

*THE
APL
HANDBOOK
OF
TECHNIQUES*

*THE
APL
HANDBOOK
OF
TECHNIQUES*

"Compiled by DP Scientific Marketing"

First Edition (April 1978)

Requests for copies of IBM publications should be made to your IBM representative or to the branch office serving your locality.

Address comments concerning the contents of this publication to IBM Corporation, Technical Publications, Dept. 824, 1133 Westchester Avenue, White Plains, New York 10604. Comments become the property of IBM.

© Copyright International Business Machines Corporation 1978

PREFACE

The *APL Handbook of Techniques* is intended to augment the "bag of tricks" of the active **APL** user. As in the case of the primitive functions, the defined functions illustrated in this handbook may be used without full understanding of their methodology; however, any time spent analyzing the statements will be richly rewarded with new insights into the power of **APL** and the amazing foresight of Ken Iverson and Adin Falkoff.

What you are holding is a compendium of hundreds of functions submitted by professional programmers within IBM. These many contributions have been generalized, extended and harmonized into families (such as Text-Editing, Logical Operations, Report Formatting, Multi-Precision Arithmetic and Workspace Management). That **APL** is an art form is quickly evident by examining the various styles represented.

Various criteria were used in selecting and refining these functions: elegance, space and execution time. However, as you become familiar with each of the functions, you should experiment with your own variations, thereby imparting a personal style into your work. Once understood, the functions can be modified with confidence in the integrity of **APL** and its predictability.

Preparing this collection has been a very rewarding experience for me. I have often said that I am the greatest benefactor of this publication, as many of the functions were used to prepare the book itself. But the task was aided by the many contributors and the assistant editors. I would like to thank Len Lewis of DPD Scientific Marketing who believed from the beginning that such a publication was indeed possible. For the idea and the model, thanks to Curt Bury and Dr. Kent Haralson, respectively. For their contributions and long hours of testing, thanks to Larry Breed, Norm Brenner, Sylvia Eusebi, Ed Eusebi, Len Gilman, Tim Hollis, Rainer Kogon, Dieter Lattermann, Beth Luc, Blair Martin, John McCleary, John McPherson, Joe Myers, Don Orth and Harry Saal.

Dave Macklin
December, 1977

De gustibus non est disputandum

INTRODUCTION

There are many publications from which one can learn how the operators work, and how to combine characters to form **APL** expressions, but few are written with the intention of developing the reader's style.

This handbook contains no explanation of the **APL** primitives; we assume you already understand them. Similarly, fundamental operations are not emphasized. The goal of this handbook is to furnish you with a collection of meaningful, useful **APL** functions, each demonstrating a particular technique. By carefully examining each function, you should begin to expand your **APL** awareness, thus becoming more proficient in the use of the language.

As with any programming language, there is no single way to solve a problem. However, preferred methods yield elegant functions which are either time or space efficient. Conversely, some approaches produce **APL** functions which can be inefficient or limited in scope. To improve your style, study this book and others like it. Examine functions written by experienced **APL** problem-solvers; modify those functions to suit your own needs. Fine tune your ability to recognize most efficient **APL** technique for solving that problem facing you.

Variable Usage

Within this publication you will notice both "global" and "local" **APL** variables. Without the global concept, variables which are used by sets of functions would have to be identified on each page they are used. Some of the global variables are:

```
AV← ' ABCDEFGHIJKLMNOPQRSTUVWXYZ'
ALF←AV, 'Δ ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
DIGITS←'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
BK (BACKSPACE CHARACTER)- SEE VTCC PAGE 44
CR (CARRIAGE RETURN) " "
ID (IDLE CHARACTER) " "
LN (LINEFEED CHARACTER) " "
TB (TAB CHARACTER) " "
```

Programming Note

A very interesting technique is employed in this publication. It can help you to understand how **APL** functions work. On many pages, you will find an "ANALYSIS" section. The first line of this section will be the expression being analyzed. As you read on, you will notice a line-by-line explanation of the interim results, as though the expression were being executed. By carefully examining this analysis, you will learn how and why the function accomplishes the stated technique.

The information contained in this document has not been submitted to any formal IBM test.
Potential users should evaluate its usefulness in their environment.

FORMAT OF EACH PAGE

Each page of this handbook contains exactly one **primary** and, optionally, one or more **subordinate** (secondary) functions. If they appear, **subordinate** functions are located to the side of the page. To locate *any* function (**primary** or **subordinate**), refer to the complete subject index or the KWIC index in the Appendix.

		abstract	subordinate function name
function name	DIFF	DIFFERENCES BETWEEN ADJACENT ELEMENTS	[UNSCAN]
calling sequence	SYNTAX:	R←DIFF A	
what it does		<ul style="list-style-type: none"> • A IS ANY NUMERIC STRUCTURE. • R HAS ONE FEWER COLUMNS THAN A, AND CONSISTS OF THE SUCCESSIVE DIFFERENCES. A[...;I+1]-A[...;I] 	
APL code	FUNCTIONS:	[1] $\nabla R \leftarrow DIFF A$ $R \leftarrow ((-ppA) + 1) + -A - 1 \phi A$	$\nabla R \leftarrow UNSCAN V$ [1] $R \leftarrow V[\square IO], DIFF V$ [2] $A \leftarrow V \leftarrow + \backslash R$ ∇
	EXAMPLE:		subordinate function APL code
	DIFF 11 22		
	DIFF -3 4p12		
	-1 -1 -1		
	-1 -1 -1		
	-1 -1 -1		
	ANALYSIS:		
	[1] $R \leftarrow ((-ppA) + 1) + -A - 1 \phi A$		
	53 25 18 24		
	40 11 9 20		
	51 53 14 3		
	[1] $R \leftarrow ((-ppA) + 1) + -A - 1 \phi A$		
	25 18 24 53		
	11 9 20 40		
	53 14 3 51		
	[1] $R \leftarrow ((-ppA) + 1) + -A - 1 \phi A$		
	28 7 -6 -29		
	29 2 -11 -20		
	-2 39 11 -48		
	[1] $R \leftarrow ((-ppA) + 1) + -A - 1 \phi A$		
	-28 -7 6 29		
	-29 -2 11 20		
	2 -39 11 48		
	[1] $R \leftarrow ((-ppA) + 1) + -A - 1 \phi A$		
	-2		
	[1] $R \leftarrow ((-ppA) + 1) + -A - 1 \phi A$		
	0 -1		
	[1] $R \leftarrow ((-ppA) + 1) + -A - 1 \phi A$		
	-28 -7 6		
	-29 -2 11		
	2 -39 -11		

TABLE OF CONTENTS

	Page
I. Matrix Manipulation Functions.....	1
II. Report Formatting Functions.....	24
III. Workspace Management Functions.....	40
IV. Multiprecision Arithmetic Functions.....	47
V. Mathematical and Numerical Functions.....	63
VI. Utility and Miscellaneous Functions.....	79
 APPENDIX	
APL Bibliography.....	92
KWIC Index.....	93
SORT of Function Name vs. Abstract	
by Abstract.....	106
by Function Name.....	108
Subject Index.....	110

Section I

Matrix Manipulation Functions

ADDCOLS ADD COLUMNS TO A MATRIX VECTOR OR SCALAR

SYNTAX: Z←A ADDCOLS B

- CONVERTS SCALAR AND VECTOR RIGHT ARGUMENTS INTO ONE-ROW MATRICES AND PADS THEM OUT ON THE LEFT (POSITIVE LEFT ARGUMENT) OR ON THE RIGHT (NEGATIVE LEFT ARGUMENT). RIGHT ARGUMENT MAY BE EITHER NUMERIC OR CHARACTER.
- USES: VMATRIX

FUNCTION:

```
∇ Z←A ADDCOLS B
[1] Z←((1↑ρB),-A+(×A)×1↑ρB)↑B←MATRIX B
∇
∇
```

EXAMPLE:

```
      3 ADDCOLS 2 3ρ16
0 0 0 1 2 3
0 0 0 4 5 6
      -2 ADDCOLS 3 4 5 7
3 4 5 7 0 0
      0 ADDCOLS 2 3ρ'ABCDEF'
```

ANALYSIS:

- ␣ THE RIGHT ARGUMENT IS CONVERTED TO A MATRIX, THEN IS PADDED
- ␣ ON THE LEFT OR RIGHT, OR MADE EMPTY,
- ␣ AS SPECIFIED BY THE SIGNUM OF THE LEFT ARGUMENT (×A).

ADDROWS ADD ROWS TO A MATRIX VECTOR OR SCALAR

SYNTAX: $Z \leftarrow A \text{ ADDROWS } B$

- CONVERTS SCALAR AND VECTOR RIGHT ARGUMENTS INTO ONE-ROW MATRICES AND PADS THEM OUT ON TOP (POSITIVE LEFT ARGUMENT) OR ON THE BOTTOM (NEGATIVE LEFT ARGUMENT). RIGHT ARGUMENT MAY BE EITHER NUMERIC OR CHARACTER.
- USES: ∇ MATRIX

FUNCTION:

$\nabla \quad Z \leftarrow A \text{ ADDROWS } B$
[1] $Z \leftarrow ((-A + (\times A) \times 1 \uparrow \rho B), 1 \uparrow \rho B) \uparrow B \leftarrow \text{MATRIX } B$
 ∇

EXAMPLE:

 2 ADDROWS 2 2 p1 4
0 0
0 0
1 2
3 4
 '|', -2 ADDROWS 'ABCDEF'
|ABCDEF
|
|
 1 ADDROWS 3 4 5 6
0 0 0 0
3 4 5 6
 p1 ADDROWS 3
2 1

ANALYSIS:

- Ⓐ THE RIGHT ARGUMENT, CONVERTED TO A MATRIX, IS PADDED
- Ⓐ AT THE TOP OR BOTTOM OR MADE EMPTY, AS
- Ⓐ SPECIFIED BY THE SIGNUM OF THE LEFT ARGUMENT ($\times A$).

CCAT CATENATE BY COLUMNS [VERTAB CMATRIX ROWFORM]

SYNTAX: R←A CCAT B

- SIDE-BY-SIDE CATENATION OF GENERAL STRUCTURES AND TYPES.
- SCALARS WILL BE REPLICATED IF TYPES AGREE.
- VECTORS BECOME ONE-COLUMN MATRICES BEFORE CATENATION.
- NUMERIC TYPES WILL BE PADDED WITH ZEROS.
- CHARACTER TYPES WILL BE PADDED WITH BLANKS.
- NUMERIC TYPES WILL BE FORMATTED IF TO BE CATENATED WITH CHARACTER TYPES. SEE VBESIDE
- A MATRIX IS RETURNED.

FUNCTIONS:

```

      ▽ R←A CCAT B
[1]  A COLUMN CATENATION, SCALAR REPLICATION
[2]  VERTAB
[3]  CFORMAT
[4]  CMATRIX
[5]  ROWFORM
[6]  R←A, B
      ▽
      ▽ VERTAB
[1]  A ASSUMES A AND B HAVE BEEN LOCALIZED
[2]  A←VERT A Δ B←VERT B
      ▽

```

EXAMPLES:

```

      ▽ ROWFORM;R
[1]  A ASSUMES A AND B HAVE BEEN LOCALIZED
[2]  →0 IF 0=(ρρA)×ρρB
[3]  A←(R,1+ρA)↑A Δ B←((R←(1+ρA)↑1+ρB),1+ρB)↑B
      ▽

      V CCAT S
A◦
B◦
C◦
      V CCAT U
A*
B□
C
      V CCAT M
AXXX
BXXX
C
      M CCAT T
XXXXX
XXXXX
XX
      S CCAT M CCAT S
◦XXX◦
◦XXX◦

```

CHAR BUILD CHARACTER ARRAY TO NUMERIC PATTERN

SYNTAX: R←K CHAR N

- DISPLAYS OR OTHER STRUCTURES CAN BE FASHIONED FROM LOGICAL OR NUMERIC STRUCTURES, LATER TO BE OVERLAID BY ∇FILLS.
- USES: ∇ONESIN ∇Δ

FUNCTION:

```
∇ R←K CHAR N;∇IO
[1]  A K IS SINGLE CHARACTER TO BE PLACED
[2]  A ACCORDING TO POSITIONS OF ONES IN N
[3]  R←(×/ρN)ρ' 'Δ' ∇IO←0
[4]  R[ONESIN,N]←K
[5]  R←(ρN)ρR
∇
```

EXAMPLE:

```
      PATTERN
1 0 0 0 1
0 1 0 1 0
0 0 1 0 0
0 1 0 1 0
1 0 0 0 1
      'X'CHAR PATTERN
X      X
  X  X
    X
  X  X
X      X
```


EDIT EDIT LATENT EXPRESSION OR CHARACTER STRUCTURE [SEDIT]

SYNTAX: EDITED←EDIT UNEDITED

- LATENT EXPRESSIONS AND OTHER CHARACTER STRINGS ARE OFTEN DIFFICULT TO MODIFY, (CHANGE, ADD, DELETE), ESPECIALLY WHEN QUOTATION MARKS ARE INVOLVED. EDIT PERMITS YOU TO DEAL WITH THE FINAL APPEARANCE OF THE VECTOR.
- VECTORS MAY EXPAND OR CONTRACT.
- USE VMEDIT TO MODIFY MATRICES.

FUNCTIONS:

```

      ▽ EDITED←EDIT UNEDITED;SLASH
[1]   □←EDITED←((+/' '=SLASH)↑UNEDITED),□,(ρ,SLASH←□)↑□←UNEDITED
      ▽

      ▽ EDITED←SEdit UNEDITED;SLASH
[1]   A SAME AS EDIT, BUT INSERT IS AN EXECUTABLE EXPRESSION
[2]   □←EDITED←((+/' '=SLASH)↑UNEDITED),(±□),(ρ,SLASH←□)↑□←UNEDITED
      ▽
```

EXAMPLES:

A CHARACTER VECTOR WITH QUOTES WOULD HAVE TO BE KEYED, THUS:

```

      □←A←'HE SAID, ''''HELP ME, I'M DROWNING!''''
HE SAID, ''HELP ME, I'M DROWNING!''
```

```

A←EDIT''          (TO CREATE ITEM)
                  (OPPORTUNITY TO DELETE)
(OPPORTUNITY TO ADD, NOW)
HE SAID, ''HELP ME, I'M DROWNING.''
HE SAID, ''HELP ME, I'M DROWNING!'' (FOR PROOFREADING)
      A←EDIT A
HE SAID, ''HELP ME, I'M DROWNING!''
                  (SPACE TO INSERTION POSITION)
PLEASE
HE SAID, ''HELP ME PLEASE, I'M DROWNING!''
      A←EDIT A
HE SAID, ''HELP ME PLEASE, I'M DROWNING!''
                  (SPACE TO INSERTION POSITION)
PLEASE
HE SAID, ''PLEASE HELP ME PLEASE, I'M DROWNING!''
      A←EDIT A
HE SAID, ''PLEASE HELP ME PLEASE, I'M DROWNING!''
                  // (TO DELETE)
                  (NO INSERTION)
HE SAID, ''PLEASE HELP ME, I'M DROWNING!''
```

```

      SEDIT 'TYPEWRITER IS A X-LETTER WORD.'
TYPEWRITER IS A X-LETTER WORD.
```

```

      /
      ▽ρ'TYPEWRITER'
TYPEWRITER IS A 10-LETTER WORD.
```

ERECT ERECT WORD MATRIX FROM CHARACTER STRUCTURE [DLTMB]

SYNTAX: R←ERECT A

- BUILDS TABLE THAT CAN BE ADDRESSED RANDOMLY OR SEQUENTIALLY
- INPUT WORD SEPARATORS ARE SINGLE OR MULTIPLE BLANKS
- INPUT NEED NOT BE A VECTOR.
- HIGH-SPEED DESIGN COMPUTES ADDRESSES. (SEE ∇SHAPE)
- USES: ∇DIFF ∇DLTMB ∇Δ

FUNCTIONS:

```

      ∇ R←ERECT A;□IO;L;S;D;COLS;ROWS;Z
[1]  A AVOIDS OUTER PRODUCT FOR SPEED; ASSUMES BLANK DELIMITERS
[2]  COLS←[/D←~1+DIFF 0,S1\ROWS+1+~1+S++\L←' '=A←DLTMB A Δ □IO←1
[3]  Z←(ROWS×COLS)ρ' '
[4]  Z[((~L)/+\L\COLS-~1+D)+1+/~L]←(~L)/A
[5]  R←(ROWS,COLS)ρZ
      ∇

      ∇ R←DLTMB A;Z
[1]  A DELETE LEADING, TRAILING, MULTIPLE BLANKS
[2]  R←1+(Z∇1φZ←A≠' ')/A←,' ',A
      ∇
```

EXAMPLE:

```

      M
VFORM                      VARIABLE FORMAT BY ROW OF A MATRIX [ ESCAPE ESCAPEX ]
XVEC                      EXPAND LOGICAL VECTOR
ZDIV                      ZERO TOLERANT DIVISION [ CDIV ]
      TIME'J←ERECT M'
10 MSEC 0 BYTES
      TIME'J←' ' 'SHAPE,M'
22 MSEC 0 BYTES
      ERECT M
VFORM
VARIABLE
FORMAT
BY
ROW
OF
A
MATRIX
[
ESCAPE
ESCAPEX
]                      (NOTE USE OF Z AS LOCAL VARIABLE TO SAVE TIME AND SPACE)
XVEC
EXPAND
LOGICAL
VECTOR
ZDIV
ZERO
TOLERANT
DIVISION
[
CDIV
]
```

FIRSTM SELECT FIRST OR ONLY APPEARANCE IN MATRIX [FIRSTV]

SYNTAX: L←FIRSTM M

RETURNS LOGIC VECTOR THAT CAN SELECT:
 • ROWS OF A MATRIX, IGNORING DUPLICATES
 • THEIR INDICES
 • CORRESPONDING ROWS OF INVERTED FILES
 SEE ∇RIOTA

FUNCTIONS:

∇ L←FIRSTM M	∇ L←FIRSTV V	
[1] L←v f < \M^.=QM	[1] L←(V1V)=1pV	
∇	∇	FOR VECTOR ARGUMENTS.
		SEE ∇DREP.

EXAMPLES:

```

      □←M←ERECT'TOM DICK TOM HARRY DICK HARRY'
TOM
DICK
TOM
HARRY
DICK
HARRY
      (FIRSTM M) f M
TOM
DICK
HARRY
      (FIRSTM M) / 11 p M
1 2 4
      FIRSTM M
1 1 0 1 0 0

```

ANALYSIS: (FIRSTM M) f M

[1] L←v f < \M^.=QM
 TRANSPOSE ASSURES CONFORMITY

```

TDTHDH
OIOAIA
MCMRCR
K RKR
Y Y

```

```

[1] L←v f < \M^.=QM
      -----
1 0 1 0 0 0
0 1 0 0 1 0
1 0 1 0 0 0
0 0 0 1 0 1
0 1 0 0 1 0
0 0 0 1 0 1
[1] L←v f < \M^.=QM
      -----
1 1 0 1 0 0
      < \ CAPTURES POSITION OF FIRST 1 ENCOUNTERED.
      LOGIC VECTOR TO SELECT ROWS
TOM
DICK
HARRY

```

FRAME FRAME AN ARRAY [MATRIX CHARACTER]

SYNTAX: Z←FRAME A

- EMPLOYS THE CHARACTERS ' _ | ' TO BUILD A FRAME AROUND ANY ARRAY AFTER RESHAPING IT AS A MATRIX.
- NO DATA IS TRUNCATED.
- USES: ∇CHARACTER ∇TABULATE ∇ADJUSTUP ∇ADJUSTDOWN ∇FRAMETEST ∇MATRIX ∇IF

FUNCTIONS:

<pre> ∇ Z←FRAME A;∇IO [1] →L1 IF CHARACTER Z←A [2] Z←TABULATE Z [3] L0:Z←1 ADJUSTUP' _ ',[∇IO+1]Z [4] Z←' ',(Z,[1]' _ '),' ' [5] →0 Δ Z←(1↑ρZ)ADJUSTDOWN Z [6] L1:→L0 IF 0=FRAMETEST Z←MATRIX Z ∇ </pre>	<pre> ∇ T←CHARACTER A [1] T←0≠0\0ρA ∇ ∇ Z←MATRIX A [1] Z←((×/¯1↑Z),¯1↑Z←1 1,ρA)ρA [2] A RESULT HAS TWO DIMENSIONS ∇ </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------

EXAMPLE:

```

LV←'THIS IS A LITERAL VECTOR'
FRAME LV
|-----|
|THIS IS A LITERAL VECTOR|
|-----|

```

ANALYSIS:

- A LINE 6 CHECKS WHETHER THE ARGUMENT, IF CHARACTER, IS ALREADY FRAMED.
- A LINE 3 FRAMES THE TOP, PLACING THE CHARACTER ' _ | '
- A IN THE FIRST AND LAST COLUMNS.
- A LINES 4 AND 5 FRAME THE SIDES AND BOTTOM.

FRAMETEST CHECKS A MATRIX FOR FRAMING

SYNTAX: Z←FRAMETEST A

- EXAMINES A MATRIX FOR THE PRESENCE OF FRAMING ELEMENTS
'|_-' AROUND ITS PERIPHERY AND RETURNS A 1 IF PRESENT,
0 OTHERWISE.
- USES: VIF

FUNCTION:

```

      V Z←FRAMETEST A
[1]  Z←~□IO+1
[2]  →0 IF 0=x/ρA
[3]  →0 IF A[1;1]≠'|'
[4]  →0 IF 1≠^/, (A[1,1↑ρA;],⊘A[;1,-1↑ρA])ε'|_-'
[5]  Z+1
      V

```

EXAMPLES:

```

      FRAMETEST 3 4ρ'ABCD'
0
      X
|-----|
|SALES|
|-----|
      FRAMETEST X
1
      Y
SALES
-----
1 2 3
4 5 6
      FRAMETEST Y
0

```

ANALYSIS:

- ⌘ LINES 2,3,4 SET THE RESULT TO 0 IF THE ARGUMENT IS EMPTY, OR THE
- ⌘ [1;1] ELEMENT IS NOT '|' OR THE ELEMENTS IN THE FIRST AND LAST ROWS,
- ⌘ FIRST AND LAST COLUMNS ARE NOT ALL MEMBERS OF '|_-' . OTHERWISE THE
- ⌘ RESULT IS 1.

