⎕CR¨⊂[1]CM)⌿CM←⎕NL 3⍝ Find functions that contain string C.
CM←(1∊¨(⊂C)⍷¨⎕CR¨⊂[2]CM)⌿CM←⎕NL 3⍝ Find functions that contain string C.
IS←↑ρ⎕NL 2 3 4                   ⍝ The number of objects in a workspace.
CM←'⎕EM' ⎕EC 'expression'        ⍝ Simulate error and continue.
'⎕ES ⎕ET' ⎕EA 'expression'       ⍝ Do-or-die error checking.
A←¯1↑⎕EC 'expression'            ⍝ Capture result of expression or error MSG.
A←⎕EC¨,/(⊂'SΔ'),(⎕NL 3 4),⊂'←1'  ⍝ Put stop control on unlocked objects.
A←⎕EC¨,/(⊂'TΔ'),(⎕NL 3 4),⊂'←⍳99'⍝ Put trace control on unlocked objects.
A←⎕EC¨,/(⊂'SΔ'),(⎕NL 3 4),⊂'←⍳0' ⍝ Remove stop control from all objects.
A←⎕EC¨,/(⊂'TΔ'),(⎕NL 3 4),⊂'←⍳0' ⍝ Remove trace control from all objects.
SΔfn←1+BS↑⎕LC                    ⍝ Stop function "fn" on next line if BS true.
→B/I                             ⍝ Branch to line in I of first true B.
→PS⊃I                            ⍝ Branch to line in position PS of I.
IS:→(100<NS←NS+1)/IS             ⍝ Branch to label on condition.
→IS+0~B                          ⍝ IF. Branch on condition B.
→IS×1~B                          ⍝ IF NOT. Branch if condition B false.
→⎕LC+IS                          ⍝ Branch to offset IS from current line.
→0                               ⍝ RETURN.  Leave function, return to caller.
→                                ⍝ EXIT. Leave all levels of program.
A←⍎PS⊃VV                         ⍝ Execute statement PS in VV of statements.
A←⍎↑BS↓'else part' 'then part'   ⍝ If Then/Else.
V←⎕FX(⊂C),2 ⎕TF¨⊂[1]⎕NL 2        ⍝ Create a function C to recreate all vars.
V←⎕FX(⊂C),2 ⎕TF¨⊂[2]⎕NL 2        ⍝ Create a function C to recreate all vars.
M←¯1⌽¯1⊖↑∨/O⍷¨⊂M                 ⍝ Life: next generation given O<=>140 3x3 wins.
((,B)/,A)←fn(,B)/,A              ⍝ WHERE.  Execute "fn" on condition B mask.
fn¨BS/⊂A                         ⍝ Conditional execution of monadic function.
⍎'Z←X lo ',((1<≡Y)/'PR¨'),'Y'    ⍝ Z←X(lo PR)Y.  Perform "lo" as scalar.
Z←X lo↑Y ⎕IO←B                   ⍝ Z←X(lo IO B)Y;⎕IO.  Run "lo" in origin B.
Z←⊃[I](⊂[I]AX)lo⊂AY              ⍝ Z←AX(lo OAX I)AY.  Run "lo" on axes I of AX.
Z←⊃[I](⊂AX)lo⊂[I]AY              ⍝ Z←AX(lo OAY I)AY.  Run "lo" on axes I of AY.
```

Manipulating Characters Algorithms

```
M←MU[¯1++\B;]                    ⍝ Replicate MU given boolean start vector.
M←MU[+\B;]                       ⍝ Replicate MU given boolean start vector.
M←M[⌈\B×⍳⍴B;]                    ⍝ Replace rows of M given boolean start vector.
V←L/V                            ⍝ Duplicate items in vector V, L times.
M←LS⌿,[¯.5]V                     ⍝ Duplicate vector V, LS times.
M←LS⌿,[.5]V                      ⍝ Duplicate vector V, LS times.
V←(LS×⍴V)⍴V                      ⍝ Duplicate vector V, LS times.
V←(,L∘.≥⍳⌈/L)\V                  ⍝ Expand V given length vector L.
V←(V⍳↑1↓C)↑V←(1+V⍳↑C)↓V          ⍝ Keep everything from ↑C to ↑1↓C in V.
V←(¯1+V⍳↑1↓C)↑V←(V⍳↑C)↓V         ⍝ Keep everything from ↑C to ↑1↓C in V.
V←∊V,¨⊂NS⍴S                      ⍝ Insert NS items S after each item of V.
V←∊(⊂NS⍴S),¨V                    ⍝ Insert NS items S before each item of V.
V←(V,X)[⍋(⍳⍴V),P]                ⍝ Insert X after positions P in V.  (⍴P)=⍴X
V←¯1↓(,⌽1,∨\CS≠⌽CM)/,CM,CS       ⍝ Matrix to vector at character CS.
V←Y[⍋Y∊X]                        ⍝ Move items X to end of Y.
V←∊(-L+1)↑¨V                     ⍝ Open gaps before each item of V, L wide.
V←∊(L+1)↑¨V                      ⍝ Open gaps between each item in V, L wide.
V←((~B)-B\L)/V                   ⍝ Open gaps between points B in V, L wide.
V←∊NS↑¨V                         ⍝ Open NS-1 spaces between each item in V.
M←,['']V                         ⍝ One column matrix from vector V.
M←,[¯.5]V                        ⍝ One row matrix from vector V.
M←,[.5]V                         ⍝ One row matrix from vector V.
M←X,[.5]Y                        ⍝ Two column matrix from two vectors.
M←X,[1.5]Y                       ⍝ Two column matrix from two vectors.
M←((⌈.5×⍴V),2)⍴V                 ⍝ Two column matrix from one vector.
M←X,[¯.5]Y                       ⍝ Two row matrix from two vectors.
M←X,[.5]Y                        ⍝ Two row matrix from two vectors.
M←⊃[0]V W X Y                    ⍝ N column matrix from N vectors.
M←⊃[1]V W X Y                    ⍝ N column matrix from N vectors.
M←⍉⊃V W X Y                      ⍝ N column matrix from N vectors.
M←⊃V W X Y                       ⍝ N row matrix from N vectors.