GRADEUP GENERATE ASCENDING ROW INDICES [AV ALF NFORM LJNFORM]

SYNTAX: I←C GRADEUP K

- TO SORT A LEFT-JUSTIFIED MATRIX ALPHABETICALLY
- C IS A COLLATING SEQUENCE; K IS A CHARACTER MATRIX.
- IF UNIQUE DISTINCTIONS OCCUR ONLY AT RIGHT SIDE, AND IF THE COLLATING SEQUENCE IS LONG, IT MAY BE NECESSARY TO SORT IN MORE THAN ONE PASS, FIRST ACCORDING TO THE RIGHTMOST COLUMNS.
- USES: ∇NFORM

FUNCTIONS:

∇ I←C GRADEUP K [1] I←C NFORM K ∇	∇ N←C NFORM K;∇IO [1] N←(ρC)⊥C⊥K Δ ∇IO←0 ∇
-----------------------------------------	--------------------------------------------------

[1] ∇ R←C LJNFORM K R←C NFORM((1↑ρR),11)↑R←ERECT,K,' ' ∇

EXAMPLES:

AV←' ABCDEFGHIJKLMNOPQRSTUVWXYZ'
 A A USEFUL GLOBAL, WITH 11-COLUMN RESOLUTION
 ALF←AV,'Δ_ABCDEFGHIJKLMNOPQRSTUVWXYZΔ0123456789'
 A USED BY ∇VARS...LESS RESOLUTION BUT MORE CHARACTERS

AV GRADEUP 3 1ρ'ZYX'
 3 2 1
 □←A←A[AV GRADEUP A←' 'SHAPE'TOM DICK HARRY';]
 DICK
 HARRY (ALPHABETICALLY SORTED)
 TOM

A[φAV GRADEUP A;]
 TOM
 HARRY (REVERSE ORDER)
 DICK

INDEX COLUMN INDEX IN MATRIX B WHOSE MEMBERS ALL BELONG TO A

SYNTAX: Z←A INDEX B

- RETURNS THE INDEX POSITION OF EACH COLUMN OF B ALL OF WHOSE ELEMENTS ARE IN A. B MUST BE A MATRIX. THE SHAPE OF A IS NOT RESTRICTED. THE ARGUMENTS MAY BE EITHER CHARACTER OR NUMERIC.

FUNCTION:

```

      ∇ Z←A INDEX B
[1]  Z←(∧/B∈A)/∪1↓ρB
      ∇

```

EXAMPLE:

```

      A←∪10
      B←?3 8ρ∪15
      B
1     1     1     3     5     1     4     4
9     8     7    11     9    12     3     1
2     1     1     2     1     3     1     7
      A INDEX B
1 2 3 5 7 8
      'A' INDEX C←3 6ρ'ABCDEF AHIJKLABCXYZ'
1
      C
ABCDEF
AHIJKL
ABCXYZ

```

ANALYSIS:

- ∘ 'ANDS' OVER THE COLUMNS OF THE LOGICAL MATRIX CREATED BY B∈A,
- ∘ THEN USES COMPRESSION TO SELECT THE CORRESPONDING COLUMN INDICES.

MEDIT

EDIT MATRIX

SYNTAX:

R←KD MEDIT M

- RETURNS A MODIFIED FORM OF STRUCTURE M AS A MATRIX.
- KD IS A PAIR OF INTEGERS, THE FIRST, K, SIGNIFIES THE NUMBER OF ROWS OF M PRECEDING THE ONE TO BE CHANGED, INSERTED, OR DELETED. THE SECOND, D, IS THE LINE NUMBER TO BE DELETED.
- THE KEYBOARD WILL UNLOCK FOR THE EXPRESSION THAT WILL GENERATE THE LINE(S) TO BE INSERTED.
- THE INSERT MAY BE AN ALTERED FORM OF THE LINE(S) DELETED.
- WHEN K=D, THE INSERT WILL APPEAR AFTER LINE K.
- WHEN K<D, D-K LINES WILL BE DELETED BEFORE ACCEPTING THE INSERT. (IF K>D, K-D ORIGINAL ROWS WILL APPEAR BEFORE AND AFTER THE INSERT.)
- USES: ∇MATRIX ∇ON ∇Δ

FUNCTION:

```
∇ R←KD MEDIT M;□IO
[1]  R MATRIX EDIT (DELETE, INSERT, CHANGE)
[2]  R←(((1↑KD),1↑ρM)↑M)ON(⊘)ON((1↑KD),0)↑M←MATRIX M Δ □IO←1
∇
```

EXAMPLES:

```
      1 2 MEDIT 3 4ρ112      (DELETING ROW 2)
10      (INSERTING NOTHING)
      1 2 3 4
      9 10 11 12
      1 1 MEDIT 3 4ρ112
1      (INSERTING 1)
      1 2 3 4
      1 0 0 0
      5 6 7 8
      9 10 11 12
      1 2 MEDIT 3 4ρ112
ΦM[2;] (INSERTING FUNCTION OF EXISTING ROW)
      1 2 3 4
      8 7 6 5
      9 10 11 12
      3 0 MEDIT 3 4ρ112      (SUPERIMPOSING FIRST THREE ROWS)
0      (INSERTING ZEROS)
      1 2 3 4
      5 6 7 8
      9 10 11 12
      0 0 0 0
      1 2 3 4
      5 6 7 8
      9 10 11 12
      1 1 MEDIT 2 3ρ'ABCDEF'
' ↑ THIS IS THE LETTER B '
ABC
↑ THIS IS THE LETTER B
DEF
```


M2V COMPRESS CHARACTER MATRIX EXPAND RESULT [V2M]

SYNTAX: V←M2V M AND M←V2M V

THESE COMPLEMENTARY FUNCTIONS ALLOW TWO-WAY CONVERSION BETWEEN CHARACTER MATRICES AND CHARACTER VECTORS.

M2V: CONVERTS A CHARACTER MATRIX M TO A CHARACTER VECTOR V. EACH ROW OF M, WITH TRAILING BLANKS OMITTED, BECOMES A 'LINE' IN V, ENDED BY A CARRIAGE RETURN.

V2M: CONVERTS A CHARACTER VECTOR V TO A CHARACTER MATRIX M. EACH 'LINE' (A CHARACTER STRING ENDING IN A CARRIAGE RETURN) BECOMES A ROW OF M, WITH PADDING AS REQUIRED. BOTH V AND M WILL APPEAR THE SAME WHEN DISPLAYED, BUT THE VECTOR REPRESENTATION IS USUALLY MORE ECONOMICAL IN STORAGE. THE GLOBAL CR MUST EXIST IN WORKSPACE. (SEE VTCC PAGE).

FUNCTIONS:

```

      ▽ V←M2V M
[1]  V←~1+(,φ1,v\ ' '≠φM)/,M,CR
      ▽
      ▽ M←V2M V;□IO
[1]  □IO←1
[2]  M←(ρM)ρ(,M←M°.≥1[ /0,M←M-1+0,~1+M←M/1ρM)\(~M←V=CR)/V←V,CR
      ▽

```

EXAMPLES:

```

      □←V←'LINE 1.',CR,'LINE NUMBER 2'
LINE 1.
LINE NUMBER 2
      ρV
21
      □←M←V2M V          (CONVERT VECTOR TO MATRIX)
LINE 1.
LINE NUMBER 2
      ρM
2 13
      V2←M2V M          (CONVERT MATRIX BACK TO VECTOR FORM)
      ∧/V=V2          (COMPARE TWO VECTORS)
1

```

ANALYSIS:

M2V[1]: A COLUMN OF CARRIAGE RETURNS IS CATENATED ONTO M AND THE RESULT RAVELED AND COMPRESSED BY A BOOLEAN VECTOR TO REMOVE TRAILING BLANKS IN EACH ROW. THE FINAL CARRIAGE RETURN IS THEN REMOVED.

V2M[2]: AFTER A CARRIAGE RETURN IS CATENATED ONTO V, IT IS SEARCHED FOR CARRIAGE RETURNS AND THEY ARE COMPRESSED OUT. THIS RESULT IS THEN EXPANDED BY A BOOLEAN VECTOR WHICH HAS THE EFFECT OF PADDING LINES TO THE SAME LENGTH. THE RESULT IS RESHAPED INTO A MATRIX.

PREEDIT PREPARE MATRIX FOR FUNCTION-LIKE EDITING [POSTEDIT]

SYNTAX: R←TNAME PREEDIT M

- TNAME IS A TEMPORARY NAME TO BE ASSIGNED TO A COPY OF THE MATRIX, M, SO THAT IT MAY BE EDITED AS IF IT WERE A DEFINED FUNCTION. TNAME IS A CHARACTER STRING.
- WHEN EDITING IS COMPLETE, KEY: R←POSTEDIT TNAME, WHERE R CAN BE THE OLD OR NEW NAME OF THE EDITED MATRIX.
- USES: VESCAPE VON

FUNCTIONS:

```

      ▽ R←TNAME PREEDIT M
[1]   (TNAME, ' IN USE')ESCAPE 0≠□NC TNAME
[2]   R←□FX TNAME ON 'A', ▽M
      ▽
                ▽ R←POSTEDIT TNAME;J
[1]   'NOT A NAME'ESCAPE~CHARACTER TNAME
[2]   (TNAME, ' NOT A FUNCTION')ESCAPE 3≠□NC TNAME
[3]   R←1 1+□CR TNAME
[4]   J←□EX TNAME
      ▽
```

EXAMPLES:

```

      A←ERECT'TOM DICK HARRY'
      A
TOM
DICK
HARRY
      B←'JOE'PREEDIT A
      B
JOE
      ▽JOE[□]                    (FOR FUNCTION EDITING)
      ▽ JOE
[1]   A TOM
[2]   A DICK                    (LAMP SYMBOLS PROTECT INTEGRITY OF DATA)
[3]   A HARRY
      ▽
[4]   [Δ2]                    (TO DELETE ROW 2)
[2]   ▽
      C←POSTEDIT B
      C
TOM                            (LAMP SYMBOLS HAVE BEEN REMOVED)
HARRY
```

RCAT CATENATE STRUCTURES BY ROWS [COLFORM VERT]

SYNTAX: R← A RCAT B

- OVER-UNDER CATENATION OF GENERAL STRUCTURES AND TYPES.
- SCALARS WILL BE REPLICATED.
- PADDING WILL BE BLANK FOR CHARACTERS, ZERO FOR NUMBERS.
- NUMERIC TYPES WILL BE FORMATTED IF TO BE CATENATED TO CHARACTER TYPES. SEE: VON VCCAT
- A MATRIX IS RETURNED.
- USES: VCFORMAT VCMATRIX VCOLFORM VVERT

FUNCTIONS:

```

V R←A RCAT B
[1]  A ROW CATENATION, SCALAR REPLICATION
[2]  CFORMAT
[3]  CMATRIX
[4]  COLFORM
[5]  ⚡'R←VERT A,',(0≠(ρρA)+ρρB)/' [␣IO] ', 'B'
V
      V COLFORM;R
[1]  A ASSUMES A AND B HAVE BEEN LOCALIZED
[2]  →0 IF 0=(ρρA)×ρρB
[3]  A←((1↑ρA),R)↑A Δ B←((1↑ρB),R←(1↑ρA)[1↑ρB]↑B
V
      V Z←VERT X
[1]  Z←((ρX),(1=ρρX)ρ1)ρX
V

```

EXAMPLES:

```

S←' ° '
U←' *␣ '
V←' ABC '
S RCAT U
° °
*␣
U RCAT V
*␣
ABC
1 RCAT 15
1 1 1 1 1
1 2 3 4 5
S RCAT 15
° ° ° ° ° ° ° ° ° °
1 2 3 4 5
U RCAT 15
*␣
1 2 3 4 5

```

REPL REPLACE ALL OCCURRENCES OF ELEMENT IN ARRAY

SYNTAX: R←VV REPL A

VV IS A TWO-POSITION CHARACTER OR NUMERIC VECTOR.
A IS AN ARRAY OF THE SAME TYPE.
ALL APPEARANCES OF 1↑VV WILL CHANGE TO 1↓VV.

FUNCTION:

```

      ∇ R←VV REPL A
[1]   R[(R=1↑VV)/⊖ρR←,A]←1↓VV
[2]   R←(ρA)ρR
      ∇

```

EXAMPLES:

```

      '- 'REPL'-1234.56'  MINUS SIGN BECOMES APL NEGATIVE
-1234.56
      0 1E75 REPL (⊖3)∘.=⊖3  CHANGE ZEROS TO 1E75

```

ANALYSIS: 1 -1 REPL 2 5ρ1 0

```

[1]   R[(R=1↑VV)/⊖ρR←,A]←1↓VV
      -----SEND IN REPLACEMENTS
-1

```

```

[1]   R[(R=1↑VV)/⊖ρR←,A]←1↓VV

```

```

1 0 1 0 1
0 1 0 1 0

```

```

[1]   R[(R=1↑VV)/⊖ρR←,A]←1↓VV
      -----NOW A VECTOR

```

```

1 0 1 0 1 0 1 0 1 0

```

```

[1]   R[(R=1↑VV)/⊖ρR←,A]←1↓VV

```

```

1 2 3 4 5 6 7 8 9 10

```

```

[1]   R[(R=1↑VV)/⊖ρR←,A]←1↓VV

```

```

1 0 1 0 1 0 1 0 1 0

```

```

[1]   R[(R=1↑VV)/⊖ρR←,A]←1↓VV

```

```

1 3 5 7 9

```

```

[1]   R[(R=1↑VV)/⊖ρR←,A]←1↓VV

```

```

-1

```

```

[2]   R←(ρA)ρR

```

```

-1 0 -1 0 -1 0 -1 0

```

```

[2]   R←(ρA)ρR

```

```

2 5

```

```

[2]   R←(ρA)ρR

```

```

-1 0 -1 0 -1
0 -1 0 -1 0

```

RIOTA MATRIX ROW IOTA

SYNTAX: R←X RIOTA Y

- RIOTA EXTENDS TO MATRIX ARGUMENTS THE FUNCTION OF DYADIC \vee (A \vee B...THE LEAST INDEX IN VECTOR A OF THE ELEMENTS(S) IN SCALAR OR VECTOR B).
- THE RESULT R IS A VECTOR OF THE RESPECTIVE ROW INDICES OF THE FIRST OCCURRENCE OF THE ROWS OF Y IN X, IGNORING TRAILING BLANKS. IF A ROW OF Y DOES NOT OCCUR IN X, THE CORRESPONDING ELEMENT OF R IS SET TO 1+1 \vee pX. NON-MATRIX ARGUMENTS ARE RESHAPED. SCALAR AND VECTOR ARGUMENTS ARE TREATED AS 1-ROW MATRICES.
- USES: \vee MATRIX \vee Δ

FUNCTION:

```

 $\vee$  R←X RIOTA Y
[1] Y←MATRIX Y  $\Delta$  X←MATRIX X
[2] R← $\square$ IO++/ $\sim$  $\vee$ \(((0\ 1\times pX)[pY])\uparrow Y)\wedge.=Q((0\ 1\times pY)[pX])\uparrow X
 $\vee$ 

```

EXAMPLE: \square IO←1

```

X←3 4p'AAAABBBBCCCC'
Y←2 5p'CCCC XXXXX'
X RIOTA Y
3 4

```

ANALYSIS:

THE LEFT ARGUMENT, X, IS TRANSPOSED AND BECOMES THE RIGHT ARGUMENT IN THE MATRIX INNER PRODUCT

```

CCCC  $\wedge$ . = ABC
XXXXX   ABC
        ABC
        ABC

```

THE RIGHT ARGUMENT IN THAT EXPRESSION HAS A FIFTH (BLANK) ROW TO SATISFY THE INNER PRODUCT REQUIREMENT THAT THE LAST DIMENSION (5) OF THE LEFT ARGUMENT MUST BE THE SAME AS THE FIRST DIMENSION OF THE RIGHT ARGUMENT. EACH ROW OF THE LEFT ARGUMENT IS COMPARED AGAINST EACH COLUMN OF THE RIGHT ARGUMENT GIVING THE MATRIX

```

0 0 1
0 0 0 WHICH IS TRANSLATED INTO THE VECTOR 2 3 BY +/ $\sim$  $\vee$ \. ADDING
 $\square$ IO GIVES THE RESULT 3 4 (FOR ORIGIN 1) OR 2 3 (FOR ORIGIN 0).

```

SHAPE

SHAPE MATRIX FROM CHARACTER STRING

SYNTAX:

R←C SHAPE X;L

- X IS A CHARACTER VECTOR COMPOSED OF PHRASES OF VARIABLE LENGTH, SEPARATED BY ANY OF THE CHARACTERS IN VECTOR C.
- A MEMBER OF C MAY EVEN BE PART OF A PHRASE, IF IT IS SURROUNDED BY QUOTES IN X.
- SEE VERECT

FUNCTION:

```

▽ R←C SHAPE X;L
[1] R←((=\X≠''')^X∈C)/\pX←X,1+C
[2] L←[ /R←R-□IO,1+~1+R
[3] R←(0≠R)≠0 ~1+((pR),1+L)ρ(,(R○.≥(~□IO)+1L),1)\X
▽

```

EXAMPLES:

```

      ' 'SHAPE'TOM DICK HARRY'
TOM
DICK
HARRY
      ';;. 'SHAPE'SEMICOLON';';COMMA';',',PERIOD'.'. '
SEMICOLON';'
COMMA';'
PERIOD'.'. '

```

ANALYSIS:

```

[1]A 'LOGICAL VECTOR SELECTS 12 21 31 AS END POINTS
[2]A MAXIMUM LENGTH COMPUTED AS 12
[3] R←(0≠R)≠0 ~1+((pR),1+L)ρ(,(R○.≥(~□IO)+1L),1)\X

1 2 3 4 5 6 7 8 9 10 11 12
[3] R←(0≠R)≠0 ~1+((pR),1+L)ρ(,(R○.≥(~□IO)+1L),1)\X

1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 0 0 0
[3] R←(0≠R)≠0 ~1+((pR),1+L)ρ(,(R○.≥(~□IO)+1L),1)\X

1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 0 0 0
[3] R←(0≠R)≠0 ~1+((pR),1+L)ρ(,(R○.≥(~□IO)+1L),1)\X

SEMICOLON';';COMMA';',',PERIOD'.'. ' ;
[3] R←(0≠R)≠0 ~1+((pR),1+L)ρ(,(R○.≥(~□IO)+1L),1)\X

3 13
[3] R←(0≠R)≠0 ~1+((pR),1+L)ρ(,(R○.≥(~□IO)+1L),1)\X

SEMICOLON';';
COMMA';',
PERIOD'.'. ' ;

```

(NOW STRIP SUPERFLUOUS PUNCTUATION)

ULINE UNDERLINE SPECIFIED ROWS OF CHARACTER MATRIX [USCORE]

SYNTAX: R←N ULINE K

- RETURNS AN EXPANDED MATRIX WITH UNDERSCORES INSERTED, AS INDICATED BY N, A VECTOR OF ROW NUMBERS [IO+1.
- UNDERLINES WILL NOT APPEAR IN COLUMNS THAT ARE ALWAYS BLANK.
- GIVEN A CHARACTER VECTOR, VUSCORE IS MUCH FASTER THAN VULINE.
- USES: VMATRIX VESCAPEX VXVEC VCVEC VFILLS

FUNCTIONS:

```

▽ R←N ULINE K;L;[IO;J;I
[1] K←MATRIX ▽K
[2] ' 'MATRIX HAS ROWS NUMBERED: ' ',▽J'ESCAPEX~^/NεJ+ιI+([IO+1)↑ρK
[3] R←(L+(1 XVEC I CVEC 1+N),0)▽K
[4] R←(-J,0)↑R FILLS(~L)▽(((J+I>[ /N)+ρ,N),1+ρK)ρ(▽K≠' ')\'-'
▽

```

```

▽ R←USCORE KV
[1] R←KV,[IO-0.5](KV≠' ')\'-'
▽

```

EXAMPLES:

```

NAMES BESIDE SCORES
TOM | 4 60 56
DICK | 107 84 62
HARRY | 18 64 90

```

```

(↑1↑Δ+/SCORES)ULINE NAMES BESIDE SCORES
TOM | 4 60 56
DICK | 107 84 62
HARRY | 18 64 90

```

THE WINNER AND HIS SCORES

```

ρSCORES
3 3 A NUMERIC MATRIX
4 ULINE SCORES
MATRIX HAS ROWS NUMBERED: 1 2 3
3 ULINE SCORES
4 60 56
107 84 62
18 64 90

```

ANALYSIS:

- [1] FORCES 2=ρρK
- [2] CHECKS FOR ILLEGAL ROW NUMBERS, PRINTS MESSAGE, THEN ESCAPES.
- [3] GENERATES EXPANSION VECTOR, THEN EXPANDS K.
- [4] REPLICATES UNDERLINES, IF NECESSARY, EXPANDS THEM, THEN MERGES THEM WITH THE RESULT OF [3], AND FINISHES WITH SOME HOUSEKEEPING, IF NECESSARY.

VFORM VARIABLE FORMAT BY ROW OF A MATRIX [ESCAPE ESCAPEX]

SYNTAX: R←F VFORM M

- THE APL FORMAT FUNCTION (▼) ACTS UNIFORMLY ON ALL ROWS OF A MATRIX WHILE ALLOWING VARIABLE WIDTHS AND DECIMAL PLACES FROM COLUMN TO COLUMN. VFORM PERMITS A UNIFORM WIDTH FOR ALL COLUMNS WHILE ALLOWING INDIVIDUAL ROW DECIMAL PLACES. M IS A NUMERIC MATRIX. F IS A NUMERIC VECTOR OF THE FORM W,D1,N1,D2,N2,... WHERE W IS THE FORMAT WIDTH FOR ALL COLUMNS. D1 IS THE NUMBER OF DECIMAL PLACES IN THE FIRST BLOCK OF ROWS; N1 IS THE NUMBER OF ROWS IN THE FIRST BLOCK, ETC. ρF MUST BE ODD, AND THE SUM OF N'S MUST EQUAL THE NUMBER OF ROWS IN THE MATRIX.
- USES V HANG, ALTHOUGH V ESCAPE MAY BE SUBSTITUTED.

FUNCTIONS:

```

      V R←F VFORM M;I;J;□IO
[1]   □IO←0
[2]   'NOT A MATRIX'HANG 2≠ρρM
[3]   'WRONG LENGTH LEFT ARGUMENT'HANG ~2|ρF
[4]   F←F[0],(((~1+ρF)÷2),2)ρ1↓F
[5]   'WRONG ROW COUNT'HANG(1↑ρM)≠+/J←F[;2]
[6]   F←F[;0 1]
[7]   R←(0,(1↑ρM)×' 'ρF)ρI←0
[8]   L1:R←R,[0]F[I;]▼M[(+/I↑J)+1J[I];]
[9]   →L1×(ρJ)>I←I+1
      V

```

EXAMPLES:

```

      C←5 4ρ120
      C
      1   2   3   4
      5   6   7   8
      9  10  11  12
     13  14  15  16
     17  18  19  20
      8 1 1 0 2 2 2 VFORM C
      1.0      2.0      3.0      4.0
        5      6      7      8
        9     10     11     12
     13.00    14.00    15.00    16.00
     17.00    18.00    19.00    20.00

```

```

      V MSG ESCAPE CONDITION
[1]   A WILL LEAVE NO TRACE....BETTER TO
[2]   A USE V HANG TO CHECK DOMAIN ERRORS
[3]   →0 IF~CONDITION
[4]   MSG
[5]   →

```

```

      V QEXP ESCAPEX CONDITION
[1]   →0 IF~CONDITION
[2]   1QEXP
[3]   →
[4]   A WILL EXECUTE THE QUOTED
[5]   A EXPRESSION, THEN ESCAPE
      V

```

ANALYSIS:

IN LINE 8, J[I] IS THE NUMBER OF ROWS IN THE BLOCK BEING FORMATTED, F[I;] IS THE UNIFORM WIDTH AND THE NUMBER OF DECIMAL PLACES FOR THE BLOCK. EACH BLOCK IS FORMATTED AND CATENATED TO R IN LINE 8 UNTIL THE NUMBER OF BLOCKS EQUALS THE COUNT I IN LINE 9.

WIDTH MEASURE FORMATTED MATRIX

SYNTAX: W←K WIDTH MATRIX

- RETURNS THE ACTUAL WIDTH OF ALL FIELDS AS A VECTOR.
- BLANK AREAS ARE NOT CONSIDERED SIGNIFICANT.
- A GLOBAL LOGIC VECTOR CAPABLE OF COMPRESSING (THEN EXPANDING) THE FORMATTED MATRIX WILL BE NAMED ACCORDING TO THE CHARACTER(S) OFFERED AS K.
- USES: VDMZ

FUNCTION:

```

V W←K WIDTH MATRIX;V;□IO
[1] W←1++V°. = 11V++\ (□IO+1), ~DMZ⊕K, '←V/MATRIX≠' ' ' '
V

```

EXAMPLES:

```

'B'WIDTH 9 2VMM
7 5 5 4

B
0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1

9 2VMM
1000.00      87.92      79.58      8.33
 920.42      87.92      80.25      7.67
 840.17      87.92      80.91      7.00

B/9 2VMM
1000.0087.9279.588.33
 920.4287.9280.257.67
 840.1787.9280.917.00

```

WORD SELECT NTH WORD IN CHARACTER STRUCTURE [DTMB]

SYNTAX: W←NTH WORD K

- IF WORDS IN A VECTOR ARE DELIMITED BY BLANKS, OR APPEAR IN SEPARATE ROWS OF AN ARRAY AND ARE SIMILARLY DELIMITED, THEY CAN BE CHOSEN BY THE NATURAL NUMBERS INDICATING THEIR POSITION, FROM LEFT TO RIGHT AND TOP TO BOTTOM.

FUNCTIONS:

▽ W←NTH WORD K
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K
▽

▽ R←DTMB K;A;Z
[1] a DELETE TRAILING AND MULTIPLE BLANKS
[2] R←(Z▽1φZ←A≠' ')/A←, ' ',K
▽

EXAMPLE:

2 WORD □←A
TOM DICK HARRY
DICK

ANALYSIS:

[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K

TOM DICK HARRY
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K
----- INITIAL BLANK

TOM DICK HARRY
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K
----- TRAILING AND MULTIPLE BLANKS OUT

TOM DICK HARRY
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K

TOM DICK HARRY
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K
----- LOCATION OF ONLY BLANKS

1 0 0 0 1 0 0 0 0 1 0 0 0 0 0
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K
----- FIELDS NUMBERED

1 1 1 1 2 2 2 2 3 3 3 3 3 3
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K
----- COMPARED

0 0 0 0 1 1 1 1 0 0 0 0 0 0
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K
----- SELECTION

DICK
[1] W←1+(NTH=+\ ' '=W)/W←DTMB, ' ',K
----- BLANK FIELD-MARKER DROPPED

DICK

ADJUSTDOWN EXTEND THE '|' IN REPORT FORMATTING [ROWINDICES]

SYNTAX: Z←A ADJUSTDOWN B

- EXTENDS TO ROW A, THE CHARACTER '|' USED AS A SEPARATOR IN MATRIX B
- USES: ∇IF ∇INDEX ∇ROWINDICES ∇Δ

FUNCTIONS:

```

      ∇ Z←A ADJUSTDOWN B;C;D;□IO
[1]   →(0≥C←1↑ρD←B[((|A)-1)ROWINDICES Z←B;])/0 Δ □IO←1
[2]   →L1 IF C≤3
[3]   D←(⊖D)[13;]
[4]   L1:Z[A;'|'INDEX D]←'|'
      ∇

```

EXAMPLES:

```

      ∇ R←N ROWINDICES M;□IO
[1]   A FIRST OR LAST N ROWNUMBERS OF MATRIX
[2]   R←(R≠0)/R←N↑11↑ρM Δ □IO←1
      ∇

```

E

SALES

```

1  2  3  4|ABCDE
5  6  7  8|FGHIJ
9 10 11 12|KLMNO

```

7 ADJUSTDOWN E

SALES

```

1  2  3  4|ABCDE
5  6  7  8|FGHIJ
9 10 11 12|KLMNO

```

|-----

ANALYSIS:

- ⌘ LINE 1 PICKS OUT AND STORES IN D THAT PART OF MATRIX B WHOSE
- ⌘ ROW NUMBERS ARE LESS THAN |A.
- ⌘ LINES 2 AND 3 SELECT THE LAST 3 LINES OF D, WHERE LINE 4 LOCATES
- ⌘ THE COMMON OCCURRENCE OF THE SEPARATOR '|' AND EXTENDS
- ⌘ THEM 1 LINE DOWNWARD.

Section II

Report Formatting Functions

ADJUSTUP EXTENDS '|' IN REPORT FORMATTING

SYNTAX: Z←A ADJUSTUP B

- EXTENDS SEPARATOR '|' UP ONE LINE FROM ROW A IN MATRIX B.
- USES: ∇IF ∇INDEX ∇ROWINDICES ∇Δ

FUNCTION:

```

      ∇ Z←A ADJUSTUP B;C;D;□IO
[1]   →0 IF 0≥C+1+ρD+B[((|A)-1+ρB)ROWINDICES Z+B;]Δ □IO+1
[2]   →L1 IF C≤3
[3]   D←D[13;]
[4]   L1:Z[A;'|'|INDEX D]←'|'
      ∇

```

EXAMPLES:

```

      D
      SALES
      -----
1  2  3  4|ABCDE
5  6  7  8|FGHIJ
9 10 11 12|KLMNO
      3 ADJUSTUP D
      SALES
      -----
1  2  3  4|ABCDE
5  6  7  8|FGHIJ
9 10 11 12|KLMNO
      1 ADJUSTUP 2 ADJUSTUP 3 ADJUSTUP D
      SALES
      -----
1  2  3  4|ABCDE
5  6  7  8|FGHIJ
9 10 11 12|KLMNO

```

ANALYSIS:

- ⌘ LINE 1 PICKS OUT AND STORES IN D THAT PART OF THE ARRAY B WHOSE ROW NUMBERS ARE GREATER THAN |A.
- ⌘ LINES 2 AND 3 SELECT THE NEXT 3 (OR FEWER IF THERE AREN'T 3) LINES,
- ⌘ WHILE LINE 4 LOCATES ON THOSE LINES OCCURRENCES OF THE SEPARATOR '|' AND EXTENDS THEM UPWARD 1 LINE.

BARGRAPH

PLOT HORIZONTAL INTEGER BARGRAPHS

SYNTAX:

R←Q BARGRAPH V

- PRODUCE HORIZONTAL HISTOGRAMS OR GANTT CHARTS
- TWO CLASSES OF INPUT (CODED PLUS AND MINUS) TREATED, ONE INVISIBLE WHILE THE OTHER IS PLOTTED.
- THE CHARACTERS USED FOR THE BARS ARE USER-SPECIFIED.
- THE OUTPUT CAN BE CATENATED TO NAMES, FOR EXAMPLE.
- USES: ∇Δ

FUNCTION:

```

∇ R←Q BARGRAPH V;□IO
[1]  A Q IS CHARACTER TO BE USED FOR BARS
[2]  A V IS A VECTOR OF POSITIVE AND NEGATIVE INTEGERS
[3]  A NEGATIVE INTEGERS STORED IN V WILL BE IGNORED
[4]  A THUS THE SAME VECTOR CAN BE USED TWICE
[5]  R←(-2↑Q)[V◊.≥1+1[ /V]Δ □IO←0
∇

```

ANALYSIS: '□'BARGRAPH 1 -2 3 -4 5

```

[5]  R←(-2↑Q)[V◊.≥1+1[ /V]Δ □IO←0

```

```

1 -2 3 -4 5
[5]  R←(-2↑Q)[V◊.≥1+1[ /V]Δ □IO←0

```

```

5
[5]  R←(-2↑Q)[V◊.≥1+1[ /V]Δ □IO←0

```

WIDTH DETERMINED

```

1 2 3 4 5
[5]  R←(-2↑Q)[V◊.≥1+1[ /V]Δ □IO←0

```

NEGATIVES IGNORED

```

1 0 0 0 0
0 0 0 0 0
1 1 1 0 0
0 0 0 0 0
1 1 1 1 1
[5]  R←(-2↑Q)[V◊.≥1+1[ /V]Δ □IO←0

```

BLANK,Q

```

□
[5]  R←(-2↑Q)[V◊.≥1+1[ /V]Δ □IO←0

```

SHAPE OF INDEXING
MATRIX DETERMINES
SHAPE OF OUTPUT.

□

□□□

□□□□□

EXAMPLE:

FRAME(' 'SHAPE'TOM JANE DICK MARY')BESIDE('◊'BARGRAPH-V)FILLS'□'BARGRAPH V

```

|-----|
|TOM|□|
|JANE|◊◊|
|DICK|□□□|
|MAYR|◊◊◊◊◊|
|-----|

```

BESIDE PRESENTS TWO STRUCTURES SIDE BY SIDE IN REPORT FORMAT

SYNTAX: Z←A BESIDE B

- CAN FORMAT A REPORT WITH ROW HEADINGS AND A SEPARATOR, OR SIMPLY JOIN TWO DISPARATE MATRICES. SEE VCCAT
- USES: VPREPARE VCOMPARE VIF VADDROWS

FUNCTION:

```
V Z←A BESIDE B;I;J
[1] I←1↑ρA←PREPARE A
[2] J←1↑ρB←PREPARE B
[3] →(L1,L2,L0)IF I COMPARE J
[4] L0:→L2 Δ B←(I-J)ADDROWS B
[5] L1:A←(J-I)ADDROWS A
[6] L2:Z←A,'|',B
V
```

EXAMPLES:

```
□←LM←' 'SHAPE 'SIMPSON GERONIMO JONES LEGRAND'
SIMPSON
GERONIMO
JONES
LEGRAND
```

```
AB←3 4 5 6
□←NM←?4 5ρ1000
589 931 847 527 92
654 416 702 911 763
263 48 737 329 633
757 992 366 248 983
LM BESIDE NM
SIMPSON | 589 931 847 527 92
GERONIMO | 654 416 702 911 763
JONES | 263 48 737 329 633
LEGRAND | 757 992 366 248 983
```

```
AB BESIDE NM
3| 589 931 847 527 92
4| 654 416 702 911 763
5| 263 48 737 329 633
6| 757 992 366 248 983
3 BESIDE NM
| 589 931 847 527 92
| 654 416 702 911 763
| 263 48 737 329 633
3| 757 992 366 248 983
```

ANALYSIS:

- A LINES 1 AND 2 PREPARE THE ARGUMENTS FOR SIDE BY SIDE PLACEMENT.
- A NUMERIC ARGUMENTS ARE FORMATTED WITH AUTOMATIC WIDTH AND
- A NO DECIMAL POSITIONS IF INTEGER, 2 DECIMAL POSITIONS OTHERWISE.
- A ANY FRAMING ALREADY PART OF A CHARACTER ARGUMENT IS REMOVED.
- A VECTOR OR SCALAR ARGUMENTS ARE CONVERTED INTO ONE-ROW MATRICES.
- A LINES 3,4,5 CHECK THE NUMBER OF ROWS IN BOTH ARGUMENTS AND ADD THE
- A APPROPRIATE BLANK ROWS TO PAD OUT THE SMALLER.

BLANK DELETE SPECIFIC STRING FROM STRUCTURE [LIM]

SYNTAX: R←STR BLANK A

- IF A REPORT, OR ANY STRUCTURE, CONTAINS UNWANTED ITEMS, NUMERIC OR CHARACTER STRINGS, THEN EVERY APPEARANCE OF THE SPECIFIED STRING WILL BE REPLACED BY BLANKS, OR BY ZERO, IF A IS A NUMERIC STRUCTURE.
- USES VLOC V LIM VΔ

FUNCTIONS:

▽ R←STR BLANK A
 [1] A IF STR APPEARS IN A IT WILL BECOME BLANK(OR 0)
 [2] R[(×/ρA)LIM(STR LOC R)◦.+1ρ,STR Δ □IO←0]←0\''ρA Δ R←,A
 [3] R←(ρA)ρR
 ▽

▽ R←N LIM A
 [1] A RETURNS VALUES<N, ONLY
 [2] R←(N>R)/R←,A
 ▽

EXAMPLES:

B←□←2▴2|16
 0.00 1.00 0.00 1.00 0.00 1.00
 '0.00'BLANK B
 1.00 1.00 1.00
 B←□←3 4ρ17
 0 1 2 3
 4 5 6 0
 1 2 3 4
 0 1 2 BLANK B
 0 0 0 3
 4 5 6 0
 0 0 3 4

ANALYSIS: 'AAA'BLANK'AAABBBAAAA'

[2] R[(×/ρA)LIM(STR LOC R)◦.+1ρ,STR Δ □IO←0]←0\''ρA Δ R←,A

AAA
 [2] R[(×/ρA)LIM(STR LOC R)◦.+1ρ,STR Δ □IO←0]←0\''ρA Δ R←,A

0 1 2
 [2] R[(×/ρA)LIM(STR LOC R)◦.+1ρ,STR Δ □IO←0]←0\''ρA Δ R←,A

AAABBBAAAA
 [2] R[(×/ρA)LIM(STR LOC R)◦.+1ρ,STR Δ □IO←0]←0\''ρA Δ R←,A

0 6 7 8
 [2] R[(×/ρA)LIM(STR LOC R)◦.+1ρ,STR Δ □IO←0]←0\''ρA Δ R←,A

0 1 2 (NOTICE WRAP-AROUND...9,10 FALSE INDICES)
 6 7 8
 7 8 9
 8 9 10
 [2] R[(×/ρA)LIM(STR LOC R)◦.+1ρ,STR Δ □IO←0]←0\''ρA Δ R←,A

9
 [2] R[(×/ρA)LIM(STR LOC R)◦.+1ρ,STR Δ □IO←0]←0\''ρA Δ R←,A

0 1 2 6 7 8 7 8 8 (FALSE INDICES DELETED)

CENTERON CENTERS AND CATENATES TWO STRUCTURES [CENTER]

SYNTAX: Z←A CENTERON B

- TAKES TWO CHARACTER ARRAYS OF ANY SIZE, CENTERS THE ONE WITH FEWER COLUMNS, THEN CATENATES A ABOVE B.
- USES: ∇IF ∇COMPARE ∇ADDCOLS ∇CENTER ∇ON ∇PREPARE

FUNCTIONS:

```

      ∇ Z←A CENTERON B;I
[1]   A←∇A Δ B←∇B
[2]   →(L1,L3,L2)IF 0 COMPARE I←(¯1↑ρA)-¯1↑ρB
[3]   L1:→L3 Δ B←CENTER I ADDCOLS PREPARE B
[4]   L2:A←CENTER(-I)ADDCOLS PREPARE A
[5]   L3:Z←A ON B
      ∇
                                ∇ Z←CENTER B;C
[1]   C←NUMBLANKCOLS Z←B
[2]   Z←(↑0.5×-/C)ΦZ
      ∇
```

EXAMPLES:

```

      LV←'THIS IS A LITERAL VECTOR'

      LV
THIS IS A LITERAL VECTOR

      LV CENTERON 'HELLO'
THIS IS A LITERAL VECTOR
      HELLO

      'SALES' CENTERON 'REPORT FOR OCTOBER'
      SALES
REPORT FOR OCTOBER
```

ANALYSIS:

```

A      LINE 1 FORCES CHARACTER REPRESENTATION.
A      LINE 2 CHECKS THE NUMBER OF COLUMNS IN THE ARGUMENTS.
A      IF EQUAL, LINE 5 USES ∇ON TO CATENATE THEM.
A      IF UNEQUAL, THE SMALLER IS PADDED OUT ON THE LEFT TO THE WIDTH OF
A      THE LARGER WITH ∇ADDCOLS, THEN USES ∇CENTER TO SPLIT THE NUMBER
A      OF BLANK COLUMNS, PUTTING HALF ON THE RIGHT, BEFORE CATENATING.
A      ∇PREPARE MAKES ONE-ROW MATRICES OF VECTOR AND SCALAR ARGUMENTS.
```

CITED

EXTRACT CITED STRINGS FROM CHARACTER ARRAYS

SYNTAX: R←KV CITED A

- STRINGS OF NON-BLANK INFORMATION, DEMARKED BY THE CHARACTERS PROFFERED AS KV, WILL BE EXTRACTED, SHAPED AS A MATRIX, AND SORTED.
- USES: ∇SHAPE ∇GRADEUP (AND GLOBAL VARIABLE "AV")

FUNCTION:

```
∇ R←KV CITED A
[1] R←' 'SHAPE(~R∈KV)/R←(≠\R∈KV)/R←,A
[2] R←R[AV GRADEUP R;]
∇
```

EXAMPLES:

```
'[]'CITED'TOM [DICK] HARRY'
DICK
```

```
'→←'CITED'TOM→HARRY DICK←JANE'
DICK
HARRY
```

ANALYSIS:

```
'[]'CITED'TOM [DICK JANE] HARRY'
[1] R←' 'SHAPE(~R∈KV)/R←(≠\R∈KV)/R←,A
TOM [DICK JANE] HARRY
[1] R←' 'SHAPE(~R∈KV)/R←(≠\R∈KV)/R←,A
OR ANY CHARACTER NOT IN TEXT
[]
[1] R←' 'SHAPE(~R∈KV)/R←(≠\R∈KV)/R←,A
LOCATED
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[1] R←' 'SHAPE(~R∈KV)/R←(≠\R∈KV)/R←,A
EXTRACTED
[DICK JANE]
[1] R←' 'SHAPE(~R∈KV)/R←(≠\R∈KV)/R←,A
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[1] R←' 'SHAPE(~R∈KV)/R←(≠\R∈KV)/R←,A
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1] R←' 'SHAPE(~R∈KV)/R←(≠\R∈KV)/R←,A
CLEANED AND SHAPED,
DICK READY FOR SORTING
JANE
```

COLLECT

COLLECT AND SUMMARIZE COEFFICIENTS OF COMMON CODES

SYNTAX:

R←VV COLLECT CV

- VV AND CV ARE A PAIR OF NUMERIC VECTORS TO WHICH WE HAD BEEN CATENATING PAIRS OF CORRESPONDING DATA.
- WE DESIRE TO REDUCE THE DATA BY COLLECTING AND SUMMING THOSE TERMS IN VV THAT ARE IN THE SAME CATEGORY, AS REPRESENTED BY REPETITION IN CV, THE CODE VECTOR.
- THE CODE VECTOR, ALTHOUGH NUMERIC, MAY REPRESENT ALPHABETIC CHARACTERS. SEE ∇ LJNFORM IN KWIC INDEX.
- USES ∇ DREP.