U←(¯1↓1,V≠1⌽V)/V                 ⍝ Unique.  Drop duplicates from ordered vector.
U←((V⍳V)=⍳⍴V)/V                  ⍝ Unique.  Drop duplicates from vector.
U←(∨⌿<\V∘.=V)/V                  ⍝ Unique.  Drop duplicates from vector.
MU←(¯1↓1,∨/MG≠1⊖MG)⌿MG           ⍝ Unique.  Drop duplicates from ordered list.
MU←(∨⌿<\M∧.=⍉M)⌿M                ⍝ Unique.  Drop duplicates from list.
```

Manipulating Numbers Algorithms

```
B←<\B                            ⍝ All zeros except the first one.
B←≤\B                            ⍝ All ones after the first zero.
B←∨\B                            ⍝ All ones after the first one.
B←∧\B                            ⍝ All ones to the first zero.
B←+/∧\B                          ⍝ Count of leading ones.
B←B∨≠\B                          ⍝ Parity+connectors.  Connect odd & even ones.
B←≠\B                            ⍝ Parity.  Connect odd and even ones.
B←2≠/0,B                         ⍝ Gray code or reflected binary. Inverse of ≠\.
B←2</0,B                         ⍝ Boolean first ones in each group of ones.
B←2>/B,0                         ⍝ Boolean last ones in each group of ones.
B←1⌽B                            ⍝ Boolean start vector to boolean end vector.
B←I/(⍴I)⍴1 0                     ⍝ Alternating sequence of I ones and zeros.
B←IS/L⍴1 0                       ⍝ L sequences of IS ones and zeros.
L←¯2-/P,1+⍴B                     ⍝ Length vector given first position indices.
L←¯2-/0,P                        ⍝ Length vector given last position indices.
L←+/U∘.≡V                        ⍝ Length vector given unique items in V.
L←(1↓P,1+⍴B)-P←B/⍳⍴B             ⍝ Length vector given boolean vector B.
NA←+/∧\CA=' '                    ⍝ Number of leading blanks.
NA←+/∧\⌽CA=' '                   ⍝ Number of trailing blanks.
N←+/¨(+\B)⊂N                     ⍝ Add subvectors of N given B breaks in group.
N←+/¨(L/1+⍳⍴L)⊂N                 ⍝ Add subvectors of N given L items per group.
N←+/¨(+\¯1↓1,G≠1⌽G)⊂N            ⍝ Add subvectors of N from consecutive G dups.
N←+/¨(+\¯1↓1,∨/MG≠1⊖MG)⊂N        ⍝ Add subvectors of N using ordered list MG.
N←N+.×V∘.≡U                      ⍝ Sum by bucket. ⍴N = ⍴V. U = buckets.
N←N×.⋆V∘.≡U                      ⍝ Product by bucket. ⍴N = ⍴V. U = buckets.
A←AX+.∊Y                         ⍝ Count of the number of Ys in each row of AX.
I←+\(C='(')-¯1↓0,C=')'           ⍝ Depth of parenthesis.
VN←M,.⍳V                         ⍝ Position(s) of V in each row of M.
N←M⌊.⍳V                          ⍝ Position of V in corresponding row of M.
B←(<\'/⋆'⍷CA)∨⌽<\'/⋆'⍷⌽CA        ⍝ Position of comment in each row of array CA.
B←≠\B\2≠/0,(B←BX∨BY)/BX          ⍝ State of switch given BX=on & BY=off spikes.
N←(~B)+B\N                       ⍝ Expand N, but change fill item to one.
N←(NS×~B)+B\N                    ⍝ Expand N, but change fill item to NS.
IS←|¯11|+/+\10↑⍎¨C               ⍝ ISBN check digit generator from C.  C∧.∊NUM
IS←97+¯97|IS                     ⍝ SWIFT check digit from IS bank number.
```

Numeric Range Algorithms

```
N←⌈/NM                           ⍝ Maximum value of NM.
N←⌈/|NM                          ⍝ Maximum of magnitude of NM.
N←⌈/NM,0                         ⍝ Maximum of positive value of NM.
N←N××.5-(⌈/NM)≠N←⌈/|NM           ⍝ Maximum of magnitude of NM preserving sign.
N←⌊/NM                           ⍝ Minimum value of NM.
N←⌊/|NM                          ⍝ Minimum of magnitude of NM.
N←⌊/NM,0                         ⍝ Maximum of negative value of NM.
N←N××.5-(⌊/NM)≠N←⌊/|NM           ⍝ Minimum of magnitude of NM preserving sign.
IS←↑⍒N                           ⍝ Index of the largest item.
IO←Nι⌈/N                         ⍝ Index of the largest item.
IS←↑⍋N                           ⍝ Index of the smallest item.
IO←Nι⌊/N                         ⍝ Index of the smallest item.
FA←(×NA)|NA                      ⍝ Fractional part of number with sign.
FA←1||NA                         ⍝ Magnitude of fractional part of number.
FA←1|NA                          ⍝ Fractional part of number.
NA←|NA                           ⍝ Magnitude.  Absolute Value of NA.
FA←0 1⊤NA                        ⍝ Integral+fractional part of positive number.
IA←(×NA)×⌊|NA                    ⍝ INTEGER.  Truncate to whole number.
FA←(×NA)×(⌊.5+|NA÷NS)×NS         ⍝ Rounding to nearest NSth.
IA←(×NA)×⌊.5+|NA                 ⍝ Rounding to nearest whole number.
IA←(×NA)×⌊(1≤2||NA)+|NA          ⍝ Rounding to nearest even number.
FA←⍎(|IS)⍕NA                     ⍝ Rounding to IS decimal places.
N←N×N≤NS                         ⍝ Force to 0 any N greater than NS.
N←N×NS≤N                         ⍝ Force to 0 any N less than NS.
N←(>/N∘.>0 NS)/N                 ⍝ Keep everything in range [0,NS).
N←0⌈NS⌊N                         ⍝ Force N numbers to range 0≤N≤NS.
N←N×¯1⋆B                         ⍝ Change sign on condition B.
NA←(⊂1 ¯1)×NA                    ⍝ Plus Minus. Number and its negative.
N←X+NS×ιIS                       ⍝ Arithmetic progression vector.