FUNCTION:

```

       $\nabla$  R←VV COLLECT CV;T
[1]   R←T,[ $\square$ IO+0.5]((T←DREP CV)◦.=CV)+.×VV
```

EXAMPLE:

```

       $\square$ ←I←?11p6
3 1 4 5 2 3 1 4 5 2 3      (A RANDOM PATTERN OF OCCURRENCE)
       $\square$ ←VV←11?99
98 99 23 27 68 58 32 93 79 17 29 (TALLIES, CORRESPONDING TO CAR-CODES)
      (AV KFORM, 0-1↓R), $\nabla$ 0 1↓R←VV COLLECT AV LJNFORM CARS[I;]
PONTIAC      185
BUICK        131
CHEVROLET    116      (RE-TRANSLATED, FOR REPORTING PURPOSE)
CADILLAC     106
OLDSMOBILE   85
```

ANALYSIS:

VV COLLECT CV←?11p10

```
[1]   R←T,[ $\square$ IO+0.5]((T←DREP CV)◦.=CV)+.×VV
```

```
7 16 47 90 89 58 30 9 24 74 35      VALUES
```

```
[1]   R←T,[ $\square$ IO+0.5]((T←DREP CV)◦.=CV)+.×VV
```

```
10 1 1 6 8 1 5 1 5 8 7      CODES
```

```
[1]   R←T,[ $\square$ IO+0.5]((T←DREP CV)◦.=CV)+.×VV
```

```
10 1 6 8 5 7      REPLICATES DELETED
```

```
[1]   R←T,[ $\square$ IO+0.5]((T←DREP CV)◦.=CV)+.×VV
```

```
1 0 0 0 0 0 0 0 0 0 0
```

```
0 1 1 0 0 1 0 1 0 0 0
```

```
0 0 0 1 0 0 0 0 0 0 0
```

```
0 0 0 0 1 0 0 0 0 1 0
```

```
0 0 0 0 0 0 1 0 1 0 0
```

```
0 0 0 0 0 0 0 0 0 0 1
```

```
[1]   R←T,[ $\square$ IO+0.5]((T←DREP CV)◦.=CV)+.×VV
```

```
10      7
```

```
1      130
```

```
6       90
```

```
8      163
```

```
5       54
```

```
7       35
```

HITS LOCATED

REDUCED AND LAMINATED

DREP SELECT UNIQUE ELEMENTS FROM ANY STRUCTURE

SYNTAX: R←DREP V

USEFUL IN PREPARING SPECIAL COLLATING SEQUENCES.
RESULT CONTAINS SINGLE APPEARANCE OF EACH ELEMENT,
NOT REARRANGED.

FUNCTION:

 ▽ R←DREP V
[1] R←((V1V)=1pV)/V←,V
 ▽

EXAMPLE:

 DREP 1 2 3 4 5 4 3 2 1
1 2 3 4 5

ANALYSIS:

 DREP'MISSISSIPPI'

[1] R←((V1V)=1pV)/V

MISSISSIPPI

[1] R←((V1V)=1pV)/V

MISSISSIPPI

[1] R←((V1V)=1pV)/V

11

[1] R←((V1V)=1pV)/V

1 2 3 4 5 6 7 8 9 10 11

[1] R←((V1V)=1pV)/V

MISSISSIPPI

[1] R←((V1V)=1pV)/V

1 2 3 3 2 3 3 2 9 9 2

[1] R←((V1V)=1pV)/V

1 1 1 0 0 0 0 0 1 0 0

[1] R←((V1V)=1pV)/V

MISP

HEADERON PUTS A HEADING ON A REPORT [COMPARE]

SYNAX: Z←A HEADERON B

- CENTERS A CHARACTER HEADING (LEFT ARG.) ON A REPORT (RIGHT ARG.), WITH AN EXTRA ROW OF BLANKS AND A SEPARATOR. HEADING MAY BE SCALAR, VECTOR OR MATRIX.
- USES: ∇PREPARE ∇IF ∇CENTER ∇ADDCOLS ∇ADDDROWS ∇ADJUSTUP ∇ADJUSTDOWN ∇COMPARE ∇Δ

FUNCTIONS:

<pre> ∇ Z←A HEADERON B;I;J;□IO [1] I←1↑ρA←PREPARE A Δ □IO←1 [2] J←1↑ρB←PREPARE B [3] →(L1,L2,L0)IF I COMPARE J [4] L0:→L2,ρB←CENTER(J-I)ADDCOLS B [5] L1:A←CENTER(I-J)ADDCOLS A [6] L2:Z←A,[1]'_',[1]1 ADDROWS B [7] Z←(2+1↑ρA)ADJUSTUP(1+1↑ρA)ADJUSTDOWN Z ∇ </pre>	<pre> ∇ Z←A COMPARE B [1] Z←(×B-A)Φ0 1 0 ∇ </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------

EXAMPLE:

'SALES' HEADERON ?4 5ρ100
SALES

```

-----
52 32 99 50 27
10 95 8 51 39
28 92 53 47 95
6 77 78 83 13

```

(2 8ρ'DECEMBER REPORT ') HEADERON ?4 5ρ1000
DECEMBER
REPORT

```

-----
16 689 869 630 737
726 1000 889 234 307
352 514 592 846 413
842 270 416 538 468

```

- ∇ LINES 1 AND 2 PREPARE THE HEADING BY CONVERTING SCALARS AND VECTORS TO ONE-ROW MATRICES AND REMOVING ANY PREEXISTING FRAMING ELEMENTS.
- ∇ THEY ALSO FORMAT THE RIGHT ARGUMENT.
- ∇ LINES 3,4,5 CHECK THE WIDTHS OF THE ARGUMENTS AND CENTER THE HEADING
- ∇ LINE 6 ADDS THE SEPARATOR '_' AND AN EXTRA BLANK ROW.
- ∇ LINE 7 ADDS THE FRAMING ELEMENT '|' WHERE NEEDED IN THE EXTRA BLANK
- ∇ ROW TO PRETTY UP THE REPORT. MOST OF THE TIME IT WON'T BE NEEDED.

LJUST LEFT JUSTIFY ANY ARRAY [DLB RJUST DL]

SYNTAX: R←V LJUST A

- TO SHIFT SIGNIFICANT CHARACTERS OR NUMERIC VALUES TO LEFTMOST POSITIONS.
- INSIGNIFICANT VALUES OR CHARACTERS ARE DEFINED IN V. THEY CAN BE TRUNCATED BY DT. (SEE DLB, BELOW)
- RIGHT JUSTIFICATION IS THE REVERSAL OF LEFT JUSTIFICATION.

FUNCTIONS:

<p>∇ R←V LJUST A</p> <p>[1] R←(+/\A∈V)ΦA</p> <p> ∇</p>	<p>∇ R←V RJUST A</p> <p>[1] R←ΦV LJUSTΦA</p> <p> ∇</p>
----------------------------------------------------------------	----------------------------------------------------------------

EXAMPLE:

' 'LJUST □←3 5ρ' TOM DICKHARRY'

TOM
DICK
HARRY

TOM
DICK
HARRY

ANALYSIS: ' *?'LJUST A

[1] R←(+/\A∈V)ΦA

***TOM
**DICK
*HARRY

?BETTY
??MARY
JANE

[1] R←(+/\A∈V)ΦA

?*

[1] R←(+/\A∈V)ΦA

1 1 1 0 0 0
1 1 0 0 0 0
1 0 0 0 0 0

1 0 0 0 0 0
1 1 0 0 0 0
1 1 0 0 0 0

[1] R←(+/\A∈V)ΦA

3 2 1
1 2 2

[1] R←(+/\A∈V)ΦA

TOM***
DICK**
HARRY*

BETTY?
MARY??
JANE

∇ R←QV DL V

[1] R←(~\V∈QV)/V

⌘ WILL DELETE LEADING ELEMENTS
⌘ FROM A VECTOR. QV IS A QUOTED
⌘ CHARACTER STRING OR A NUMERIC
⌘ VECTOR THAT CONTAINS EXAMPLES
⌘ TO BE DELETED.

 ∇

∇ R←DLB K

[1] R←(-(1+ρK),[/,+/\ ' '≠K)↑K

 ∇

⌘ A SPECIAL CASE TO DELETE LEADING BLANKS

ON

CONFORM AND CATENATE ANY STRUCTURES

SYNTAX:

Z←A ON B

- NAMES OR NUMBERS MAY BE ADDED TO LISTS OF ANY SHAPE OR CHARACTER, AT EITHER END. THE OUTPUT IS A MATRIX.
- NUMERIC MATRICES WILL BE PADDED WITH ZEROS IF THEY REMAIN NUMERIC.
- CHARACTER MATRICES WILL BE PADDED WITH BLANKS.
- ANY OPERAND MAY HAVE ANY STRUCTURE.
- USE ∇ CENTERON IF LEFT-JUSTIFICATION IS NOT DESIRED.
- USES: ∇ CFORMAT ∇ MATRIX

FUNCTION:

```

      ▽ Z←A ON B;□IO
[1]   CFORMAT
[2]   Z←(⌈1↑ρA←MATRIX A)⌈⌈1↑ρB←MATRIX B
[3]   Z←(((1↑ρA),Z)↑A),[□IO←0]((1↑ρB),Z)↑B
[4]   Z←⊘'AABZ'[(2×0≠ρ,B)+0≠ρ,A]
[5]   A IN CASE OPERAND WAS EMPTY
      ▽
```

EXAMPLES:

```

      '14'ON 14
14
1 2 3 4
      FRAME 'HEADING'CENTERON 3 4ρ112
|-----|
|  HEADING  |
|  0   1   2   3   |
|  4   5   6   7   |
|  8   9  10  11  |
|-----|

      ' HEADING'ON 3 3ρ19
HEADING
0  1  2
3  4  5
6  7  8
      (BLANKS COUNT AS CHARACTERS)
```


OUTPUT CENTER HEADINGS OVER FORMATTED COLUMNS [CNTR DMZ NEXTA]

SYNTAX: MAT←HEADINGS OUTPUT MATRIX

- HEADINGS, A CHARACTER VECTOR WITH BLANK DELIMITERS, WILL BE CENTERED OVER THE APPROPRIATE COLUMNS, AND UNDERSCORED TO THE FULL WIDTH OF THOSE COLUMNS IN THE PREFORMATTED MATRIX.
- ANY HEADING TOO WIDE FOR ITS COLUMNS WILL BE NOTED.
- AN INSUFFICIENT NUMBER OF HEADINGS WILL CAUSE LENGTH ERROR.
- ONLY THE LEFTMOST HEADINGS WILL BE USED.
- USES: ∇DMZ ∇NEXTA ∇CNTR ∇WIDTH ∇DLB
- GENERATES A GLOBAL VARIABLE, E, WHICH CAN BE USED TO REDUCE (THEN EXPAND) THE RESULT BY ELIMINATING BLANK COLUMNS.

FUNCTIONS:

```

∇ MAT←HEADINGS OUTPUT MATRIX;S;I;A;A;V;W;H;∇IO
[1] MAT←(1+ρMATRIX)ρ' ',S←(I←~∇IO+1)ρA←HEADINGS
[2] W←'E'WIDTH MATRIX
[3] BACK:→((I=ρW)∇0=ρH←NEXTA)/FINAL
[4] →(0≠ρA Δ S←S,A←W[I←I+1]CNTR H)/BACK
[5] →0,ρ∇←'COLUMN WIDTH (',(W),') TOO NARROW FOR ',H
[6] FINAL:MAT[E/1ρE]←S
[7] MAT←(MAT,[0.1]E\'''),[1]MATRIX

```

```

∇ H←W CNTR V
[1] H←(W≥ρV)/(-[0.5×W-ρ,V)φW↑V
∇
∇ R←DMZ N
[1] R←1+(Z∇1φZ←N≠0)/N←,0,N
∇
∇ WORD←NEXTA;L
[1] WORD←(L←^\' '≠A)/A←DLB A
[2] A←(~L)/A
∇

```

EXAMPLES:

A (SEE THE EXAMPLE FOR ∇AMORTIZE)

```

'TOM DICK HARRY ° °'OUTPUT 9 2MM
TOM    DICK    HARRY    °    °
---    ---    ---    ---    ---
1.00   1000.00   87.92   79.58   8.33
2.00    920.42   87.92   80.25   7.67
3.00    840.17   87.92   80.91   7.00

```

```

'ABERCROMBIE DICK HARRY ° °'OUTPUT 9 2MM
COLUMN WIDTH (4 7 5 5 4) TOO NARROW FOR ABERCROMBIE
A      ↑__OFFENDER

```

PAD

PADS ARRAYS WITH BLANKS OR ZEROS

SYNTAX:

$Z \leftarrow P \text{ PAD } X$

- PADS ARRAY X WITH BLANKS (IF LITERAL) OR ZEROS (IF NUMERIC). X CAN BE AN ARRAY OF ANY SHAPE AND TYPE. P IS A NUMERIC VECTOR SPECIFYING THE AMOUNT OF PADDING IN EACH DIMENSION OF X (OR THE LAST ρP).
- THE SENSE OF PADDING (RIGHT OR LEFT), (BOTTOM OR TOP), ETC., IS DETERMINED BY THE SIGNS OF THE ELEMENTS OF P JUST AS WITH THE TAKE FUNCTION (\uparrow). IF P IS NOT LONG ENOUGH TO MATCH THE RANK OF X, PADDING IS DONE ONLY FOR THE LAST ρP DIMENSIONS. THIS FUNCTION IS UNLIKE THE TAKE FUNCTION IN THAT THE CHANGE IN SIZE IS SPECIFIED AND IT DOES NOT REQUIRE DETAILED KNOWLEDGE OF THE DIMENSIONS OF X.

FUNCTION:

$\nabla \quad Z \leftarrow P \text{ PAD } X$
 $[1] \quad Z \leftarrow (((Z=0) \times Z + Z \uparrow P) \times (Z \uparrow 1, \rho X) + | (Z \leftarrow -1 \uparrow \rho \rho X) \uparrow P) \uparrow X$
 ∇

EXAMPLES:

$3 \text{ PAD } 4$ $4 \ 0 \ 0 \ 0$	$-1 \ -20 \text{ PAD } 3 \ 3 \rho 'ABC'$ $-8 \text{ PAD } 'ABCD'$ $ABCD$	ABC ABC ABC
$-3 \text{ PAD } 2 \ 4 \rho 18$ $0 \ 0 \ 0 \ 1 \ 2 \ 3 \ 4$ $0 \ 0 \ 0 \ 5 \ 6 \ 7 \ 8$	$1 \ 2 \text{ PAD } 2 \ 2 \ 4 \rho 16$ $1 \ 2 \ 3 \ 4 \ 0 \ 0$ $5 \ 6 \ 7 \ 8 \ 0 \ 0$ $0 \ 0 \ 0 \ 0 \ 0 \ 0$	
$1 \ -3 \text{ PAD } 2 \ 4 \rho 18$ $0 \ 0 \ 0 \ 1 \ 2 \ 3 \ 4$ $0 \ 0 \ 0 \ 5 \ 6 \ 7 \ 8$ $0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0$	 $9 \ 10 \ 11 \ 12 \ 0 \ 0$ $13 \ 14 \ 15 \ 16 \ 0 \ 0$ $0 \ 0 \ 0 \ 0 \ 0 \ 0$	

ANALYSIS:

$Z \leftarrow (((Z=0) \times Z + Z \uparrow P) \times (Z \uparrow 1, \rho X) + | (Z \leftarrow -1 \uparrow \rho \rho X) \uparrow P) \uparrow X$

SIGN OF PADDING	DIMENSION VECTOR OF X	ABSOLUTE VALUE OF PADDING

PREPARE STANDARDIZE STRUCTURE FOR REPORT FORMATTING [IF]

SYNTAX: Z←PREPARE A

- A SUBROUTINE THAT NORMALIZES DATA FOR THE FORMATTING FUNCTIONS: ▽BESIDE ▽CENTERON ▽HEADERON
- USES: ▽CHARACTER ▽FRAMETEST ▽MATRIX ▽TABULATE ▽IF

FUNCTIONS:

```

      ▽ Z←PREPARE A
[1] →L1 IF ~CHARACTER A
[2] →0 IF 0=FRAMETEST Z←MATRIX A
[3] →0, ρZ←1 1↓~1~1↓Z
[4] L1:Z←TABULATE A
      ▽
      ▽ R← A IF B
[1] R←B/A
      ▽

```

EXAMPLES:

```

      X
ABCD
ABCD
ABCD

      PREPARE X
ABCD
ABCD
ABCD

      ρPREPARE X
3 4
      Y
|-----|
|SALES|
|-----|
      PREPARE Y
SALES
      Z
1 2 3 4
5 6 7 8
9 10 11 12
      PREPARE Z
1 2 3 4
5 6 7 8
9 10 11 12
      ρPREPARE Z
3 12

```

ANALYSIS:

- LINE 1 BRANCHES TO LINE 4 IF THE ARGUMENT IS NUMERIC. 'TABULATE'
- THEN FORMATS IT.
- LINES 2 AND 3 TAKE THE ARGUMENT, WHICH IS MADE INTO A MATRIX, AND
- EXAMINE IT FOR FRAMING ELEMENTS ON THE PERIPHERY OF THE ARRAY.
- THESE ARE REMOVED IF PRESENT.

TABULATE NUMERIC STRUCTURES IN CONTROLLED FORMAT [HANG]

SYNTAX: Z←TABULATE A

- A MUST BE A NUMERIC STRUCTURE. TABULATE WILL DIAGNOSE THE DOMAIN ERROR IF IT IS NOT. THE STATE INDICATOR WILL BE PRESERVED FOR ANALYSIS. THIS IS BETTER THAN TO ESCAPE FROM THE CALLING FUNCTION WITH NO TRACE.
- SEE ∇ESCAPE.
- RETURNS FORMATTED NUMERIC MATRICES.
- ANY FRACTIONS PRESENT WILL FORCE TWO DECIMAL PLACES.
- VECTORS WILL BECOME ONE-COLUMN MATRICES.
- USES: ∇INTEGER ∇MATRIX ∇CHARACTER ∇HANG

FUNCTIONS:

```

      ∇ Z←TABULATE A
[1]   'A IS NOT NUMERIC'HANG CHARACTER A
[2]   Z←(2×~INTEGER A)∇∇MATRIX∇A
      ∇

```

EXAMPLES:

```

      TABULATE 2 3p16
1 2 3
4 5 6
      TABULATE 2 3p016
3.14 6.28 9.42
12.57 15.71 18.85
      TABULATE 13
1
2
3

```

```

      TABULATE 'ABC'
A IS NOT NUMERIC
HANG[5]

```

```

      ∇ MSG HANG CONDITION
[1]   SΔHANG←0
[2]   →0 IF~CONDITION
[3]   MSG
[4]   SΔHANG←□LC+1
[5]   A INSPECT STATE INDICATOR
[6]   A CHECK DOMAIN OF VALUES
      ∇

```

THE PROGRAM HAS STOPPED FOR INSPECTION
OF THE INPUT DATA AND ITS POSSIBLE SOURCE.

Section III

Workspace Management Functions

LISTFN

LISTS A FUNCTION IN STANDARD FORM

SYNTAX:

R←LISTFN FN

- RETURNS A CHARACTER MATRIX WHICH APPEARS TO BE A LISTING OF THE FUNCTION WHOSE NAME, IN QUOTES, IS THE ARGUMENT. IT CONTAINS ∇'S, STATEMENT NUMBERS, AND EXTENDED LABELS AND COMMENTS.
- AN EMPTY MATRIX IS RETURNED FOR NON-EXISTENT AND LOCKED FUNCTIONS.

FUNCTION:

```
∇ ZQQ←LISTFN XQQ;FQQ;IO
[1]  IO←1
[2]  →(0=1↑ρZQQ←CR XQQ)↑0
[3]  ZQQ←2(⊖FX 5 0+CR'LISTFN'),' ZQQ'
[4]  →0
[5]  Z←FQQ A;B;N
[6]  N←1+ρB←(A[;1]='A')∇B\(+∕∇B≠Z)>+∕∇'''=(B+∇/Z←A=':')≠A
[7]  Z←N↑((N[9]ρ2),(0[90[N-9]ρ1
[8]  Z←(('',[1]')['',Zφ(3 0∇(N,1)ρ1N),'']),Bφ' ',A),[1]''
[9]  Z[1,N+2;5]←'∇'
∇
```

EXAMPLE:

```
∇TEST[[]]∇
∇ TEST
[1]  A←13
[2]  ACOMMENT
[3]  L1:'END'
∇

Q←Q←LISTFN 'TEST'
∇ TEST
[1]  A←13
[2]  ACOMMENT
[3]  L1:'END'
∇

ρQ
5 14
```

NOTES: THE TECHNIQUE USED HERE ILLUSTRATES A METHOD OF DEALING WITH THE PROBLEM OF NAME-SHADOWING IN APL: ACCESS TO GLOBAL OR SEMI-GLOBAL OBJECTS IS INHIBITED IF IDENTICAL LOCAL NAMES OCCUR IN AN ACTIVE FUNCTION. TO AVOID THIS IN SITUATIONS WHERE ACCESS TO GLOBAL OBJECTS IS NECESSARY, SOME PROGRAMMERS USE HIGHLY IMPROBABLE NAMES SUCH AS THE ZQQ, XQQ AND FQQ USED HERE. THIS IN TURN MAKES THE CODE HARDER TO UNDERSTAND AND MAINTAIN. THE METHOD USED HERE IS A COMPROMISE. IMPROBABLE NAMES ARE USED TO OBTAIN DATA TO PASS TO A LOCAL FUNCTION WITH "GOOD" NAMES. LINE 3 CREATES AND EXECUTES THIS LOCAL FUNCTION, WHICH IS "DEFINED" IN LINES 5 THROUGH 9 OF LISTFN.