N←X+NS×¯1+ιIS                    ⍝ Arithmetic progression vector.
N←X+(×N)×ι1+⌊|N←Y-X              ⍝ Index Generator.  Range from X to Y.
N←X+0,(×N)×ι⌊|N←Y-X              ⍝ Index Generator.  Range from X to Y.
N←X+IS×ι0⌈(IS≠0)+⌊(Y-X)÷IS       ⍝ Index Generator with step IS. From X to Y.
N←X+IS×¯1+ι0⌈(IS≠0)+⌊|(Y-X)÷IS   ⍝ Index Generator with step IS. From X to Y.
N←∊NX+ι¨IX                       ⍝ Sequence from NX for IX items.
N←¯1+∊NX+ι¨IX                    ⍝ Sequence from NX for IX items.
NA←(×NAX)×(|NAY)||NAX            ⍝ REMAINDER from division of NAX by NAY.
NA←NAY|NAX                       ⍝ MODULO of NAX and NAY.
NA←(×NAY)×|NAX                   ⍝ SIGN. Transfer of sign from NAY to NAX.
NA←NS+(-NS)|NA                   ⍝ Residue replacing 0 with NS.
```

Numerical Geometry Algorithms

```
ZA←AX×¯12○○AY÷180                ⍝ From magnitude AX and degrees AY to complex.
ZA←AX×¯12○AY                     ⍝ From magnitude AX and radians AY to complex.
NA←|0J1⊥⊖A                       ⍝ Get magnitude of A. 2=↑⍴A (x,y) pairs
NA←|AX+¯11○AY                    ⍝ Get magnitude of real AX and imaginary AY.
NA←12○0J1⊥⊖A                     ⍝ Get angle in radians of A. 2=↑⍴A (x,y) pairs
NA←12○AX+¯11○AY                  ⍝ Get angle (rad) of real AX and imaginary AY.
NA←(180÷○1)×12○0J1⊥⊖A            ⍝ Get angle in degrees of A. 2=↑⍴A (x,y) pairs
NA←(180÷○1)×12○AX+¯11○AY         ⍝ Get angle (deg) of real AX and imaginary AY.
NA←9 11∘.○ZA                     ⍝ Split complex array into real & imaginary.
ZA←AX+¯11○AY                     ⍝ Join XA real and YA imaginary to complex.
ZA←(9○ZAX)+¯11○11○ZAY            ⍝ Real from ZAX and imaginary from ZAX.
ZA←¯11○+ZA                       ⍝ Swap real & imaginary.
S←+/|2-/Z                        ⍝ Length of polygon.
ZA←ZA+6J9                        ⍝ Move figure by x=6, y=9.
ZA←ZA×1D30                       ⍝ Rotate figure 30 degrees.
ZA←ZA××ZS                        ⍝ Rotate figure in direction of point ZS.
ZA←ZA×2.1                        ⍝ Isometric scaling by 2.1.
ZA←ZA×2.1D30                     ⍝ Rotate & scale at same time.
ZA←(2,¯11○3)+.×9 11∘.○ZA         ⍝ Rectangular scale by x=2,y=3.
ZA←ZA+¯11○.3×9○ZA                ⍝ Skew by 30 percent in y.
ZA←+ZA                           ⍝ Mirror in X.
ZA←-+ZA                          ⍝ Mirror in Y.
ZA←(¯10+~BA)○ZA                  ⍝ Mirror in X if B.
VZA←,¨ZA                         ⍝ Pinpoint figure rather than polygon.
Z←⋆○0J2×(⍳NS+1)÷NS               ⍝ Regular unit polygon of NS edges.
Z←⋆○0J2×(0,⍳NS)÷NS               ⍝ Regular unit polygon of NS edges.
VZ←0,¨⋆○0J2×(1+⍳NS)÷NS           ⍝ NS spokes of unit wheel.
VZ←0,¨⋆○0J2×(⍳NS)÷NS             ⍝ NS spokes of unit wheel.
V←×11○(ZS-1↓Z)×+2-/Z             ⍝ Sign of point ZS relative to edges of Z.
Z←(⍳5)∘.+¯11○⍳10                 ⍝ All pixels in a 5 by 10 window.
Z←V+¯11○f V                      ⍝ Plot of scalaroid function 'f' for data V.
Z←0J1⊥1 0⌽2/⊃V(⍳1+⍴V)            ⍝ Outline of bar chart of data V.
Z←0J1⊥1 0⌽2/0,⊃V(⍳⍴V)            ⍝ Outline of bar chart of data V.
Z←¯3↓,Z-[0]((2-/Z÷3),0)∘.×0 1,(1+1D60),2 ⍝ Koch island new generation.
Z←¯3↓,Z-[1]((2-/Z÷3),0)∘.×0 1,(1+1D60),2 ⍝ Koch island new generation.
VZ←(⊂ZX)+(⊂ZY-ZX)×(⍳NS+1)÷NS     ⍝ Cascade NS-fold fill between two polygons.
VZ←(⊂ZX)+(⊂ZY-ZX)×(0,⍳NS)÷NS     ⍝ Cascade NS-fold fill between two polygons.
M←2 2⊤1+⍳4                       ⍝ Unit square.
M←2 2⊤⍳4                         ⍝ Unit square.
M←2 2 2⊤1+⍳8                     ⍝ Unit cube.
M←2 2 2⊤⍳8                       ⍝ Unit cube.
M←2 2 2 2⊤1+⍳16                  ⍝ Unit tesseract.
M←2 2 2 2⊤⍳16                    ⍝ Unit tesseract.
Z←0J1⊥M[1 0;]                    ⍝ Parallel projection of 3D object in M.
Z←0J1⊥M[2 1;]                    ⍝ Parallel projection of 3D object in M.
Z←0J1⊥M[1 0;]×D÷D-M[2;]          ⍝ Perspective projection from distance D.
Z←0J1⊥M[2 1;]×D÷D-M[3;]          ⍝ Perspective projection from distance D.
VZ←(¯.5 .5×NS)+⊂Z                ⍝ Stereo pair.  (Eye separation NS)
Z←0J1⊥0 1⌽2/|⌊/M,[0.5]-M←11 9∘.○Z⍝ Window enclosing Z.