PROMPT PROMPT AND CHECK TERMINAL ENTRY OR ELSE DEFAULT

SYNTAX: R←DEF PROMPT MSG

- PROMPT CAN EXAMINE KEYBOARD ENTRY, AND EITHER ACCEPT IT UNCRITICALLY, RETURN A SPECIFIED DEFAULT CHARACTER VECTOR, OR, IF NUMERIC VALUES ARE REQUESTED, WILL CHECK THE CHARACTER SET USED, AND CAN CHECK FOR THE SPECIFIED LENGTH.
- MSG IS A CHARACTER STRING, I.E., THE PROMPTING MESSAGE
- DEF, IF CHARACTER, IS THE DEFAULT CHARACTER STRING, OR '' IF A SCALAR NUMERIC VALUE, N:
 - WILL ACCEPT ANY NUMERIC VECTOR IF N=0
 - WILL REJECT VECTOR UNLESS N=ρVECTOR
 - IF AN INTEGER VECTOR, V, WILL ACCEPT A NUMERIC VECTOR IF ITS LENGTH IS ONE MEMBER OF V.
- USES: ∇CHARACTER ∇EMPTY ∇IF ∇Δ

FUNCTION:

```
∇ R←DEF PROMPT MSG;J;K
[1]  →0 IF 0=ρ,DEF Δ J←R←R+∇Δ R←ρ,∇←MSG
[2]  →0 IF CHARACTER DEF Δ R←R,(0=ρ,R)/∇DEF
[3]  NSCREEN:→ERR IF~(∇EMPTY K)^^/K←Jε' -.0123456789E'
[4]  →0 IF∇/DEF=(DEF>0)×ρ,R←∇J
[5]  ERR:→1 Δ ∇←'NOT EXACTLY ',((∇/DEF>0)/∇DEF`,` NUMBER`,`(∇/DEF>1)/`S'
[6]  ρ CHARACTERS MAY BE ADDED OR DELETED FROM NSCREEN
∇
```

EXAMPLES:

```
OPROMPT'THE AGES OF THE MEMBERS OF YOUR FAMILY, IN DESCENDING ORDER..'
THE AGES OF THE MEMBERS OF YOUR FAMILY, IN DESCENDING ORDER..54 43 22 16
54 43 22 16
3PROMPT'THE MONTH DAY YEAR OF YOUR BIRTH, AS NUMBERS..'
THE MONTH DAY YEAR OF YOUR BIRTH, AS NUMBERS..3 13
NOT EXACTLY 3 NUMBERS
THE MONTH DAY YEAR OF YOUR BIRTH, AS NUMBERS..3 13 1954
3 13 1954
2PROMPT'GIVE ME TWO NUMBERS..'
GIVE ME TWO NUMBERS..-14 1.234E56
-14 1.234E56
1PROMPT'AN EXAMPLE OF A SCALAR NUMERIC VALUE: '
AN EXAMPLE OF A SCALAR NUMERIC VALUE: ASDF
NOT EXACTLY 1 NUMBER
AN EXAMPLE OF A SCALAR NUMERIC VALUE: -1
-1
1PROMPT'ANOTHER...'
ANOTHER...-0
0
'NO RESPONSE'PROMPT'WHAT IS YOUR NAME? '
WHAT IS YOUR NAME?
NO RESPONSE
2 3 PROMPT'GIVE ME TWO OR THREE NUMBERS...'
GIVE ME TWO OR THREE NUMBERS...77
NOT EXACTLY 2 3 NUMBERS
GIVE ME TWO OR THREE NUMBERS...77 6
77 6
```

STATUS *CURRENT SESSION AND WORKSPACE STATUS* [*NOW*]

SYNTAX: *STATUS*

• *DISPLAYS DATE, TIME, CPUTIME, AVAILABLE SPACE, SUSPENSIONS*

FUNCTIONS:

```

      ▽ STATUS
[1]   NOW
[2]   'CPUTIME THIS SESSION: ',(0.001×1+1+AI), ' SECONDS''
[3]   'SPACE LEFT: ',(WA), ' BYTES''
[4]   'FUNCTIONS SUSPENDED: ',1+ρLC'
      ▽
          ▽ R←NOW
[1]   R←('/'FILLS100|1φ3+TS), ' ',': 'FILLS 1+3+TS'
          ▽
```

EXAMPLES:

```

      STATUS
7/7/77 17:20:20
CPUTIME THIS SESSION: 8.117 SECONDS
SPACE LEFT: 404708 BYTES
FUNCTIONS SUSPENDED: 0
```

TABS COMPARE REQUIRED TAB SETTINGS TO EXISTING ONES [COLNO]

SYNTAX: TABS

- AN UPCOMING REPORT MAY REQUIRE SPECIAL TAB SETTINGS.
- THE TERMINAL USER CAN BE PROMPTED TO VERIFY THE CORRECTNESS OF THE EXISTING TABS BEFORE CONTINUING WITH THE REPORT.
- SEE VTCC, PAGE 44

FUNCTIONS:

```

▽ TABS;X;Y
[1]  A VISUAL CHECK OF REQUIRED AND ACTUAL TABS
[2]  'NONE REQUIRED.' ESCAPE EMPTY □HT, □IO←0
[3]  'IMPOSSIBLE. CORRECT □HT OR □PW' ESCAPE~^/(□PW≥□HT), □HTε\129
[4]  'TABS REQUIRED: (LEFT MARGIN AT ZERO)'
[5]  '.*' [X←(1Y+1+[/□HT)ε□HT]
[6]  0 COLNO Y
[7]  (1+X)\'↑'
[8]  5pIDLC
[9]  'EXISTING TABS, INDICATED BY ↑. CLEAR AND SET AS REQUIRED.'
▽
      ▽ R←FROM COLNO TO;□IO
[1]  R← 1 0 ▽ 10 10 ▽FROM+1(129[TO)-FROM--~□IO←0
      ▽

```

EXAMPLES:

```

□HT←''
TABS
NONE REQUIRED.
□HT←11 33 55
TABS
TABS REQUIRED: (LEFT MARGIN AT ZERO)
.....↑.....↑.....↑
00000000001111111111222222222233333333334444444444555555
01234567890123456789012345678901234567890123456789012345
      ↑      ↑      ↑
EXISTING TABS, INDICATED BY ↑. CLEAR AND SET AS REQUIRED.

```

SYNTAX: Z←BKSP Z←CRTN Z←IDLC Z←LNFD Z←TABC

- Z IN EACH CASE IS ONE BACKSPACE, CARRIAGE RETURN, IDLE, LINE FEED, OR TAB CHARACTER, IF THAT CHARACTER EXISTS IN THE ATOMIC VECTOR (⌈AV) IN THE APL SYSTEM YOU ARE USING. THE CHARACTERS ARE STORED AS THE GLOBAL VARIABLES BK, CR, ID, LN, AND TB FOR RAPID ACCESS. THESE FUNCTIONS DEPEND ON THE SPECIFIC CONFIGURATION OF ⌈AV IN THE VARIOUS APL SYSTEMS TO DATE. EACH FUNCTION ASSUMES THAT AT LEAST ONE ELEMENT OF ⌈AV IS UNIQUE TO EACH SYSTEM.
- KNOWN EXCEPTIONS: THE TAB AND IDLE CHARACTERS DO NOT EXIST IN VS APL ⌈AV. IDLC WILL DELIVER THE ◦ CHARACTER; TABC THE →.

FUNCTIONS:EXAMPLES:

<p>∇ Z←BKSP</p> <p>[1] →(0≠⌈NC'<u>BK</u>')/A1</p> <p>[2] <u>BK</u>←⌈AV[⌈IO+158 200 41['⊖I'⌈⌈AV[⌈IO+73]]]]</p> <p>[3] A1:Z←<u>BK</u></p> <p>∇</p>	<p>'⌈',BKSP,'÷'</p> <p>⊖</p>
<p>∇ Z←CRTN</p> <p>[1] →(0≠⌈NC'<u>CR</u>')/A1</p> <p>[2] <u>CR</u>←⌈AV[⌈IO+156 202 73['⊖I'⌈⌈AV[⌈IO+73]]]]</p> <p>[3] A1:Z←<u>CR</u></p> <p>∇</p>	<p>'A',CRTN,'B'</p> <p>A</p> <p>B</p>
<p>∇ Z←IDLC</p> <p>[1] →(0≠⌈NC'<u>ID</u>')/A1</p> <p>[2] <u>ID</u>←⌈AV[⌈IO+163 191 64['⊖I'⌈⌈AV[⌈IO+73]]]]</p> <p>[3] A1:Z←<u>ID</u></p> <p>∇</p>	<p>'A',IDLC,'B'</p> <p>AB</p>
<p>∇ Z←LNFD</p> <p>[1] →(0≠⌈NC'<u>LN</u>')/A1</p> <p>[2] <u>LN</u>←⌈AV[⌈IO+159 201 169['⊖I'⌈⌈AV[⌈IO+73]]]]</p> <p>[3] A1:Z←<u>LN</u></p> <p>∇</p>	<p>'A',LNFD,'B'</p> <p>A</p> <p>B</p>
<p>∇ Z←TABC</p> <p>[1] →(0≠⌈NC'<u>TB</u>')/A1</p> <p>[2] <u>TB</u>←⌈AV[⌈IO+162 185 192['⊖I'⌈⌈AV[⌈IO+73]]]]</p> <p>[3] A1:Z←<u>TB</u></p> <p>∇</p>	<p>'A',TABC,'B'</p> <p>A</p> <p>B</p>

TIME RUNNING TIME AND NEW SPACE FOR AN APL STATEMENT [ALT]

SYNTAX: TIME STMT

- EXECUTES THE APL STATEMENT IN THE QUOTED CHARACTER STRING, STMT. DISPLAYS ITS RUNNING TIME AND NEW SPACE.
- DIFFERENT WAYS OF PERFORMING THE SAME FUNCTION OR EQUIVALENT OPERATION CAN BE COMPARED.
- LANGUAGE ELEGANCE, CLARITY, AND MAINTAINABILITY SHOULD BE WEIGHED IN JUDGMENT AGAINST TIME AND SPACE USAGE.
- ZERO SPACE MAY BE REPORTED IF THE VARIABLES HAD BEEN NAMED EARLIER IN THIS WORKSPACE AND WERE SUFFICIENTLY LARGE AT THAT TIME.
- TRAFFIC ON THE HOST MACHINE MAY CAUSE TIMING TO VARY SLIGHTLY. TIME YOUR STATEMENTS MORE THAN ONCE, THEN AVERAGE THE RESULTS.
- TEMPORARY STORAGE IS NOT MEASURED, ALTHOUGH IT MAY BE SIGNIFICANTLY LARGE, E.G. OUTER PRODUCTS.

FUNCTIONS:

```
▽ TIME STMT;F;Z
[1]  HOW MUCH TIME/SPACE DOES AN APL STATEMENT USE?
[2]  'F'ALT STMT
[3]  Z←(1↑1+□AI),□WA
[4]  F
[5]  (F-7+(1↑1+□AI)-1↑Z), ' MSEC ', (F-1↑Z)-48+□WA, ' BYTES'
[6]  A ↑ CHANGE TO -6, FOR EXAMPLE, IF ADJUSTMENT NEEDED.
▽
```

```
▽ NAME ALT EXP;J
[1]  J←□FX NAME ON EXP
[2]  AINTRANSITIVE SYNONYM
▽
```

EXAMPLES:

```
TIME 'Z←11000'
1 MSEC 24 BYTES
TIME 'Z←Z-7'
0 MSEC 0 BYTES
A NOTICE THAT SOME APL SYSTEMS KEEP AN ARITHMETIC PROGRESSION VECTOR
A STORED IN COMPACT FORM AS LONG AS POSSIBLE. WHEN WE SQUARE IT:
TIME 'A←Z*2'
8 MSEC 392 BYTES
A ...WE SUDDENLY SEE STORAGE BEING ALLOCATED
A FOR IT SINCE IT CAN NO LONGER BE STORED MERELY AS
A STARTING POINT, STEP, AND LENGTH. FOR TIMING, COMPARE:
TIME 'B←Z×Z'
10 MSEC 392 BYTES
A SURPRISINGLY, Z×Z SEEMS TO TAKE LONGER THAN Z*2.
A TIMING AND STORAGE COMPARISONS MAY WELL BE DIFFERENT
A ON DIFFERENT APL SYSTEMS. THE 5110 DOES NOT SUPPORT □AI.
TIME ''
0 MSEC 0 BYTES
A IF THE RESULTS OF TESTING THE PREVIOUS NULL STATEMENT WERE
A NOT 0 0, ADJUST THE CONSTANTS IN LINE 5 OF THE APL FUNCTION.
```

VARS *DISPLAY CHARACTERISTICS OR CONTENTS OF VARS SELECTIVELY*

SYNTAX: $R \leftarrow B$ VARS K

- IN THE AUTOMATIC PRODUCTION OF APPLICATION DOCUMENTATION, IT IS OFTEN DESIRED TO DISPLAY ALL VARIABLES, OR ONLY THOSE OF PARTICULAR INITIAL CHARACTERS.
 - WHEN THESE STRUCTURES ARE TOO LARGE TO BE DISPLAYED, THEIR CHARACTERISTICS ONLY, MAY BE REQUESTED.
- 1 VARS'' WILL DISPLAY EVERYTHING
- 1 VARS'' 'Z'' WILL DISPLAY ALL VARIABLES WITH INITIAL Z
- 0 VARS'' WILL DISPLAY CHARACTERISTICS ONLY OF ALL VARS
- 0 VARS'' 'XYZ'' FOR CHARACTERISTICS OF VARS INITIALLY ∈ 'XYZ'
- USES: ∇DLTMB ∇IF ∇LOGICAL ∇INTEGER ∇FLOATING ∇CHARACTER ∇EMPTY ∇GRADEUP ∇IS ALF

FUNCTION:

```

∇ R←B VARS K;I;J;IO
[1]  A LOCAL OR GLOBAL VARIABLES AND THEIR CHARACTERISTICS
[2]  →(0=1↑ρR←R[⊡'ALF GRADEUP R←','K','⊡NL 2';])/~IO←1
[3]  K←8 10↑CR'VARS'
[4]  BACK:I←(14)IF(LOGICAL J),(INTEGER J),(FLOATING J),CHARACTER J←⊡R[1;]
[5]  ''
[6]  →EMP IF EMPTY J
[7]  R[1;], ' IS ',DLTMB(,K[I;]),(,K[IO+4+ρρJ;]),(0≠ρρJ) / 'OF SHAPE ',ρρJ
[8]  ⊡B/R[1;]
[9]  FWD:→(0≠1↑ρR←1 0↓R)φ0,BACK
[10] EMP:→FWD,ρ⊡←R[1;], ' IS EMPTY'
[11] A THE FOLLOWING IS NEVER EXECUTED.
[12] LOGICAL
[13] INTEGER
[14] FLOATING
[15] CHARACTER
[16] SCALAR
[17] VECTOR
[18] MATRIX
[19] ARRAY
∇

```

EXAMPLE:

0 VARS'' 'RST''

R IS CHARACTER MATRIX OF SHAPE 47 76

T IS CHARACTER VECTOR OF SHAPE 12

TT IS EMPTY

Section IV

Multiprecision Arithmetic Functions

ADD MULTIPRECISION INTEGER ADDITION

SYNTAX: C←A ADD B

- ADD A TO B, GIVING C, USING MULTIPRECISION INTEGER ARITHMETIC (I.E. TRUNCATING ALL RESULTS TO MP INTEGERS)
- A AND B ARE NUMERIC VALUES.
THEY MAY BE: INTEGER OR FLOATING POINT, OF EITHER ORDINARY OR EXTENDED PRECISION.
- C WILL BE A MULTIPRECISION INTEGER (ZERO EXPONENT)
- AN EXTENDED PRECISION NUMBER, INTEGER OR FLOATING POINT, IS REPRESENTED BY AN INTEGER VECTOR OF CONSECUTIVE 1-7 DECIMAL DIGIT SEGMENTS, THE FIRST OF WHICH IS THE EXPONENT, ZERO FOR INTEGERS. ALL SEGMENTS HAVE THE SAME SIGN, WITH THE EXCEPTION OF THE EXPONENT.(SEE ∇FADD). LEADING ZEROS ARE ELIMINATED.
- USE ∇FIX TO CONVERT A NUMERIC VARIABLE TO MP INTEGER FORM.
- THE MULTIPRECISION INTEGER ARITHMETIC PACKAGE COMPRISES:
∇ADD ∇SUB ∇MUL ∇DIV ∇SQRT ∇FIX ∇CAN ∇ROUNDS
- FUNCTIONS OF THE MULTIPRECISION INTEGER ARITHMETIC PACKAGE ARE USED BY THE MULTI-PRECISION FLOATING POINT ARITHMETIC PACKAGE
- USES: ∇FIX ∇CAN

FUNCTION:

```

      ∇ C←A ADD B
[1]  A MULTIPRECISION INTEGER ADD
[2]  C←CANφ(C↑φA)+(C←(ρA←FIX A)↑φB)↑φB←FIX B
      ∇

```

EXAMPLES:

```

      A
0 1 2345678 9012345 6789012
      B
0 222 3333333 4444444
      A ADD B
0 1 2345901 2345679 1233456
      A ADD -3
0 1 2345678 9012345 67890.09
      0 1 0 0 0 ADD -1
0 9999999 9999999 9999999

```

ALPREC

ALTER PRECISION OF A SCALAR OR MULTIPRECISION NUMBER

SYNTAX:

$Z \leftarrow N$ ALPREC X

- FOR X A SCALAR OR MULTIPRECISION INTEGER OR FLOATING POINT NUMBER OF PRECISION M (I.E. $(\rho X) = M+1$), Z WILL BE SET TO THE FLOATING POINT NUMBER OF PRECISION $M+N$ WHOSE VALUE IS THE SAME AS OR TRUNCATED FROM THAT OF X . (SCALARS HAVE IMPLICIT PRECISION 3.)
- IN BRIEF, X IS EXTENDED ON THE RIGHT WITH ZEROES (IF $N > 0$) OR TRUNCATED ON THE RIGHT (IF $N < 0$). THE EXPONENT $(1 \uparrow X)$ IS INCREASED BY N TO COMPENSATE.
- SINCE ALL THE FUNCTIONS IN THE MULTIPRECISION FLOATING POINT PACKAGE DETERMINE THE PRECISION OF THE RESULT FROM THE PRECISIONS OF THE INPUT, IT IS OFTEN NECESSARY TO INDICATE THE PRECISION OF SOME EXACT QUANTITY (SUCH AS AN INTEGER) SO AS NOT TO CAUSE INVOLUNTARY SHORTENING OF PRECISION. SEE THE EXAMPLES BELOW.
- DECREASING THE PRECISION OF A NUMBER (BY USING NEGATIVE N) MAY OCCASIONALLY BE NECESSARY IF THE RESULT OF AN INTERMEDIATE CALCULATION IS ASSIGNED A SPURIOUSLY HIGH PRECISION BY ONE OF THE ARITHMETIC FUNCTIONS.
- USES: ∇ FLOAT

FUNCTION:

∇ $Z \leftarrow N$ ALPREC X

[1] INCREASE THE PRECISION OF X BY N (OR DECREASE IF N IS NEGATIVE).

[2] $Z \leftarrow (-N - 1 \uparrow X), 1 \uparrow ((\rho, X) + N \leftarrow N[2 - \rho, X] \uparrow X \leftarrow \text{FLOAT } X$

∇

EXAMPLES:

D

~ 5 76543 2109876 5432109 8765432 1098765

FORMAT D

0.0076543 2109876 5432109 8765432 1098765

FORMAT 1 FDIV D

130.6451593 9386058 9254313

A NOTE THE LOSS OF PRECISION.

A INSTEAD, ONE, WHICH WE KNOW IS EXACT, MUST BE ASSIGNED

A A PRECISION AT LEAST AT GOOD AS THAT OF THE NUMBER D .

$\square \leftarrow \text{ONE} \leftarrow 5$ ALPREC 1

~ 7 1 0 0 0 0 0 0 0

$\square \leftarrow Z \leftarrow \text{ONE}$ FDIV D

~ 5 130 6451593 9386058 9254312 9891047 5895018

FORMAT Z

130.6451593 9386058 9254312 9891047 5895018

A SIMILARLY, WE CAN AVOID A LOSS OF PRECISION IN ADDITION, HERE:

$\square \leftarrow Z \leftarrow 1$ FADD D

~ 2 1 76543 2109876

A AND THE RIGHT WAY, USING THE PREPARED ONE:

FORMAT $Z \leftarrow \text{ONE}$ FADD D

1.0076543 2109876 5432109 8765432 1098765

CAN

EDIT MULTIPRECISION INTEGERS INTO CANONICAL FORMAT

SYNTAX:

Z←CAN A

- INPUT A IS EDITED INTO THE CANONICAL MULTIPRECISION INTEGER FORMAT AS DESCRIBED UNDER FUNCTION ADD
- THIS FUNCTION IS USED BY EVERY OTHER FUNCTION IN THE MULTIPRECISION INTEGER AND FLOATING POINT ARITHMETIC PACKAGES, EXCEPT ∇ ROUNDS.
- USES: ∇ ROUNDS

FUNCTION:

```

       $\nabla$  Z←CAN A;S;L
[1]  AEDIT A MULTIPRECISION INTEGER INTO CANONICAL FORM
[2]  Z←1 ROUNDS A
[3]  APROPAGATE CARRIES LEFTWARD
[4]  L1:Z←(S,0)+0,Z-10000000×S←(×Z)×L | Z÷10000000
[5]  →(V/0≠S)/L1
[6]  ADROP LEADING ZEROES (BUT PREVENT 0→10)
[7]  L2:→(1=ρZ←(((0≠1+Z)11)-□IO)÷Z)/L3
[8]  AMAKE ALL TERMS (EXCEPT THE EXPONENT) THE SAME SIGN
[9]  →(~V/S←(-L+1↑S)=S←×Z)/L3
[10] →L2,ρZ←Z+(L×10000000×S)+(1↓-L×S),0
[11] L3:Z←0,Z
       $\nabla$ 
```

EXAMPLES:

```

      A
0 1 2345678 9012345 6789012
      B
0 222 3333333 4444444 5555555
      A+B
0 221 987655 4567901 1233457
      CAN A+B
0 221 987654 5432098 8766543
      CAN 123456789012345
0 1 2345678 9012345
      CAN 7.8
0 8
```

DIV MULTIPRECISION INTEGER DIVISION

SYNTAX: *C←A DIV B*

- *DIVIDE A BY B, GIVING C, USING MULTIPRECISION INTEGER ARITHMETIC. A AND B MAY BE IN ANY NUMERIC FORMAT.*
- *SEE THE DESCRIPTION UNDER VADD.*
- *THE REMAINDER OF THE DIVISION IS LEFT IN GLOBAL VARIABLE REM*
- *USES: VADD VSUB VMUL VFIX VCAN*

FUNCTION:

```

V C←A DIV B;Q;T
[1]  AMULTIPRECISION INTEGER DIVIDE WITH REMAINDER
[2]  →(2≠ρB←FIX B)/L2
[3]  ASPECIAL CODE FOR SPEED IF B IS SCALAR
[4]  →C←(2×(B←1+B)=1↑REM←0,1↑A←FIX A)ρ0
[5]  L1:C←C,Q←⌊(T←10000000⊥REM,1↑A)÷B
[6]  REM←0,T-Q×B
[7]  →(0<ρA←1+A)/L1
[8]  →0,ρC←CAN C
[9]  L2:REM←A←FIX A
[10] →(BA.=' 'ρC←0 0)⊥L4,0
[11] L3:Q←CAN⌊(10000000⊥3↑REM)÷10000000⊥2↑B
[12] C←C ADD Q←(¬1+(ρQ)+(ρREM)-ρB)↑Q
[13] REM←REM SUB B MUL Q
[14] L4:→(×(ρREM)-ρB)↑L3,0
[15] →L3×(|1↑2↑REM)>|1↑2↑B
V

```

EXAMPLES:

```

      A
0 -1 -2345678 -9012345 -6789012
      B
0 222 3333333 4444444
      A DIV B
0 -55528
      REM
0 46 4327157 4297420
      B DIV A
0 0
      REM
0 222 3333333 4444444
      A DIV 0
0 0 (DOMAIN ERRORS PREVENTED, AS IN VCDIV)
      0 1 0 0 0 0 DIV 7
0 1428571 4285714 2857142 8571428
      REM
0 4

```

FADD

MULTIPRECISION FLOATING POINT ADDITION

SYNTAX:

C←A FADD B

- A AND B ARE NUMERIC VARIABLES.
THEY MAY BE: INTEGER OR FLOATING POINT OF EITHER ORDINARY OR EXTENDED PRECISION.
- THE RESULT WILL BE MULTIPRECISION FLOATING POINT.
- A MULTIPRECISION FLOATING POINT NUMBER HAS THE SAME FORMAT AS A MULTIPRECISION INTEGER (I.E. INTEGER VECTOR; SEE DESCRIPTION UNDER VADD). THE LEADING INTEGER (EXPONENT) INDICATES HOW MANY 1-7 DIGIT ELEMENTS FROM THE RIGHT END OF THE NUMBER THE DECIMAL POINT BELONGS. IF NEGATIVE, MOVE ←.
- NUMERIC VARIABLES OF ANY FORMAT MAY BE CONVERTED INTO MP FLOATING POINT BY THE FUNCTION FLOAT.
- THE PRECISION OF A MULTIPRECISION FLOATING POINT NUMBER IS INDICATED BY ITS LENGTH, AND ALL MULTIPRECISION FLOATING POINT OPERATIONS SET THE LENGTH OF THE RESULT ACCORDING TO THE PRECISION OF THE OPERANDS. IN PARTICULAR, THE RESULT OF AN ADD OR SUBTRACT HAS A PRECISION SUCH THAT ITS LEAST SIGNIFICANT ELEMENT IS GOVERNED BY THE SIGNIFICANCE OF THE OPERAND OF GREATER MAGNITUDE.
- THE MULTIPRECISION FLOATING POINT ARITHMETIC PACKAGE COMPRISES: VFADD VFSUB VFMUL VFDIV VFLOAT VFSQRT
VFEXP VPI VALPREC VFORMAT VSCALE
- FADD USES: VFLOAT VADD

FUNCTION:

```

      V C←A FADD B;M
[1]  A MULTIPRECISION FLOATING POINT ADD
[2]  M←⌊C←(1↑A←FLOAT A),1↑B←FLOAT B
[3]  C←(⌈C,1↑(0,1↑(M-1↑C)↑A)ADD 0,1↑(M-1↑C)↑B
      V
```

EXAMPLES:

```

      A
0 47152 2357206
      B
0 222 3333333 4444444 5555555
      A FADD B
0 222 3333333 4491596 7912761
      C←-2 12345 6789012 3456789
      A FADD C
0 47152 2369551
      A NOTE THE TRUNCATION
      A TO SEE THE NUMBERS IN USUAL FORM, USE VFORMAT
      FORMAT A
47152 2357206.
      FORMAT C
12345.6789012 3456789
      FORMAT A FADD C
47152 2369551.
```

FDIV MULTIPRECISION FLOATING POINT DIVISION

SYNTAX: C←A FDIV B

- A, B, AND C ARE MULTIPRECISION FLOATING POINT VALUES.
(SEE DESCRIPTION FOR VFADD)
- THE PRECISION OF THE RESULT IS THE SAME OR SLIGHTLY GREATER THAN THE SMALLER OF THE PRECISIONS OF THE TWO OPERANDS.
- USES: VFLOAT VDIV
- DIVISION BY ZERO WILL RETURN ZERO, AS IN VCDIV.

FUNCTION:

```

V C←A FDIV B;REM
[1]  MULTIPRECISION FLOATING POINT DIVIDE
[2]  C←((1000×1↑1+A)≤1↑1+B)+1+(ρB)+0[(ρB←FLOAT B)-ρA←FLOAT A
[3]  C←((1↑A)-(1↑B)+C),1↑(0,1↑(C+ρA)↑A)DIV 0,1↑B
V

```

EXAMPLES:

```

      B
0 222 3333333 4444444 5555555
      FORMAT B
222 3333333 4444444 5555555.
      C
-2 12345 6789012 3456789
      FORMAT C
12345.6789012 3456789
      FORMAT B FDIV C
180090 0016298 1001574.
      FORMAT (B FMUL C)FDIV B
12345.6789012 3456788 9994910
      HALF
-6 5000000 0 0 0 0 0
      FORMAT HALF
0.5000000 0000000 0000000 0000000 0000000 0000000
      FORMAT B FDIV HALF
444 6666666 8888889 1111110.0000000

```

FEXP MULTIPRECISION FLOATING POINT EXPONENTIAL FUNCTION

SYNTAX: Z←FEXP X

- RETURNS *X AS A MULTIPRECISION FLOATING POINT NUMBER.
- X MAY BE SCALAR OR MULTIPRECISION FLOATING POINT, MP INTEGER BEING A CASE OF FLOATING POINT.
- SEE: ∇ADD ∇FADD
- THE PRECISION OF Z IS CHOSEN TO BE THE SAME AS THAT OF X. USE ∇ALPREC TO INCREASE PRECISION
- USES: ∇FLOAT ∇ADD ∇MUL ∇DIV

FUNCTION:

```

∇ Z←FEXP X;I;N;T
[1]  ∇MULTIPRECISION FLOATING POINT *X BY THE STANDARD POWER SERIES
[2]  Z←T←0,(1-N←1↑X←FLOAT X)↑1+I←0
[3]  X←0,1↓X
[4]  LOOP:Z←Z ADD T←N↓(T MUL X)DIV I←I+1
[5]  →(∇/T≠0)/LOOP
[6]  Z←N,1↓Z
∇

```

EXAMPLES:

```

      ∇CREATE THE CONSTANT 1 BY USE OF FUNCTION ALPREC
      □←ONE←4 ALPREC 1
-6 1 0 0 0 0 0 0
      FORMAT ONE
1.0000000 0000000 0000000 0000000 0000000 0000000
      FEXP ONE
-6 2 7182818 2845904 5235360 2874713 5266249 7757231
      FORMAT FEXP ONE
2.7182818 2845904 5235360 2874713 5266249 7757231
      □←MINUSONE←4 APPREC -1
-6 -1 0 0 0 0 0 0
      FORMAT MINUSONE
-1.0000000 0000000 0000000 0000000 0000000 0000000
      FORMAT FEXP MINUSONE
0.3678794 4117144 2321595 5237701 6146086 7445811

```


FIX *CONVERT TO MULTIPRECISION INTEGER*

SYNTAX: *Z←FIX X*

- *IF X IS SCALAR, IT IS ROUNDED AND CATENATED BEHIND A ZERO.*
- *IF X IS MP INTEGER, IT IS LEFT UNCHANGED.*
- *IF X IS MULTIPRECISION FLOATING POINT, IT IS TRUNCATED TO THE MULTIPRECISION INTEGER WITH THE SAME LEADING SEGMENTS.*
- *SEE THE DESCRIPTIONS FOR: ∇FADD ∇ADD*
- *USES: ∇CAN*

FUNCTION:

```
∇ Z←FIX X
[1]  A←CONVERT A NUMBER TO A MULTIPRECISION INTEGER
[2]  →(1<ρZ+X)/MP
[3]  →0,Z←CAN Z
[4]  MP:→(0=1↑Z)/0
[5]  Z←CAN(0↑-1+(1↑X)+ρX)↑1↑X
∇
```

EXAMPLES:

```
      B
0 222 3333333 4444444 5555555
      FORMAT B
222 3333333 4444444 5555555.
      FIX B
0 222 3333333 4444444 5555555
      C
-2 -12345 -6789012 -3456789
      FORMAT C
-12345.6789012 3456789
      FIX C
0 -12345
      D
-3 76543 2109876 5432109
      FORMAT D
0.0076543 2109876 5432109
      FIX D
0 0

      FIX 7
0 7
      FIX 7.8
0 8
```

FLOAT *CONVERT TO MULTIPRECISION FLOATING POINT [SCALE]*

SYNTAX: *Z←FLOAT X*

- *IF X IS SCALAR, IT WILL BE CONVERTED TO AN MP FLOATING POINT NUMBER OF PRECISION 3.*
- *SEE THE DESCRIPTION OF MP FLOATING POINT FORMAT FOR ∇FADD.*
- *USES ∇CAN.*

FUNCTIONS:

 ∇ *Z←FLOAT X*
[1] *↖CONVERT A NUMBER TO MULTIPRECISION FLOATING POINT*
[2] *→(0<ρ Z←X)/0*
[3] *X←⌊100000000⊙(X=0)+|X*
[4] *Z←(X-2),1↖CAN Z×100000000*2-X*
 ∇

 ∇ *R←SCALE MP*
[1] *↖ A SCALAR APPROXIMATION OF MP*
[2] *R←(100000000*1↖MP)×10000000011↖MP*
 ∇

EXAMPLES:

FLOAT 876
-2 876 0 0
 FLOAT -1.2345
-2 -1 -2345000 0
 FLOAT 1E20
0 1000000 0 0
 FLOAT -2 3 1415926 5358979
-2 3 1415926 5358979

 SCALE 6 ALPREC 1234.1234567890
1234.123457

FMUL

MULTIPRECISION FLOATING POINT MULTIPLICATION

SYNTAX:

$C \leftarrow A \text{ FMUL } B$

- A AND B ARE SCALAR OR MULTIPRECISION NUMBERS.
- RETURNS A MULTIPRECISION PRODUCT.
- SEE DESCRIPTION UNDER ∇ FADD.
- THE PRECISION OF C IS SUCH THAT THE RELATIVE ACCURACY OF ITS LEAST SIGNIFICANT DIGIT IS JUST BETTER THAN THE LEAST SIGNIFICANT OF THE TWO OPERANDS.
- USES: ∇ FLOAT ∇ MUL

FUNCTION:

$\nabla \text{ } C \leftarrow A \text{ FMUL } B$
[1] ∇ MULTIPRECISION FLOATING POINT MULTIPLY
[2] $C \leftarrow (0, 1 \nabla A \leftarrow \text{FLOAT } A) \text{ MUL } 0, 1 \nabla B \leftarrow \text{FLOAT } B$
[3] $C \leftarrow ((1 \nabla A) + (1 \nabla B) - C), (C \leftarrow (2 - \rho C) \lceil 2 - (\rho A) \lceil \rho B) \nabla 1 \nabla C$
 ∇

EXAMPLES:

A
0 47152 2357206
HALF
-6 5000000 0 0 0 0 0
FORMAT HALF
0.5000000 0000000 0000000 0000000 0000000 0000000
A FMUL HALF
-1 23576 1178603 0
FORMAT A FMUL HALF
23576 1178603.0000000
C
-2 12345 6789012 3456789
C FMUL HALF
-3 6172 8394506 1728394 5000000
FORMAT C FMUL HALF
6172.8394506 1728394 5000000
TWO
-5 2 0 0 0 0 0
FORMAT TWO
2.0000000 0000000 0000000 0000000 0000000
TWO FMUL HALF
-6 1 0 0 0 0 0
FORMAT TWO FMUL HALF
1.0000000 0000000 0000000 0000000 0000000 0000000

FORMAT CONVERT MULTIPRECISION NUMBER TO CHARACTER STRING

SYNTAX: Z←FORMAT X

- RETURNS A CHARACTER STRING WITH SEVEN-DIGIT GROUPS OF DECIMAL DIGITS PUNCTUATED BY SINGLE BLANKS OR DECIMAL POINT.

FUNCTION:

```

      ▽ Z←FORMAT A;B;E
[1]   A←CONVERT A MULTIPRECISION NUMBER TO A CHARACTER STRING
[2]   A←GET THE EXPONENT, AND INSERT LEADING AND TRAILING ZEROES
[3]   A←((0[2-E+ρA]ρ0),1+A,(0[E+1+A]ρ0
[4]   Z←( 8 0 ▽(|A|+10000000-(ρA)↑10000000),' '
[5]   Z[B+(8×1+ρA)-7×[IO]←' '
[6]   A←INSERT A DECIMAL POINT ON TOP OF A BLANK, E GROUPS FROM THE RIGHT
[7]   Z[1+(1[E-1]↑B)←'.'
[8]   A←DROP LEADING BLANKS, INSERT - JUST IN FRONT
[9]   Z←((0>1+A)/'-' ),(-[IO]-(Z=' ')\0)↑Z
      ▽
```

EXAMPLES:

```

      B
0 222 3333333 4444444 5555555
      FORMAT B
222 3333333 4444444 5555555.
      C
-2 -12345 -6789012 -3456789
      FORMAT C
-12345.6789012 3456789
      D
-3 76543 2109876 5432109
      FORMAT D
0.0076543 2109876 5432109
      FORMAT -5 74
0.0000000 0000000 0000000 0000000 0000074
```

FSQRT

MULTIPRECISION FLOATING POINT SQUARE ROOT

SYNTAX:

C←FSQRT A

- RETURNS THE MULTIPRECISION FLOATING POINT FORM OF THE SQUARE ROOT OF A.
- A MAY BE SCALAR, MP INTEGER, OR MP FLOATING POINT.
- THE PRECISION OF THE RESULT IS THAT OF THE OPERAND. (SEE THE DESCRIPTION FOR VFADD.)
- USES: VFLOAT VSQRT

FUNCTION:

```

      ▽ C←FSQRT A;E;M
[1]  ▽MULTIPRECISION FLOATING POINT SQUARE ROOT
[2]  M←[0.5×1+(C+2|E+1↑A)-ρA←FLOAT A
[3]  C←(M+|0.5×E),1↑SQRT 0,1↑((ρA)+C-2×M)↑A
      ▽
```

EXAMPLES:

```

      B
0 222 3333333 4444444 5555555
      FORMAT B
222 3333333 4444444 5555555.
      □←Z←FSQRT B
-2 47152 2357205 3020324 9027073
      FORMAT Z
47152 2357205.3020324 9027073
      FORMAT Z FMUL Z
222 3333333 4444444 5555554.9929000
      □←TWO←3 ALPREC 2
-5 2 0 0 0 0 0
      FORMAT TWO
2.0000000 0000000 0000000 0000000 0000000
      □←Z←FSQRT TWO
-5 1 4142135 6237309 5048801 6887242 969807
      FORMAT Z FMUL Z
1.9999999 9999999 9999999 9999999 9999997
```

FSUB

MULTIPRECISION FLOATING POINT SUBTRACTION

SYNTAX: *C←A FSUB B*

- *SUBTRACT B FROM A, USING MULTIPRECISION FLOATING POINT ARITHMETIC. A AND B MAY BE IN ANY NUMERIC FORMAT. (SEE DESCRIPTION UNDER ∇FADD)*
- *USES: ∇FLOAT ∇FADD.*

FUNCTION:

∇ *C←A FSUB B*
[1] *∇MULTIPRECISION FLOATING POINT SUB*
[2] *C←(A←FLOAT A)FADD(1↑B),-1↑B←FLOAT B*
 ∇

EXAMPLES:

A
0 47152 2357206
 B
0 222 3333333 4444444 5555555
 A FSUB B
0 -222 -3333333 -4397292 -3198349
 C
-2 12345 6789012 3456789
 A FSUB C
0 47152 2344861
 □←D←-3 76543 2109876 5432109
-3 76543 2109876 5432109
 C FSUB D
-2 12345 6712469 1346913
 FORMAT C
12345.6789012 3456789
 FORMAT D
0.0076543 2109876 5432109
 FORMAT C FSUB D
12345.6712469 1346913

MUL

MULTIPRECISION INTEGER MULTIPLICATION

SYNTAX: $C \leftarrow A \text{ MUL } B$

- A IS MULTIPLIED BY B, GIVING C, USING MULTIPRECISION INTEGER ARITHMETIC; A AND B MAY BE IN ANY NUMERIC FORMAT
- SEE THE DESCRIPTION UNDER ∇ ADD
- USES: ∇ FIX ∇ CAN

FUNCTION:

```
 $\nabla$  C←A MUL B
[1]   $\nabla$ MULTIPRECISION INTEGER MULTIPLY
[2]   $\nabla$ MAKE A THE SHORTER OF THE ARGUMENTS TO SAVE SPACE
[3]   $\rightarrow((\rho A)=(\rho A \leftarrow \text{FIX } A) \downarrow \rho B \leftarrow \text{FIX } B))/L1$ 
[4]   $C \leftarrow A$ 
[5]   $A \leftarrow B$ 
[6]   $B \leftarrow C$ 
[7]   $\nabla$ CHECK FOR POSSIBLE OVERFLOW ( $720 = (2*56) \div 1E7*2$ ).
[8]   $L1: \rightarrow (720 > 1 \uparrow \rho C \leftarrow A \circ . \times B) / L2$ 
[9]   $C \leftarrow ((\lfloor C \div 10000000 \rfloor), 0) + 0, 10000000 \mid C$ 
[10]  $L2: C \leftarrow \text{CAN} + \neg(\square IO - \downarrow \rho A) \Phi C, ((\rho A), \neg 1 + \rho A) \rho 0$ 
 $\nabla$ 
```

EXAMPLES:

```
      A
0 -1 -2345678 -9012345 -6789012
      B
0 222 3333333 4444444
      A MUL B
0 -274 -4855942 -5116597 -4938071 -3415514 -3649328
      123456 MUL 876543
0 10821 4492608
      5 MUL B
0 1111 6666667 2222220
```

SQRT

MULTIPRECISION INTEGER SQUARE ROOT

SYNTAX: *Z←SQRT A*

- *A* MAY BE SCALAR OR MULTIPRECISION BUT WILL BE TRUNCATED TO AN INTEGER.
- RUNNING TIME IS PROPORTIONAL TO $(\rho A) \times \rho A$, AND IS MUCH FASTER FOR NUMBERS WITH MANY TRAILING ZEROES.
- USES: ∇ FIX ∇ SUB ∇ MUL ∇ DIV ∇ CAN.