Z←0J1⊥0 1⌽2/|⌊/M,[1.5]-M←11 9∘.○Z⍝ Window enclosing Z.
Z←(,[⍳0](1+⍳NS)÷NS)⊥M+.×Z        ⍝ NS-point spline.(M=Bezier matrix, Z ctrl pts)
Z←(,[⍳0](⍳NS)÷NS)⊥M+.×Z          ⍝ NS-point spline.(M=Bezier matrix, Z ctrl pts)
V←(×/(+/X÷2)-0,X)⋆.5             ⍝ Area of a triangle given side length. 3=⍴X
NS←|.5×+/Y×(¯1⌽X)-1⌽X            ⍝ Area of a polygon given X,Y endpoints.
A←10 12∘.○ZA                     ⍝ From complex to magnitude and radians. 2=↑⍴A
AV←AX∘.,AY                       ⍝ Cartesian product: all pairs of AX, AY.
```

Selecting Positions Algorithms

```
P←(⊂[1]CM)⍳¨' '                 ⍝ Position of first blanks in rows of M.
P←(⊂[2]CM)⍳¨' '                 ⍝ Position of first blanks in rows of M.
P←(CM≠' ')⌈.×⍳¯1↑⍴CM            ⍝ Position of trailing blanks in rows
P←(⊂[1]MY)⍳⊂[1]MX               ⍝ Row positions of MX in MY.
P←(⊂[2]MY)⍳⊂[2]MX               ⍝ Row positions of MX in MY.
P←(<\MX∧.=⍉MY)+.×⍳↑⍴MY          ⍝ Row positions of MX in MY (0 for not found).
P←B/⍳⍴B                         ⍝ Positions of ones in boolean vector B.
P←(+/B)↑⍒B                      ⍝ Positions of ones in boolean vector B.
P←B/⍳↑⍴A                        ⍝ Row positions given boolean vector B.
VP←(,BA)/,↑∘.,/⍳¨⍴BA            ⍝ Vector of positions of ones in boolean array.
PM←⍉(⍴A)⊤P                      ⍝ Coordinates of A corresponding to offsets P.
PM←1+⍉(⍴A)⊤P-1                  ⍝ Coordinates of A corresponting to offsets P.
P←¯1↓+\0,L                      ⍝ Positions of ones given length vector L.
P←¯1↓+\1,L                      ⍝ Positions of ones given length vector L.
P←(<\~CM∊' ')+.×⍳¯1↑⍴CM         ⍝ Position of the first non-blank char by row.
P←(↑⌽⍴CM)-(1,CM=' ')⊥1          ⍝ Position of the last non-blank char by row.
P←1+(↑⌽⍴CM)-(1,CM=' ')⊥1        ⍝ Position of the last non-blank char by row.
PO←(C≠' ')⍳1                    ⍝ Position of the first non-blank char.
PO←(⍴C)-(1,C=' ')⊥1             ⍝ Position of the last non-blank char.
PO←1+(⍴C)-(1,C=' ')⊥1           ⍝ Position of the last non-blank char.
PO←B⍳1                          ⍝ Position of the first satisfied condition.
PO←(+\X≡¨Y)⍳NS                  ⍝ Position of the NSth Y in X.
PO←VV⍳⊂C                        ⍝ Position of first occurrence of C in VV.
PO←(⌽X)⍳Y                       ⍝ Position of last Y in X - from left.
PO←(⍴X)-(1,X≠Y)⊥1               ⍝ Position of last Y in X.
PO←1+(⍴X)-(1,X≠Y)⊥1             ⍝ Position of last Y in X.
P←(C⍷CX)/⍳⍴CX                   ⍝ Positions of start of C in string CX.
P←(CX∊C)/⍳⍴CX                   ⍝ Positions of items in set C in string CX.
P←↑(~CX∊C)/⍳⍴CX                 ⍝ Position of first item in CX not in C.
```

Sorting Algorithms

```
NM←NM[⍋NM;]                      ⍝ Sorting NM in ascending row order.
NM←NM[⍒NM;]                      ⍝ Sorting NM in descending row order.
NM←NM[⍋NM×I;]                    ⍝ Choosing sorting direction I +A, 0U, or -D.
CM←CM[⍋⎕AF CM;]                  ⍝ Sorting CM in ascending row order.
CM←CM[⍒⎕AF CM;]                  ⍝ Sorting CM in descending row order.
CM←CM[SEQ⍒CM;]                   ⍝ Sorting CM in reverse SEQ order.
CM←CM[SEQ⍋CM;]                   ⍝ Sorting CM in SEQ row order.
M←M[⍒L/L;]                       ⍝ Sort by highfliers - M ↔ groups of length L.
V←V[⍋⍋IV]                        ⍝ Mesh V according to mask pattern IV.
V[⍋B]←V←X,Y                      ⍝ Mesh X and Y in V using boolean pattern B.
V←∊X,¨Y                          ⍝ Merge X and Y alternately.
CVV←CVV[⍋⎕AF⊃CVV]                ⍝ Sorting CVV in alphabetical order.
IV←⍋⍋NA                          ⍝ IV is the ranking of NA in same order.
IV[I]←IV←⍳⍴I←⍋NA                 ⍝ IV is the ranking of NA in same order.(fast)
(,A)←(,A)[⍋,A+(↑⌽⍴A)/(⌈/,A)×⍳×/¯1↓⍴A] ⍝ Sort each row in ascending order.
(,⍉A)←(,A)[⍋,A+(⍴A)⍴(⌈/,A)×⍳×/¯1↓⍴A]  ⍝ Sort each column in ascending order.
A←⊃(⊂¨⍋¨A)▯¨A←⊂[¯1+⍴⍴A]A←1/A     ⍝ Sort each row in ascending order.
A←⊃(⊂¨⍋¨A)▯¨A←⊂[⍴⍴A]A←1/A        ⍝ Sort each row in ascending order.
A←⊃[I](⊂¨⍋¨A)▯¨A←⊂[I←¯2+⍴⍴A]A    ⍝ Sort each column in ascending order. 2≤⍴⍴A
A←⊃[I](⊂¨⍋¨A)▯¨A←⊂[I←¯1+⍴⍴A]A    ⍝ Sort each column in ascending order. 2≤⍴⍴A
V[⍋~B]←V←(B/X),(~B)/Y            ⍝ Mask Operator.  Merge X and Y using B.
V←(,BA)/,A                       ⍝ Pack an array into a vector based on BA.
((,BA)/,A)←V                     ⍝ Unpack a vector into an array based on BA.
```

Statistics Descriptive Algorithms

```
AVG←(+/N)÷1⌈ρN                   ⍝ Average (mean) of N.
AVG3←(3+/NA)÷3                   ⍝ Three wise rolling average.
CAVE←(+⌿NM)÷1⌈↑ρNM               ⍝ Column averages of NM.
CAVG←(+⌿NM)÷1⌈+⌿0≠NM             ⍝ Column averages of NM. (non-zero)
RAVE←(+/NM)÷1⌈↑⌽ρNM              ⍝ Row averages of NM.
RAVG←(+/NM)÷1⌈+/0≠NM             ⍝ Row averages of NM. (non-zero)
WAVG←(N+.×NM)÷+/N                ⍝ Weighted average of vector/matrix columns.
WAVG←(NM+.×N)÷+/N                ⍝ Weighted average of vector/matrix rows.
MODE←(I=⌈/I←+⌿N∘.=NU)/NU←(∨⌿<\N∘.=N)/N ⍝ Mode(s) of data.
MED←.5×+/N[(⍋N)[⌈.5×¯1 0+ρN←,N]] ⍝ Median of non-empty N.
MED←.5×+/N[(⍋N)[⌈.5×0 1+ρN←,N]]  ⍝ Median of non-empty N.
RANGE←(⌈/N)-⌊/N                  ⍝ Range of non-empty N.
STD←((+/(,A-(+/,A)÷N)⋆2)÷N←1⌈ρ,A)⋆.5⍝ Total theoretical standard deviation of A.
STD←((+/(,A-(+/,A)÷1⌈ρ,A)⋆2)÷1⌈¯1+ρ,A)⋆.5⍝ Total standard deviation of A.
STD←(((N×+/A⋆2)-(+/A)⋆2)⋆.5)÷N←1⌈↑⌽ρA ⍝ Row theoretical standard deviation of A.
STD←(((N×+/A⋆2)-(+/A)⋆2)÷N×1⌈¯1+N←1⌈↑⌽ρA)⋆.5⍝ Row standard deviation of A.
VAR←(+/(,A-(+/,A)÷N)⋆2)÷N←1⌈ρ,A  ⍝ Total theoretical variance of A.
VAR←(+/(,A-(+/,A)÷1⌈ρ,A)⋆2)÷1⌈¯1+ρ,A  ⍝ Total variance of A.
VAR←((N×+/A⋆2)-(+/A)⋆2)÷(N←1⌈↑⌽ρA)⋆2 ⍝ Row theoretical variance of A.
VAR←((N×+/A⋆2)-(+/A)⋆2)÷N×1⌈¯1+N←1⌈↑⌽ρA ⍝ Row variance of A.
V←X~Y                            ⍝ Difference of sets.  Elements of X not in Y.
V←(X∊Y)/X                        ⍝ Intersection of two sets of numbers.
V←Y~Y~X                          ⍝ Intersection of two sets of numbers.
V←Y,(~X∊Y)/X                     ⍝ Union of two sets of numbers.
V←Y,X~Y                          ⍝ Union of two sets of numbers.
N←+/X∘.=Y                        ⍝ Frequency of X in Y.
M←2|⌊(⍳2⋆IS)∘.÷1+2⋆IS-⍳IS        ⍝ Truth table with IS variables.
M←2|⌊(¯1+⍳2⋆IS)∘.÷2⋆IS-⍳IS       ⍝ Truth table with IS variables.
```

Statistics Distribution Algorithms

```
V←N+.×Y⌹N←X∘.⋆0 1                ⍝ Least squares linear fit given X,Y values.
C←Y⌹X∘.⋆⍳IS+1                    ⍝ IS degree polynomial fit given X,Y values.
C←Y⌹X∘.⋆(⍳IS+1)-1                ⍝ IS degree polynomial fit given X,Y values.
N←(,['']NA)⊥⌽N                   ⍝ Eval. asc. ord. N-coeff poly. at points NA.
N←(,['']NA)⊥N                    ⍝ Eval. dec. ord. N-coeff poly. at points NA.
C←K!N                            ⍝ Combinations of N things taken K ways.
M←⌽(K=+⌿M)/M←(Nρ2)⊤⍳1+2⊥N↑Kρ1    ⍝ Binary matrix of (N,K) combinations.
I←⍋N                             ⍝ Inverting a permutation.
NS←(!K)×K!N                      ⍝ Number of permutations of (N,K) combinations.
M←(<⌿M)/M←(2,IS⋆2)ρ(,⍉M),,M←IS ISρ1+⍳IS ⍝ All possible pairs of 1 through IS.
M←(<⌿M)/M←(2,IS⋆2)ρ(,⍉M),,M←IS ISρ⍳IS ⍝ All possible pairs of 1 through IS.
M←(∧⌿2<⌿M)/M←1+((-K)↑⍳N+1)⊤⍳(!K)×K!N ⍝ Numeric matrix of (N,K) combinations.
M←(∧⌿2<⌿M)/M←1+((-K)↑⍳N)⊤⍳(!K)×K!N ⍝ Numeric matrix of (N,K) combinations.
M←⍉(⍳1+IS)∘.!⍳1+IS               ⍝ Binomial coefficients from 1-IS.
M←⍉(0,⍳IS)∘.!0,⍳IS               ⍝ Binomial coefficients from 1-IS.
N←(N!X)×(Y⋆N)×(1-Y)⋆X-N←⍳X+1     ⍝ Binomial distribution of X trials at prob. Y.
N←(N!X)×(Y⋆N)×(1-Y)⋆X-N←¯1+⍳X+1  ⍝ Binomial distribution of X trials at prob. Y.
N←(⍳1+IS)!IS                     ⍝ Coefficients of the binomial.
N←(0,⍳IS)!IS                     ⍝ Coefficients of the binomial.
N←÷Y×(X-1)!Y×X-1                 ⍝ Beta function.
N←!N-1                           ⍝ Gamma function.
N←(⋆-Y)×(Y⋆X)÷!X                 ⍝ Poisson distribution of states X and Y avg.
M←÷1+(⍳IS)∘.+⍳IS                 ⍝ Hilbert matrix of order IS.
M←÷¯1+(⍳IS)∘.+⍳IS                ⍝ Hilbert matrix of order IS.
V←V∘.!V←⍳IS+1                    ⍝ Pascal's triangle of order IS.
V←V∘.!V←0,⍳IS                    ⍝ Pascal's triangle of order IS.
N←+/Y×(X⋆N)÷!N←⍳ρY               ⍝ Taylor series at point X, coefficients Y.
N←+/Y×(X⋆N)÷!N←¯1+⍳ρY            ⍝ Taylor series at point X, coefficients Y.
CA←' ⋆'[BA]                      ⍝ Plotting a curve from boolean values.
CA←' ⋆'[1+BA]                    ⍝ Plotting a curve from boolean values.
CM←⊃(⌊N)ρ¨'⋆'                    ⍝ Create a histogram from numeric vector.
```

Structural Algorithms

```
A←(' ',S)[BA]                    ⍝ Build array from boolean pattern.  Insert S.
A←(' ',S)[1+BA]                  ⍝ Build array from boolean pattern.  Insert S.
A←(' ',V)[BA×(ρBA)ρι¯1↑ρBA]      ⍝ Build array from boolean pattern.  Insert V.
A←(' ',V)[1+BA×(ρBA)ρι¯1↑ρBA]    ⍝ Build array from boolean pattern.  Insert V.
A←⊃⊃,/¨(⊂[1]BM)⊂¨⊂V              ⍝ Build array from boolean pattern.  Reduce A.
A←⊃⊃,/¨(⊂[2]BM)⊂¨⊂V              ⍝ Build array from boolean pattern.  Reduce A.
A←1/A                            ⍝ Change A, only if it is scalar, to vector.
A←1/¨A                           ⍝ Change scalars to vectors at depths 0-2.
AA←(⌈/(-⌈/ρ¨ρ¨AA)↑¨ρ¨AA)ρ¨AA      ⍝ Force each item to same shape by reshape.
AV←(⌈/ρ¨AV)↑¨AV                  ⍝ Force each item to same shape by overtake.
A←,['']A                         ⍝ Ghost Buster.  Inc rank by one on last dim.
A←,[¯.5]A                        ⍝ Increase rank by one on the first dim.
A←,[.5]A                         ⍝ Increase rank by one on the first dim.
A←,[IS+.5]A                      ⍝ Increase rank by one after dim IS.
A←((-ρρAY)↑((ρρAY)ρ1),ρAX)ρAX    ⍝ Increase rank of AX to rank of AY.
A←,[2↑ιρρA]A                     ⍝ Decrease rank of A by 1.  Rank 2 or higher.
AV←⊂[¯1+(0≠ρρA)/ρρA]A            ⍝ Decrease rank of A by 1.
AV←⊂[(0≠ρρA)/ρρA]A               ⍝ Decrease rank of A by 1.
A←(⍋(-ρρA)↑1 0)⍉A                ⍝ Transpose every submatrix of A.
IO←ρ,A                           ⍝ Number of elements in A as vector.
IS←×/ρA                          ⍝ Number of elements in A.
IS←×/1↓ρA                        ⍝ Number of elements in a plane of 3D A.
IO←¯1↑ρA                         ⍝ Number of columns in A as vector.
IS←↑⌽ρA                          ⍝ Number of columns in A.
IO←1↑ρM                          ⍝ Number of rows in M as vector.
IS←↑ρM                           ⍝ Number of rows in M.
I←ιρρA                           ⍝ All axes of array A.
I←ι¯1↑ρA                         ⍝ All column indices of array A.
I←ι↑ρM                           ⍝ All row indices of matrix M.
I←ιρV                            ⍝ All indices of vector V.
IO←ρρA                           ⍝ Rank of A.
AV←ρ¨ρ¨AA                        ⍝ Rank of each item in an array.
IO←↑ρ¨ρ¨AA                       ⍝ Rank of the first item in an array.
IO←ρρ↑AA                         ⍝ Rank of the first item in an array.
(,A)←⊂AX                         ⍝ Replace all items, shape unchanged.
(B/,A)←(+/B)ρV                   ⍝ Replace selected items, shape unchanged.
A←↑A                             ⍝ The first item in any rank array.
A←↑⌽,A                           ⍝ The last item in any rank array.
A←↑0ρ⊂↑A                         ⍝ The prototype of A.
A←↑0ρ⊂A                          ⍝ The type of A.
IO←1+(0,ρ,A)ι0+.=↑¨0ρ¨,A         ⍝ Type of simple A. 1-char, 2-num, 3-mixed.
IO←(0,ρ,A)ι0+.=↑¨0ρ¨,A           ⍝ Type of simple A. 1-char, 2-num, 3-mixed.
B←A∊ι0                           ⍝ Zeros, same shape when A is simple.
B←A≠A                            ⍝ Zeros, same shape and structure.
B←0+/V                           ⍝ Zeros, same shape plus one.
B←A=A                            ⍝ Ones, same shape and structure.
B←0×/V                           ⍝ Ones, same shape plus one.
```

Text Arrangement Algorithms

```
CM←(-⌊.5×+/∧\⌽CM=' ')⌽CM         ⍝ Centering left justified CM.
CM←(⌈.5×+/∧\CM=' ')⌽CM           ⍝ Centering right justified CM.
CM←(⌈.5×↑-/+/¨∧\¨B(⌽B←CM=' '))⌽CM⍝ Centering non-justified CM.
CM←(+/∧\CM=' ')⌽CM               ⍝ Left justify matrix CM.
CM←(1-(1,CM=' ')⊥1)⌽CM           ⍝ Right justify matrix CM.
C←(-⌊.5×0⌈NS-⍴C)⌽NS↑C            ⍝ Centering C in field width NS.
C←(-NS)↑C                        ⍝ Left justify C in field width NS.
C←NS↑C                           ⍝ Right justify C in field width NS.
CM←(1+B)⌿CM                      ⍝ Replicate CM at rows indicated by B.
(,M)←(,M)[⍋,(2×⍳↑⍴M)+[0]' '=M]   ⍝ Move blanks to end of each row of M.
(,M)←(,M)[⍋,(2×⍳↑⍴M)+[1]' '=M]   ⍝ Move blanks to end of each row of M.
CM←(' '∨.≠CM)/CM                 ⍝ Remove blank columns.
CM←(CM∨.≠' ')⌿CM                 ⍝ Remove blank rows.
CM←(¯1↓1,B∨1⌽B←' '∨.≠CM)/CM      ⍝ Remove duplicate blank columns.
CM←(¯1↓1,B∨1⌽B←CM∨.≠' ')⌿CM      ⍝ Remove duplicate blank rows.
V←1↓(B∨1⌽B←0,C≠' ')/' ',C        ⍝ Remove lead, trail, and duplicate blanks.
CM←(∨\' '∨.≠CM)/CM               ⍝ Remove leading blank columns.
CM←(∨\CM∨.≠' ')⌿CM               ⍝ Remove leading blank rows.
CM←(1-(1,' '∧.=CM)⊥1)↓[1]CM      ⍝ Remove trailing blank columns.
CM←(1-(1,' '∧.=CM)⊥1)↓[2]CM      ⍝ Remove trailing blank columns.
CM←(1-(1,CM∧.=' ')⊥1)↓[0]CM      ⍝ Remove trailing blank rows.
CM←(1-(1,CM∧.=' ')⊥1)↓[1]CM      ⍝ Remove trailing blank rows.
V←(∨\C≠' ')/C                    ⍝ Remove leading blanks.
V←(⌽∨\⌽C≠' ')/C                  ⍝ Remove trailing blanks.
((1=⌈\' 0'⍳C)/C)←' '             ⍝ Replace leading zeros with blanks.
((2=⌈\' 0'⍳C)/C)←' '             ⍝ Replace leading zeros with blanks.
VV←⍎∊'''',C,↑''''((C=' ')/C)←⊂''' ''' ⍝ Sentence to vector of words.
VV←((2>/1,B)/⍳⍴B)↓¨(1+(2</B,1)/⍳⍴B←C=' ')↑¨⊂C ⍝ Sentence to vector of words.
VV←(¯1+(2>/1,B)/⍳⍴B)↓¨((2</B,1)/⍳⍴B←C=' ')↑¨⊂C⍝ Sentence to vector of words.
VV←(C≠' ')⊂C                     ⍝ Sentence to vector of words.
C←∊' ',¨VV                       ⍝ Sentence from vector of words.
CM←⊃(~V∊C)⊂V                     ⍝ Vector to matrix at selected characters.
C←(IS×⍴C)⍴C                      ⍝ Copies. Create IS copies of C.
```

Text Change/Select Algorithms

```
C←(1+C⍳'←')↓C←2 ⎕TF 'C'          ⍝ Doubles quotes in an expression.
C←(C⍳'←')↓C←2 ⎕TF 'C'            ⍝ Doubles quotes in an expression.
C←'''',((1+C='''')/C),''''       ⍝ Doubles quotes in an expression.
C←(B∨≠\B←C='''')/C               ⍝ Text (including quotes) in expression.
VV←((~B)∧≠\B←C='''')⊂C           ⍝ Text (without quotes) in expression.
C←(≠\C='''')/C                   ⍝ Text (with first quote) in expression.
V←(1+V∊X)/V                      ⍝ Doubles each occurrence of X within V.
VVY←(VVX,' ')[VX⍳VY]             ⍝ Find description of VY from VX index to VVX.
C←NS⊃(C≠' ')⊂C                   ⍝ Finds word number NS in C.
CM←(∨/C⍷CM)⌿CM                   ⍝ Finds the rows of CM containing C.
CM←(,1↑[1]C⍷CM)⌿CM               ⍝ Finds the rows of CM that start with C.
CM←(,1↑[2]C⍷CM)⌿CM               ⍝ Finds the rows of CM that start with C.
VV←VVX~VVY                       ⍝ Proof.  Returns items of VVX not in VVY list.
C←(∧\C≠⎕TC[1])/C                 ⍝ Keep everything up to the 1st return.
C←(∧\C≠⎕TC[2])/C                 ⍝ Keep everything up to the 1st return.
C←(¯1+C⍳⎕TC[1])↑C                ⍝ Keep everything up to the 1st return.
C←(¯1+C⍳⎕TC[2])↑C                ⍝ Keep everything up to the 1st return.
C←(≠\C=' ')/C                    ⍝ Keep even words in a phrase.
C←(≠\¯1↓1,C=' ')/C               ⍝ Keep odd words in a phrase.
S←↑N↓V                           ⍝ Get (N+1)th item in vector V.
M←⊃∊¨⊂[1]↑M(B/,M)←(+/B←,M='⌀')⍴':HP1.' ':EHP1.'⍝ Alternate beg/end tags.
M←⊃∊¨⊂[2]↑M(B/,M)←(+/B←,M='⌀')⍴':HP1.' ':EHP1.'⍝ Alternate beg/end tags.
CM←⊃∊¨⊂[1]↑CM((,CM∊C)/,CM)←↑C     ⍝ Insert the first item of C where C is in CM.
CM←⊃∊¨⊂[2]↑CM((,CM∊C)/,CM)←↑C     ⍝ Insert the first item of C where C is in CM.
C←C~¨' '                         ⍝ Remove blanks in each string.
C←C~' '                          ⍝ Remove blanks.
C←C~'.,:;?!'                     ⍝ Remove punctuation.
((,CA=' ')/,CA)←'-'              ⍝ Replace all blanks with dashes.
((,CA∊1↓C)/,CA)←↑C               ⍝ Replace all occurrences of element in array.
CM←C,[¯.5]'¯'                    ⍝ Underlines a string.
CM←C,[.5]'¯'                     ⍝ Underlines a string.
CM←C,(¯1↑⎕TC),(C≠' ')\'¯'        ⍝ Underlines non-blanks in a string.
```

Trigonometry Algorithms

```
NA←NA×○÷180                      ⍝ Convert from degrees to radians.
NA←NA×180÷○1                     ⍝ Convert from radians to degrees.
NA←12○0J1⊥⊖NA                    ⍝ Convert from NA pairs to radians. 2=↑⍴NA
NA←12○AX+¯11○AY                  ⍝ Convert from AX, AY coordinates to radians.
NA←|0J1⊥⊖NA                      ⍝ Get magnitude of NA pairs. 2=↑⍴NA
NA←|AX+¯11○AY                    ⍝ Get magnitude of AX, AY coordinates.
FA←1○NA                          ⍝ Sine of NA in radians.
FA←2○NA                          ⍝ Cosine of NA in radians.
FA←3○NA                          ⍝ Tangent of NA in radians.
FA←¯1○NA                         ⍝ Arcsine of NA in radians.
FA←¯2○NA                         ⍝ Arccosine of NA in radians.
FA←¯3○NA                         ⍝ Arctangent of NA in radians.
FA←5○NA                          ⍝ Hyperbolic Sine of NA in radians.
FA←6○NA                          ⍝ Hyperbolic Cosine of NA in radians.
FA←7○NA                          ⍝ Hyperbolic Tangent of NA in radians.
FA←¯5○NA                         ⍝ Hyperbolic Arcsine of NA in radians.
FA←¯6○NA                         ⍝ Hyperbolic Arccosine of NA in radians.
FA←¯7○NA                         ⍝ Hyperbolic Arctangent of NA in radians.
FA←0○NA                          ⍝ Pythagorean: FA = side NA = side:hyp ≤ 1.
FA←4○NA                          ⍝ Pythagorean: FA = hypotenuse NA = side ratio.
FA←¯4○NA                         ⍝ Pythagorean: FA = side NA = hyp:side ≥ 1.
NA←AX+.⋆2                        ⍝ Pythagorean: Sum of the squares of AX.
FA←NAX⋆÷NAY                      ⍝ The NAYth root of the items in NAX.
FA←⋆NA                           ⍝ The Exponential. e to the NAth power.
FA←NAX⍟NAY                       ⍝ The NAX based logarithm of NAY.
FA←⍟NA                           ⍝ The natural logarithm of NA.
```

Vectorizing Algorithms

```
M←⍉⊃VV                           ⍝ Column table.  Vectors to columns of matrix.
M←⊃VV                            ⍝ Row table.  Vectors to rows of a matrix.
VV←,⌿M                           ⍝ Matrix to vector of column vectors.
VV←⊂[1]M                         ⍝ Matrix to vector of row vectors.
VV←⊂[2]M                         ⍝ Matrix to vector of row vectors.
MV←⍉⊃⊂[1]¨VM                     ⍝ Vector of matrices to matrix of vectors.
MV←⍉⊃⊂[2]¨VM                     ⍝ Vector of matrices to matrix of vectors.
VV←↑,/⊂[1]¨VM                    ⍝ Vector of matrices to vector of vectors.
VV←↑,/⊂[2]¨VM                    ⍝ Vector of matrices to vector of vectors.
VM←⊃¨⊂[1]MV                      ⍝ Matrix of vectors to vector of matrices.
VM←⊃¨⊂[2]MV                      ⍝ Matrix of vectors to vector of matrices.
AV←↑,¨/VA                        ⍝ Joining corresponding items in vectrices.
M←↑,/MW MX MY                    ⍝ Joining conforming matrices - horizontally.
M←⊃↑,/⊂[1]¨MW MX MY              ⍝ Joining matrices - vertically.
M←⊃↑,/⊂[2]¨MW MX MY              ⍝ Joining matrices - vertically.
A←⊃,/AA                          ⍝ Joining array of arrays - horizontally.
M←⊃,[0]/AA                       ⍝ Joining array of arrays - vertically.
A←⊃,[1]/AA                       ⍝ Joining array of arrays - vertically.
A←,⊃AA                           ⍝ Enlist - top down.  Remove highest nesting.
V←M[;0]                          ⍝ Vectorize - keep only 1st column of M.
V←M[;1]                          ⍝ Vectorize - keep only 1st column of M.
V←⊂[1↓⍳⍴⍴A]1/A                   ⍝ Vectorize - for any rank.
M←,[¯1↓⍳⍴⍴A]1/A                  ⍝ Matricize - for any rank.
M←((×/¯1↓⍴A),¯1↑1,⍴A)⍴A          ⍝ Matricize - for any rank.
M←(¯2↑1 1,⍴A)⍴A                  ⍝ Matricize - rank 0, 1, or 2.
VV←(+/∨\⌽M≠' ')↑¨⊂[1]M           ⍝ Reversing disclose.
VV←(+/∨\⌽M≠' ')↑¨⊂[2]M           ⍝ Reversing disclose.
V←(⊂[1]M)~¨' '                   ⍝ Reversing disclose.  Eliminating blanks.
V←(⊂[2]M)~¨' '                   ⍝ Reversing disclose.  Eliminating blanks.
VA←⊃[1↓⍳⍴⍴A]¨(+\B)⊂⊂[1↓⍳⍴⍴A]A    ⍝ Split A into a vector of arrays given B.
VV←(+\B)⊂V                       ⍝ Split V into subvectors given boolean B.
VV←(L/⍳⍴L)⊂V                     ⍝ Split V into subvectors indicated by L.
N←+/¨(+\B)⊂V                     ⍝ Sum of subvectors of V given boolean B.
N←+/¨(L/⍳⍴L)⊂V                   ⍝ Sum of subvectors of V indicated by L.
V←IS⊃(+\B)⊂V                     ⍝ ISth subvector of V given boolean B.
V←IS⊃(L/⍳⍴L)⊂V                   ⍝ ISth subvector of V given length L.
M←↑,/V,⊂M                        ⍝ Prefix vector to each row of matrix.
M←⊃,/M,⊂V                        ⍝ Postfix vector to each row of matrix.
A←⊃,/AX,⊂AY                      ⍝ Combine 2 arrays along their last dimension.
VV←⊃,/((⍴¨VV)⍴¨⊂1+LS↑1)⊂¨VV      ⍝ Reblock.  Cut VV into many ≤LS length vecs.
VV←⊂[2]B/⊃VV                     ⍝ Reduce each item of VV by B. (⍴B)∧.=∊⍴¨VV
```