FUNCTION:

```

      ▽ Z←SQRT A;F;N;REM
[1]  AMULTIPRECISION INTEGER SQUARE ROOT BY ELEMENTARY RECURSION
[2]  AGET AN ACCURATE STARTING VALUE
[3]  Z←(N←1+⌊0.5×ρA)↑CAN(10000000⌊(6+2⌊ρA)↑A←FIX A)×0.5
[4]  ADROP TRAILING ZEROES FROM A AND DOUBLE IT
[5]  →0×10=1↑1+A←CAN 2×((ρA)↑IO-(ϕ0≠A)1)↑A
[6]  LOOP:Z←N↑Z MUL F←((1+ρF)↑ 0 1 5000000 )SUB F←((N+ρA)↑Z MUL Z)DIV A
[7]  →(1<+/N↑F)/LOOP
      ▽
```

EXAMPLES:

```

      B
0 222 3333333 4444444 5555555
      □←A←SQRT B
0 47152 2357205
      A MUL A
0 222 3333333 4415961 5412025
      NOTICE THAT THIS ESTIMATE OF B*.5 IS SLIGHTLY LOW
      □←A←A ADD 1
0 47152 2357206
      A MUL A
0 222 3333333 4510266 126436
      BUT THE NEXT HIGHER INTEGER IS SLIGHTLY TOO HIGH,
      SO THAT THE TRUE SQUARE ROOT IS BRACKETED BETWEEN
      THESE TWO VALUES.
```


SUB MULTIPRECISION INTEGER SUBTRACTION

SYNTAX: *C←A SUB B*

- *SUBTRACT B FROM A, GIVING C, USING MULTIPRECISION INTEGER ARITHMETIC. A AND B MAY BE IN ANY NUMERIC FORMAT, C WILL BE MP INTEGER. SEE ∇ADD.*
- *USES: ∇ADD ∇FIX*

FUNCTION:

```
      ∇ C←A SUB B
[1]  ∇MULTIPRECISION INTEGER SUBTRACT
[2]  C←(A←FIX A)ADD-B←FIX B
      ∇
```

EXAMPLES:

```
      A
0 1 2345678 9012345 6789012
      B
0 222 3333333 4444444
      A SUB B
0 1 2345456 5679012 2344568
      A SUB 4
0 1 2345678 9012345 6789008
      B SUB B
0 0
```

Section V

Mathematical / Numerical Functions

AMORTIZE

MORTGAGE CALCULATION BY MONTHS

SYNTAX:

M←AMORTIZE DEBT,RATE,MONTHS

- DISPLAYS MORTGAGE TABLE INDICATING THE PROGRESSIVE DEBT REDUCTION, AND THE LEVEL PAYMENT SCHEDULE, AS IT BREAKS DOWN BETWEEN AMORTIZED DEBT AND INTEREST.
- DEBT IS TOTAL UNITS (E.G., DOLLARS) BORROWED.
- RATE IS ANNUAL INTEREST, AS PERCENTAGE (E.G., .095).
- MONTHS (E.G., 120 IF TEN YEAR MORTGAGE).
- RETURNS A MATRIX THAT RETAINS FULL PRECISION FOR SUMMARY CALCULATIONS. (SEE EXAMPLE)
- USES: ∇OUTPUT

FUNCTION:

```

∇ M←AMORTIZE NV3;□IO
[1]  RATE←NV3[2]÷12×□IO←1
[2]  MONTHS←NV3[3]
[3]  DEBT←NV3[1]
[4]  M←(MONTHS,5)ρI←0
[5]  PAYMENT←DEBT×RATE÷1-÷(1+RATE)*MONTHS
[6]  BACK:NEWDEBT←DEBT-AMORTIZED←PAYMENT-INTEREST←DEBT×RATE
[7]  M[I;]←(I+I+1),DEBT,PAYMENT,AMORTIZED,INTEREST
[8]  →(0<DEBT+NEWDEBT)/BACK
[9]  'M DEBT PAYMT AMORT INT'OUTPUT(9 0 0 4+M),' ',' ','9 2 0 1+M
∇

```

EXAMPLES:

```

M←AMORTIZE 1000 .06 12

```

M	DEBT	PAYMT	AMORT	INT
1	1000.00	86.07	81.07	5.00
2	918.93	86.07	81.47	4.59
3	837.46	86.07	81.88	4.19
4	755.58	86.07	82.29	3.78
5	673.29	86.07	82.70	3.37
6	590.59	86.07	83.11	2.95
7	507.48	86.07	83.53	2.54
8	423.95	86.07	83.95	2.12
9	340.01	86.07	84.37	1.70
10	255.64	86.07	84.79	1.28
11	170.85	86.07	85.21	0.85
12	85.64	86.07	85.64	0.43

```

+ / 0 2+M
1032.797156 1000 32.79715648
A          ↑-----TOTAL INTEREST PAID
A          ↑-----TOTAL DEBT REPAID
A  ↑-----TOTAL REPAID

```

CONV CONVERT DECIMAL VALUES TO ANY BASE [DIGITS CONFRAC]

SYNTAX: R←BASE CONV DEC

- VALUES ARE CONVERTED TO CHARACTER STRINGS THAT RETAIN THEIR ARITHMETIC CAPABILITY.
- THE CHARACTER STRINGS CAN BE RECONVERTED BY VDEC.
- THE GLOBAL VARIABLE, DIGITS WILL SUPPORT UP TO BASE 36. THE CATENATION OF UNDERSCORED LETTERS AND OTHER CHARACTERS TO DIGITS WILL PERMIT LARGER BASES.
- NEGATIVE NUMBERS WILL BE TREATED CORRECTLY.
- FRACTIONS WILL BE CLOSELY APPROXIMATED.
- INTEGER CONVERSIONS, E.G., HEXADECIMAL, WILL BE EXACT.
- USES: VENC VDL VCONFRAC

FUNCTIONS:

```

V R←BASE CONV DEC; IO
[1] IO←0
[2] DIGITS←'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
[3] R←' 'DL' '[DEC<0], '0'DL DIGITS[BASE ENC\|DEC]
[4] R←R,BASE CONFRAC DEC
V
V R←B CONFRAC N; IO; NN; BB; CT
[1] CT←1E-15
[2] →0 IF 0=NN+1 || N, R←1 IO←J←0
[3] BACK: R←R, [NN÷BB←B*-J←J+1
[4] →BACK IF 1≠1+NN←BB|NN
[5] R←'.', DIGITS[R]
[6] A CONVERTS DECIMAL FRACTIONS
V

```

EXAMPLES:

```

10DEC 10CONV 10DEC'1234.5678'
1234.5678
16DEC'20000'
131072
36CONV 123456789
21I3V9
36DEC 36CONV 1234567890123456
1234567890123456

```

```

[1] CT←1E-15
[2] →0 IF 0=NN+1 || N, R←1 IO←J←0
[3] BACK: R←R, [NN÷BB←B*-J←J+1
[4] →BACK IF 1≠1+NN←BB|NN
[5] R←'.', DIGITS[R]
[6] A CONVERTS DECIMAL FRACTIONS
V

```

ANALYSIS: 16 CONV 131072

```

[3] R←' 'DL' '[DEC<0], '0'DL DIGITS[BASE ENC\|DEC]
ABSOLUTE VALUE

```

```

131072
[3] R←' 'DL' '[DEC<0], '0'DL DIGITS[BASE ENC\|DEC]
(SEE VENC)

```

```

0 2 0 0 0 0
[3] R←' 'DL' '[DEC<0], '0'DL DIGITS[BASE ENC\|DEC]
SELECTED

```

```

020000
[3] R←' 'DL' '[DEC<0], '0'DL DIGITS[BASE ENC\|DEC]
LEADING ZERO DELETE

```

```

20000
[3] R←' 'DL' '[DEC<0], '0'DL DIGITS[BASE ENC\|DEC]
IF NEGATIVE

```

20000

DATE COMPUTE NORMAL DATE FROM ASTRONOMERS' DAY NUMBER

SYNTAX: Z←DATE JS

- RETURNS MONTH, DAY, YEAR, STYLE. (SEE VDAYNO)
- JS IS THE JULIAN DAY NUMBER AS WOULD BE FOUND BY VDAYNO.
- JS MAY BE A SINGLE VALUE OR A VECTOR. OPTIONALLY, IT MAY BE AN ARRAY OF SHAPE (N,2) WHERE THE SECOND COLUMN IS 0 OR 1, STATING FOR EACH JULIAN DAY WHETHER THE OLD (0) OR NEW (1) CALENDAR WAS IN USE. NORMALLY, THIS STYLE IS COMPUTED AUTOMATICALLY.

FUNCTION:

```

V Z←DATE JS;C;D;J;M;S;Y
[1] ACONVERT JULIAN DAY NUMBER (AND OPTIONAL STYLE) TO DAY,MO,YEAR,STYLE
[2] AJS MAY ALSO BE A VECTOR OF JD'S OR AN ARRAY OF JD'S AND STYLES.
[3] JS←(2↑(ρJS),1 1)ρJS
[4] S←(J>2423434)∨(J>2299171)^(JS,2361221<J+JS[;⌈IO⌋])[;⌈IO+1⌋]
[5] C←[(J+J-1684595)÷36524.25
[6] J←J+((~S)×(2-C)+[C÷4])-[36524.25×C
[7] Y←[(J+1)÷365.25025
[8] J←J+31-[365.25×Y
[9] D←J-[30.5875×M+[J÷30.5875
[10] M←M+2-12×J+[M÷11
[11] Z←M,D,(J+Y+100×C-1),[⌈IO+0.5⌋]S
V

```

EXAMPLES:

```

⌈←Z←DAYNO 5 17 1977
2443281
DATE Z
5 17 1977 1
DATE Z,Z+30
5 17 1977 1
6 16 1977 1
A IF THE OLD STYLE CALENDAR WAS IN USE AFTER 1752, OR
A THE NEW STYLE IN USE BEFORE THEN, THE USER MUST
A GIVE THE STYLE. FOR EXAMPLE, IN THE USA:
⌈←Z←DAYNO 1 1 1800
2378497
DATE Z
1 1 1800 1
A BUT IN RUSSIA
DATE 1 2ρZ,0
12 21 1799 0

```

DAYNO DAY NUMBER FOR ASTRONOMERS [MOONPHASE]

SYNTAX: Z←DAYNO DATE

- DAYS SINCE 1/1/4713 B.C. (SEE VDATE)
- DATE IS MONTH, DAY, YEAR. IT MAY BE A SINGLE SUCH TRIPLET OR A MATRIX, EACH ROW OF WHICH IS A TRIPLET.
- OPTIONALLY, THE INPUT MAY HAVE A FOURTH COMPONENT OR COLUMN OF 0 OR 1 FOR EACH DATE, STATING WHETHER THE OLD (0) OR NEW (1) STYLE CALENDAR WAS IN USE. NORMALLY, THIS IS COMPUTED AUTOMATICALLY.
- DAY OF THE WEEK MAY BE COMPUTED BY $1+7|1+\text{DAYNO DATE}$.
SUNDAY = 1, ..., SATURDAY = 7. (SEE VDAY)

FUNCTIONS:

V Z←DAYNO DATE;C;D;JF;M;S;Y

[1] A COMPUTE JULIAN DAY NUMBER. DATE IS A VECTOR OR ARRAY WHOSE ROWS ARE
[2] A MONTH, DAY, YEAR, STYLE. STYLE IS AN OPTIONAL LOGICAL VALUE = 1 IF THE
[3] A NEW STYLE (GREGORIAN) CALENDAR SHOULD BE USED. THE JULIAN DAY IS
[4] A A CONTINUOUS COUNT THAT BEGAN AT 0 AT NOON, 1/1/4712 (I.E. 4713 BC).
[5] DATE←(2+1, pDATE) pDATE
[6] Z←1001(Y+DATE[;2+IO]), [IO](M+DATE[;IO]), [IO-0.5] D+DATE[;1+IO]
[7] S←(Z>19230114) v (Z>15821025) ^ (DATE, Z>17520902)[;3+IO]
[8] C←(2~S)+0.75×S×[0.01×Y-JF+2≥M
[9] Z←31+D+(1367×JF+(M-2)÷12)-[C-1365.25×4712+Y-JF

V

V R←MOONPHASE MDYS

[1] A 0.00 IS NEW MOON; 0.75 IS LAST QUARTER
[2] R←21|÷29.53059÷9+DAYNO MDYS

V

EXAMPLES:

DAYNO 5 17 1977

2443281

□←Z←4 3p2 28 1900 3 1 1900 2 28 2000 3 1 2000

2 28 1900

3 1 1900

2 28 2000

3 1 2000

DAYNO Z

2415079 2415080 2451603 2451605

A AS CAN BE SEEN, 1900 WAS NOT A LEAP YEAR, BUT 2000 WILL BE ONE.

1+7|1+DAYNO 5 17 1977

3

A I.E. TUESDAY

MOONPHASE 5 17 1977

0.99

A I.E. JUST BEFORE NEW MOON

(DAYNO 5 17 1977)-DAYNO 1 1 1901

27895

A THE AGE IN DAYS OF THE TWENTIETH CENTURY.

A IF THE OLD STYLE CALENDAR WAS IN USE AFTER 1752, OR THE

A NEW STYLE IN USE BEFORE THEN, THE STYLE MUST BE ENTERED.

A FOR EXAMPLE, IN RUSSIA BEFORE THE REVOLUTION (COMPARE ABOVE)

DAYNO 2 28 1900 0

2415091

DAYS

DATE CALCULATIONS [DATES NDATES PAYDAY]

SYNTAX:

N←DAYS D D←DATES N R←NDATES KM

- GIVEN A NUMERIC VECTOR OR MATRIX OF THE FORM: MONTH, DAY, YEAR (1 30 1977), DAYS WILL RETURN FOR EACH, THE NUMBER OF DAYS SINCE 1/1/1, INCLUSIVE, AS IF THE GREGORIAN CALENDAR HAD BEEN IN USE CONTINUALLY, WITH NO LOSS AT THE CHANGE (IN ENGLAND ON SEPTEMBER 14, 1752.) FOR A BETTER FUNCTION, SEE VDAYNO.
- THE 7 OF DAYS, CAN SELECT THE DAYS OF THE WEEK, WITH 0←SUNDAY; 1←MONDAY, ETC.
- DATES CONVERTS THE DAYS BACK INTO CALENDAR DATES.
- NDATES CONVERTS DATES AVAILABLE AS CHARACTER MATRICES OF THE FORM: '013077' TO THAT REQUIRED BY DAYS.

FUNCTIONS:

```

V N←DAYS D;P;I;IO
[1] D←((×/I←1+ρD),3)ρD ΔIO+1
[2] P←=f0=(N←(0,[0.1]4 100 400)τD[;3])[2;;]
[3] N←(365×D[;3]-1)+-fN[1;;],[1]P
[4] N←IpN+D[;2]+(L30.56×D[;1])-30+(D[;1]≥3)×2-P
[5] A NOT ACCURATE PRIOR TO 1753. USE VDAYNO.

V
V D←DATES N;Y;M;P;R;IO
[1] M←(0,[0.1]4 100 400)τY←(L(364+D←,N)÷365.2425)°.+0,IO+1
[2] D←D-,0 -1+(R+D>M[;2])φM←(365×Y-1)+-fM[1;;],[1]P←=f0=M[2;;]
[3] D←D-L30.56×M←(D←30+D+(D>59+P)×2-P+(RφP)[;1])+30.56
[4] D←((ρN),3)ρM,D,0 -1+RφY
[5] A NOT ACCURATE PRIOR TO 1753. SEE VDAYNO

V
V R←NDATES KM
[1] R←((1+ρKM),3)ρ2,1 1 0 1 1 0 1 1 0\KM
[2] R[;IO+2]←R[;IO+2]+1900
[3] A ASSUMES 20TH CENTURY

```

EXAMPLES:

```

V R←PAYDAY MDY;IO
[1] A FRIDAY, ON OR BEFORE MDY
[2] R←(DAYS MDY)-17+IO+0
[3] R←DATES R[(7|R)15]

```

```

V
PAYDAY 6 30 1977
6 24 1977
AA
081118
021926
031354
062758

```

DATES	DAYS	NDATES	AA
8	11	1918	
2	19	1926	
3	13	1954	
6	27	1958	

DEC CONVERT TO DECIMAL

SYNTAX: R←BASE DEC Q

- CHARACTER VECTORS BELONGING TO THE GLOBAL DIGITS, ' . ' REPRESENTING SCALAR NUMBERS IN ANY BASE, WILL SEEM TO BE CONVERTED TO THEIR DECIMAL VALUES, WITH WHICH ORDINARY CALCULATIONS CAN BE MADE.
- . WILL BE UNDERSTOOD AS SEPARATING THE INTEGER PORTION FROM ANY POSSIBLE FRACTION. FRACTIONS WILL BE CLOSELY APPROXIMATED. INTEGERS WILL BE EXACT, UNLESS THEY ARE FORCED TO FLOAT.
- ARITHMETIC RESULTS CAN BE CONVERTED TO OTHER BASES THROUGH THE USE OF ∇CONV.
- DIGITS WILL SUPPORT BASES $2 \leq \text{BASE} \leq 36$.
- USES: ∇ESCAPE

FUNCTION:

```
∇ R←BASE DEC Q;□IO;P;S
[1] Q←(S+Q≠'.' )/Q+(P+Q≠'.' )/Q
[2] 'CHARACTER ERROR'ESCAPE~^/Q∈BASE↑R+DIGITS
[3] R←(1-2×0∈S)×(BASE*-0[-1++/∇\~P)×BASE↓R↓Q,1□IO+0
```

EXAMPLES:

```
(10 DEC'1234')=2101234
1
16 DEC'20000'
131072
(16 DEC'20000')+16DEC'FFFF'
196607
16 CONV (16DEC'FFFF')-16DEC'1234'
```

EDCB

```
10 CONV -12345.6789
-12345.678899999999988
10 CONV 2*-16
.00001525878906
16 DEC 16CONV 2*-16
1.525878906E-5
16 CONV 2*-16
.0001
16 CONV 2*-64
.0
16 CONV 2*-32
.00000001
```

ANALYSIS: 16 DEC'⁻EDCB.125'

```
[3] R←(1-2×0∈S)×(BASE*-0[-1++/∇\~P)×BASE↓R↓Q,1□IO+0
-----DIGITS
-----EVALUATED
-----POINT RETURNED
-----SIGN RETURNED
-60875.07153
```

SYNTAX: $R \leftarrow U \text{ DROUNDS } V$

- ROUNDING THE ELEMENTS OF A VECTOR BEFORE SUMMATION MAY CAUSE AN ERROR IN THE SUM. ROUND-OFF ERRORS DO NOT NECESSARILY COMPENSATE. IT WOULD BE GOOD PRACTICE TO CARRY MAXIMUM PRECISION UNTIL THE FINAL SUMMATION, THEN ROUNDING THE SUM.
- WHEN THIS IS NOT POSSIBLE, WE WOULD STILL WANT THE ROUNDED SUM TO EQUAL THE SUM OF THE ROUNDED ELEMENTS. SEE ∇ ROUNDS

FUNCTION:

```

      ▽ R←U DROUNDS V;□CT;□IO;E;N
[1]   E←1|V←V÷U+□CT←□IO←0
[2]   N←(⌊0.5++/V)÷+/⌊V
[3]   R←U×(⌊V)+N>ΔΨE

```

EXAMPLE:

```

      A
0.86 0.21 0.95 0.9 0.7 0.69 0.44 0.59 0.16 0.57 0.06 0.47 0.6 0.46 0.93 0.61
      +/A
9.2
      +/.1 ROUNDS A
9.4
      +/.1 DROUNDS A
9.2

```

ANALYSIS: .1 DROUNDS A

```

[1]   E←1|V←V÷U+□CT←□IO←0
      ----- ERRORS IF FLOOR USED. NOTE EFFECT OF □CT←0
0.6 0.1 0.5 1 1 0.9 0.4 0.9 0.6 0.7 0.6 0.7 1 0.6 0.3 0.1
[2]   N←(⌊0.5++/V)÷+/⌊V
      ----- FLOORS
8 2 9 8 6 6 4 5 1 5 0 4 5 4 9 6
[2]   N←(⌊0.5++/V)÷+/⌊V
      -----
82
[2]   N←(⌊0.5++/V)÷+/⌊V
      ----- ROUNDED UP FOR TESTING
92
[2]   N←(⌊0.5++/V)÷+/⌊V
      -----
10
      ADJUSTMENTS NEEDED, BUT WHERE?
[3]   R←U×(⌊V)+N>ΔΨE
      ----- LOCATION, BY SEVERITY
10 14 11 2 0 3 12 4 8 5 7 6 1 9 13 15
[3]   R←U×(⌊V)+N>ΔΨE
      ----- HERE! (10 WORST REPRESENTED BY FLOOR)
0 0 0 1 1 1 0 1 1 1 1 1 1 0 0
[3]   R←U×(⌊V)+N>ΔΨE
      -----
8.6 2.1 9.5 9 7 6.9 4.4 5.9 1.6 5.7 0.6 4.7 6 4.6 9.3 6.1
[3]   R←U×(⌊V)+N>ΔΨE
      -----
8 2 9 8 6 6 4 5 1 5 0 4 5 4 9 6
[3]   R←U×(⌊V)+N>ΔΨE
      -----
8 2 9 9 7 7 4 6 2 6 1 5 6 5 9 6

```

ENC GENERATE SUFFICIENT ENCODING POSITIONS

SYNTAX: $R \leftarrow S \text{ ENC } A$

ENCODE(τ) AND DECODE(\perp) WOULD BE FULLY COMPLEMENTARY IF SUFFICIENT RADIX POSITIONS COULD BE SUPPLIED. ∇ENC WILL PERFORM THE FULL REPRESENTATION OF ITS ARGUMENT, ACCORDING TO THE RADIX, S . $(S > 1) \wedge (1 \leq \perp / A)$

FUNCTION:

$\nabla R \leftarrow S \text{ ENC } A$
 $[1] \quad A \quad 1 < S \quad 1 \leq \perp / A$
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$
 ∇

EXAMPLES:

12 ENC 143 144 145
 0 0 0
 0 1 1
 11 0 0
 11 0 1

12 112 ENC 143 144 145
 143 144 145

12 112 12 143 144 145
 143 0 1 (TWO TWELVES NOT ENOUGH)

ANALYSIS:

12 ENC 143 144 145
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$

143 144 145
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$

12
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$
 GUARDING AGAINST ZERO

145
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$

2.002784991
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$

3
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$
 GUARD HIGH-ORDER POSITION

4
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$

12 12 12 12
 $[2] \quad R \leftarrow ((1 + \lceil S \otimes 1 \lceil \lceil / , A \rangle \rho S) \tau A$

0 0 0
 0 1 1
 11 0 0
 11 0 1

FREQ FREQUENCY DISTRIBUTION OF ELEMENTS [SORTDA KFORM]

SYNTAX: R←FREQ A

- THE ARGUMENTS MUST BE NUMERIC CODES, OR NUMERIC REPRESENTATIONS OF CHARACTER GROUPS.
- TWO NUMERIC COLUMNS WILL BE RETURNED:
THE FIRST, THE COUNT, OR FREQUENCY
THE SECOND, THE CORRESPONDING CATEGORY.
- FREQUENCY WILL APPEAR IN DESCENDING ORDER,
WITHIN WHICH THE CATEGORIES WILL ASCEND.
- USES: VDREP

FUNCTIONS:

∇ R←FREQ A
 [1] A A IS A NUMERIC STRUCTURE. USE LJNFORM OR NFORM TO CONVERT.
 [3] A IF THE ARGUMENT IS A CONVERTED CHARACTER MATRIX,
 [4] A THE SECOND COLUMN OF THE RESULT CAN BE RECONVERTED BY KFORM,
 [5] R←(12)SORTDAΦR,[∅IO+0.5]+/A∅.=R←DREP A←,A
 ∇

EXAMPLES:

∇ K←C KFORM N;∅IO
 [1] K←DLBQC[(11ρρC)τN]Δ ∅IO←0
 ∇

FREQ ?5ρ5

2 2
 2 4 (2 TWO'S, 2 FOURS, 1 THREE, NO ONES, NO FIVES)
 1 3

N←(AV, ''')NFORM VERT'THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG''S BACK'
 (∇0 -1+R),MATRIX (AV, ''') KFORM 0 1+R←FREQ N
 9 (NINE BLANKS)

4 E
 4 O
 2 A
 2 B
 2 C
 2 D
 2 H
 2 K
 2 R
 2 T
 2 U
 1 F
 1 G
 1 I
 1 J
 1 L
 1 M
 1 N
 1 P
 1 Q
 1 S
 1 V
 1 W
 1 X
 1 Y
 1 Z
 1 '

∇ R←DA SORTDA M;N
 [1] A DA IS A PAIR OF COLUMN NUMBERS IN USER'S ORIGIN.
 [2] A A CONTROLS THE INITIAL ASCENDING SORT.
 [3] A D CONTROLS THE FINAL DESCENDING SORT.
 [4] A M IS A NUMERIC MATRIX WHICH MAY RESULT FROM NFORM.
 [5] R←N[ψ,(N←M[Δ,M[;-1+DA];)][;1+DA];]
 ∇

PI COMPUTE PI TO ARBITRARY PRECISION

SYNTAX: SUM←PI N

- COMPUTE PI (3.14159+) TO 7×N DECIMAL DIGITS OF PRECISION
- THE ARCSIN POWER SERIES (6×10^{-5}) IS SUMMED--NOT THE FASTEST KNOWN METHOD, BUT FAR FROM THE SLOWEST
- RUNNING TIME IS PROPORTIONAL TO N^2
- USES: VADD VMUL VDIV

FUNCTION:

```

      V SUM←PI N;I;TERM;REM
[1]  A COMPUTE PI TO 7×N DECIMAL PLACES BY THE POWER SERIES FOR  $6 \times 10^{-5}$ 
[2]  SUM←TERM+0,(N+I+1)↑3
[3]  LOOP:TERM←(TERM MUL I)DIV 4×I+1
[4]  SUM←SUM ADD TERM DIV I←I+2
[5]  →(V/TERM≠0)/LOOP
[6]  SUM←(-N),1↓SUM
      V
```

EXAMPLES:

```

      □←P←PI 6
- 6 3 1415926 5358979 3238462 6433832 7950288 4197136
      A COMPUTE THE RAMANUJAN NUMBER *OK*.5 FOR K=163
      □←K+3 ALPREC 163
- 5 163 0 0 0 0 0
      FORMAT Z←FSQRT K
12.7671453 3480370 4661710 9520097 8089234
      FORMAT Z←P FMUL Z
40.1091699 9113251 9755350 0836229 0414003
      FORMAT Z←FEXP Z
2625 3741264 0768743.9999999 9999925 0066319 1466030 7724958
      A FOR NUMEROUS OTHER VALUES OF K, THESE NUMBERS ARE VERY
      A CLOSE TO PERFECT INTEGERS. ALL THE MORE REMARKABLE THAT
      A RAMANUJAN DISCOVERED THEM IN 1915 WITHOUT THE AID OF A
      A COMPUTER.
```

QPROBF COMPUTE CHI SQUARE PROBABILITY FUNCTION

SYNTAX: Z←CHISQ QPROBF NU

- COMPUTE THE PROBABILITY OF A GIVEN CHI SQUARE VALUE OCCURRING FOR A GIVEN NU (NUMBER OF DEGREES OF FREEDOM)
- NU IS ROUNDED DOWN TO THE NEXT LOWER EVEN INTEGER
- NOTE THE EXTREME ELEGANCE WITH WHICH IT IS POSSIBLE IN APL TO EXPRESS A POWER SERIES

FUNCTION:

```

      ∇ Z←CHISQ QPROBF NU
[1]  COMPUTE Q(CHISQ|NU), WHERE NU IS ROUNDED DOWN TO AN EVEN INTEGER
[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2
      ∇

```

EXAMPLES:

```

      5.78 QPROBF 20
0.999164
      27.3 QPROBF 20
0.127033
      27.3 QPROBF 40
0.93691

```

ANALYSIS: 5.78 QPROBF 20

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

0 1 2 3 4 5 6 7 8 9

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

1 1 2 3 4 5 6 7 8 9

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

1 1 2 6 24 120 720 5040 40320 362880

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

(NOT ALL RECIPROCALs SHOWN)

```

1 1 0.5 0.1666666667 0.04166666667 0.008333333333 0.001388888889

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

2.89

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

17.9782608

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

2.89

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

-2.89

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

0.05557621261

```

```

[2]  Z←(*-CHISQ÷2)×(CHISQ÷2)⌈Φ÷×\1[-□IO-1]NU÷2

```

```

0.9991636444

```

ROMAN

CONVERT INTEGER TO ROMAN NUMERALS

SYNTAX:

R←ROMAN N

- ROMAN NUMERALS MAY BE REQUIRED FOR CERTAIN TYPES OF PAGE OR PARAGRAPH NUMBERING. THEY ALSO ILLUSTRATE THAT THERE IS A DISTINCTION BETWEEN THE VALUE OF A NUMBER AND ITS REPRESENTATION. N IS AN INTEGER GREATER THAN ZERO. R IS A CHARACTER VECTOR REPRESENTING N AS A ROMAN NUMERAL.

FUNCTION:

```

▽ R←ROMAN N;I;□IO
[1]  □IO←0
[2]  I←0 1000τ'ρN
[3]  R←0 5τ10 10 10 10τN+I[1]
[4]  N←,Q(14)°. <,Q(0[R-1 3°.×4=R[1;]], [0]R[0;])⊖4=R
[5]  R←(I[0]ρ'M'),N/,Q4 16ρ'×M××DCMDLXCLVIXV'
▽

```

EXAMPLES:

```

      ROMAN 7
VII
      ROMAN 77
LXXVII
      ROMAN 977
CMLXXVII
      ROMAN 1977
MCMLXXVII
      ROMAN 10000
MMMMMMMMMM

```

ROUNDS

SELECTIVE SYMMETRICAL ROUNDING

SYNTAX: $R \leftarrow U \text{ ROUNDS } A$

- TO ROUNDOFF NUMBERS TO ANY GIVEN UNITS
- TO ROUND NEGATIVE NUMBERS AWAY FROM ZERO
- RESULT WILL BE THE NEAREST MULTIPLE OF THE CORRESPONDING UNIT.

FUNCTION:

$\nabla R \leftarrow U \text{ ROUNDS } A$
 [1] A IS A SCALAR OR CONFORMABLE STRUCTURE OF SPECIFIED UNITS
 [2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 ∇

EXAMPLE:

10 0.01 ROUNDS 5287 1234.006
 5290 1234.01

ANALYSIS:

A
 3.6 145 -150 -151 1.027
 U
 1.5 3 7 7 0.03
 $U \text{ ROUNDS } A$

[2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 - THE CORRESPONDING UNITS

1.5 3 7 7 0.03
 [2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 - - - - - NORMALIZED

2.4 48.33333333 -21.42857143 -21.57142857 34.23333333
 [2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 - - - - - ABSOLUTE VALUES

2.4 48.33333333 21.42857143 21.57142857 34.23333333
 [2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 - - - - - HALF-ADJUSTMENT ADDED

2.9 48.83333333 21.92857143 22.07142857 34.73333333
 [2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 - - - - - FLOOR

2 48 21 22 34
 [2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 - - - - - NORMALIZATION REVERSED

3 144 147 154 1.02
 [2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 - - - - - THE ORIGINAL SIGNS

1 1 -1 -1 1
 [2] $R \leftarrow (\times A) \times U \times \lfloor 0.5 + |A \div U|$
 - - - - - NEGATIVE NUMBERS RESTORED

3 144 -147 -154 1.02

TO NUMERIC VECTORS IN EQUAL INCREMENTS [BY IN FROM]

SYNTAX: $R \leftarrow A \text{ TO } B$ $R \leftarrow N \text{ FROM } A$ $R \leftarrow A \text{ TO } B \text{ IN } M$
 $R \leftarrow A \text{ TO } B \text{ BY } C$ $R \leftarrow N \text{ FROM } A \text{ BY } C$

A: STARTING VALUE
 B: LAST VALUE (OR BOUNDARY VALUE)
 C: INCREMENT (POSITIVE OR NEGATIVE BUT NOT ZERO)
 M: NUMBER OF INTERVALS DESIRED ($M \neq 0$).
 NUMBER OF VALUES OBTAINED = $M+1$
 N: NUMBER OF VALUES DESIRED
 R: RESULTING NUMERIC VECTOR WITH EQUAL INCREMENTS

◦ WHEN THE FUNCTIONS 'TO' AND 'FROM' ARE USED ALONE, THE INCREMENT IS UNDERSTOOD TO BE ONE. SEQUENCES OF ANY OF THE ABOVE FORMS ARE ALSO POSSIBLE, PROVIDED THAT THEY ARE SEPARATED BY COMMAS AS SHOWN IN THE EXAMPLES.

EXAMPLES:

5 6 7 8 9 10 \leftrightarrow 5 TO 10	0 2 4 6 \leftrightarrow 0 TO 6 IN 3
4.1 5.1 6.1 \leftrightarrow 4.1 TO 7	
5 4 3 2 \leftrightarrow 5 TO 2	6 4 2 0 \leftrightarrow 6 TO 0 IN 3
5 7 9 \leftrightarrow 5 TO 10 BY 2	3 4 5 6 7 \leftrightarrow 5 FROM 3
5 \leftrightarrow 5 TO 10 BY 6	
5 5.5 6 6.5 7 \leftrightarrow 5 TO 7 BY .5	15 12 9 6 \leftrightarrow 4 FROM 15 BY -3
1 TO 5, 10 TO 20 BY 2, 5 FROM 50, 40 TO 30 BY 5	
1 2 3 4 5 10 12 14 16 18 20 50 51 52 53 54 40 35 30	

FUNCTIONS:

∇ Z←A TO B;D;R;X;□IO	∇ Z←B BY C
[1] □IO←0	[1] 'ZERO IS INVALID ARGUMENT'HANG 0=1↑C
[2] R←ρρZ←1,B	[2] Z←(1,ρZ)ρZ←B,C
[3] Z←,Z	∇
[4] X← Z[2×R>1]	∇ Z←B IN M
[5] D←Z[1]-A	[1] 'ZERO IS INVALID ARGUMENT'HANG 0=1↑M
[6] →(3>R)↑L1	[2] Z←(1 1,ρZ)ρZ←B,M
[7] B←A+(D÷X)×11+X	∇
[8] →L2	
[9] L1:B←A+(X×D)×11+L D÷X	∇ Z←N FROM A;R;□IO
[10] L2:Z←B,(2+R>1)÷Z	[1] □IO←0
∇	[2] R←ρρZ←1,A
	[3] Z←(Z[1]+Z[2×R>1]×1N),(2+R>1)÷Z←,Z
	∇

NOTE: THIS IS AN EXAMPLE OF LINKING APL FUNCTIONS TOGETHER. THE CORE FUNCTIONS, 'TO' AND 'FROM', DETERMINE WHETHER OR NOT THERE WAS A 'BY' OR 'IN' CLAUSE FROM THE RANK OF THEIR RIGHT ARGUMENTS.

TRUNC

TRUNCATE HIGHER AND LOWER ORDER DIGITS

SYNTAX:

R←U TRUNC A

- SELECT PARTICULAR DECIMAL DIGIT POSITIONS
- EXPLICIT (INPUT) DECIMAL FRACTIONS WILL BE RETURNED CORRECTLY. LOW-ORDER DIGITS OF COMPUTED FRACTIONS MAY NOT BE EXACT IN DECIMAL REPRESENTATION.

FUNCTION:

```

      ▽ R←U TRUNC A
[1]   R←10|L|A÷U
[2]   A IF U IS ANY POWER OF TEN, THEN THE CORRESPONDING DECIMAL
[3]   A POSITION OF A IS RETURNED.
[4]   A IF U IS A UNIT DIVISOR, A IS FIRST CONVERTED TO THE NEW UNIT,
[5]   A THEN THE NEW UNITS PLACE IS RETURNED.
      ▽
```

EXAMPLE:

```

      (10*15)TRUNC 12345+□IO←0
5 4 3 2 1
```

ANALYSIS:

```

      .1 1 10 TRUNC 100÷7
[1]   R←10|L|A÷U
0.1 1 10
[1]   R←10|L|A÷U
142.8571428571428 14.28571428571429 1.428571428571428
[1]   R←10|L|A÷U
142.8571428571428 14.28571428571429 1.428571428571428
[1]   R←10|L|A÷U
142 14 1
[1]   R←10|L|A÷U
2 4 1
```

ZDIV ZERO TOLERANT DIVISION [CDIV]

SYNTAX: R←N ZDIV D

- DOMAIN ERRORS ARE UNDESIRE IN COMMERCIAL MATRIX OPERATIONS WHERE ZEROS USUALLY INDICATE UNAVAILABLE INFORMATION.
- N AND D ARE CONFORMABLE NUMERIC STRUCTURES OR SCALARS IN ANY COMBINATION.
- ZERO WILL BE RETURNED INSTEAD OF THE DOMAIN ERROR.

FUNCTIONS:

∇ R←N ZDIV D
[1] A RETURNS ZERO WHEN DIVISOR IS ZERO
[2] A APL RETURNS UNITY WHEN N AND D ARE BOTH ZERO
[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0
∇
∇ R←N CDIV D
[1] A COMMERCIAL DIVISION: RETURNS ZERO IF D=0
[2] R←(N×R)÷D+~R←D≠0
∇

EXAMPLES:

A←2 0 2 0 ALL COMBINATIONS OF N AND D BEING ZERO
B←3 3 0 0
A ZDIV B
0.6666666667 0 0 1
↑ NON-ZERO DIVIDED BY ZERO

ANALYSIS:

4 0 4 0 ZDIV 2 2 0 0

[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0
1 1 0 0
[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0
0 1 0 1
[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0
-----DIVISION MAY PROCEED FOR THESE CASES
1 1 0 1
[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0
-----BUT NOT THIS CASE
0 0 1 0
[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0
-----SOME GOOD DIVISORS
2 2 0 0
[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0
-----ALL GOOD DIVISORS
2 2 1 0
[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0
-----ALL GOOD NUMERATORS
4 0 0 0
[3] R←(N×R)÷(D×R)+~R←(N=0)∨D≠0

2 0 0 1

Section VI

Utility & Miscellaneous Functions

COMB ALL COMBINATIONS OF ELEMENTS [DEBLANK UNIQ]

SYNTAX: R←A COMB B

- JUXTAPOSES EACH UNIQUE ELEMENT OF A WITH EACH UNIQUE ELEMENT OF B, DISREGARDING BLANKS.
- A AND B CAN BE CHARACTER OR NUMERIC STRUCTURES.
- USES: ∇CFORMAT ∇DEBLANK ∇UNIQ

FUNCTIONS:

```

∇ R←A COMB B
[1]  A CFORMAT, DEBLANK, AND UNIQ CLEAN UP
[2]  A GLOBALS A AND B, WHICH ARE LOCAL HERE.
[3]  CFORMAT
[4]  DEBLANK
[5]  UNIQ
[6]  R←(,Q((ρB),ρA)ρA),[□IO+0.5],((ρA),ρB)ρB
∇

```

EXAMPLES:

```

      'AABC'COMB16
A1
A2
A3
A4
A5      ∇ DEBLANK
A6      [1]  A←(A≠' ')/A←,A Δ B←(B≠' ')/B←,B
B1      ∇
B2
B3
B4      ∇ UNIQ
B5      [1]  A←DREP A Δ B←DREP B
B6      ∇
C1
C2
C3
C4
C5
C6

      1 2 3 COMB 234 345 1.1
      1      234
      1      345
      1      1.1
      2      234
      2      345
      2      1.1
      3      234
      3      345
      3      1.1

```

CVEC

BUILD COMPRESSION OR LOGICAL VECTOR

SYNTAX: R←N CVEC LOC

- BINARY VECTORS OF ARBITRARY LENGTH WITH ARBITRARY ZEROS AT NUMBERED POSITIONS, IN USER'S ORIGIN.
- CAN GENERATE INPUT TO V_XVEC.

FUNCTION:

∇ R←N CVEC LOC
[1] R←Np1
[2] R[LOC]←0
[3] A RETURNS A COMPRESSION VECTOR THAT CAN SELECT ALL BUT LOC
[4] A LOC IS DESIRED ROW OF VLOC (□IO←0), OR SIMILAR NUMERIC VECTOR
[5] A N IS ORIGINAL LENGTH OF AXIS TO BE COMPRESSED
∇

EXAMPLES:

17 CVEC 1 2 3 5 7 11
0 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1
32 CVEC (0=4|121)/121
1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1

DT

DELETE TRAILING INSIGNIFICANT CHARACTERS OR VALUES

SYNTAX: R←V DT A

- INSIGNIFICANT CHARACTERS OR VALUES, AS DEFINED IN V, THAT APPEAR ON THE RIGHT SIDE OF AN ARRAY, WILL BE DROPPED
- THE ORIGINAL RANK OF A WILL BE PRESERVED.
- AN EMPTY ARRAY IS RETURNED IF NOTHING SIGNIFICANT REMAINS.

FUNCTION:

∇ R←V DT A
[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A
∇

EXAMPLES:

0 DT 2 4ρ4↑1
1 TO DELETE TRAILING ZEROS
1

'*'DT 2 4ρ'MN**MNM*'
MN*
MNM

ρ' 'DT 3↑'K'
1 TO DELETE TRAILING BLANKS

ANALYSIS:

'* ?'DT'GOOD ?'

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A

GOOD ?

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A

* ?

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A

0 0 0 0 1 1 1

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A

0 0 0 1 1 1 1

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A

4

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A
FOR RANK>2

4

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A
PROTECTS SIGNIFICANT TRAILERS

4

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A

7

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A

4

[1] R←((⁻1↓ρA),[/ , +/v\φ~A∈V)↑A

GOOD

EASTER COMPUTE THE DATE OF EASTER

SYNTAX: Z←EASTER YEAR

- COMPUTE THE DATE OF EASTER FOR ANY YEAR SINCE 33 AD
- YEAR MAY BE A SINGLE YEAR OR VECTOR OF YEARS.
IT MAY ALSO BE AN ARRAY OF SHAPE (N,2) WHERE THE
SECOND COLUMN IS 0 OR 1 FOR EACH YEAR STATING
WHETHER THE OLD (0) OR NEW (1) STYLE CALENDAR
WAS IN EFFECT THEN. NORMALLY, THIS IS COMPUTED
AUTOMATICALLY.

FUNCTION:

```

V Z←EASTER YS;C;EPACT;G;N;X;Y
[1] A COMPUTE EASTER FOR YEAR Y, OPTIONAL STYLE S.
[2] A YS MAY ALSO BE A VECTOR OF YEARS OR AN ARRAY OF YEARS AND STYLES.
[3] YS←(2↑(ρYS),1 1)ρYS
[4] S←(Y>1922)∨(Y>1583)^(YS,1752<Y+YS[;⌊IO])[;⌊IO+1]
[5] →0×1ρZ←(33>⌊Y)/'EASTER WASN'T CELEBRATED THAT EARLY.'
[6] X←S×2-⌊0.75×C+1+⌊0.01×Y
[7] EPACT←30|20+(S×10+⌊0.32×C-15)+(11×G+1+19|Y)+X
[8] N←44-EPACT+S×(EPACT=24)∨(EPACT=25)^G>11
[9] N←N+30×N<21
[10] N←N+7-7|N+7|X+⌊1.25×Y
[11] →0×11≠1↑ρZ←N
[12] Z←'EASTER ON ',((6×30.5-N)↑'MARCH APRIL '), (⌈1+31|⌊1+N),', ',⌈1↑Y
V
```

EXAMPLES:

```

EASTER 1978
EASTER ON MARCH 26, 1978
EASTER 1865
EASTER ON APRIL 16, 1865
EASTER 1
EASTER WASN'T CELEBRATED THAT EARLY.
A A VECTOR INPUT PRODUCES A VECTOR OUTPUT OF THE
A DAY NUMBERS IN MARCH.
EASTER 1978 1865
26 47
A WHEN OLD STYLE WAS KNOWN TO BE IN USE AFTER 1752,
A OR NEW STYLE BEFORE THEN, YOU MUST GIVE THE STYLE.
A FOR EXAMPLE, RUSSIA BEFORE THE REVOLUTION:
EASTER 1 2ρ1865 0
EASTER ON APRIL 11, 1865
```

EXTEND *EXTEND VECTOR WITH LAST VALUE*

SYNTAX: $R \leftarrow N \text{ EXTEND } V$

- THE APL \uparrow WOULD EXTEND A VECTOR BY PADDING IT WITH ZEROS OR BLANKS.
- EXTEND WILL FILL THE SPACE REMAINING ON THE RIGHT WITH THE RIGHTMOST VALUE.
- THIS WILL HAPPEN ONLY IF $N > \rho V$.
- EXTEND RETURNS A VECTOR OF LENGTH N , OR ρV , WHICHEVER IS GREATER.

FUNCTION:

```

      ▽ R←N EXTEND V
[1]   R←V,(0[N-ρV)ρ-1↑V
      ▽

```

EXAMPLES:

```

      10 EXTEND 0 0 0 1
0 0 0 1 1 1 1 1 1 1
      (30 EXTEND 'INDEX ITEM-'),' 20'
INDEX ITEM----- 20
      12 EXTEND 'THIS WILL NOT BE PADDED WITH-'
THIS WILL NOT BE PADDED WITH-

```

ANALYSIS:

```

      33 EXTEND'ITEM 4.'

```

```

[1]   R←V,(0[N-ρV)ρ-1↑V

```

ITEM 4.

```

[1]   R←V,(0[N-ρV)ρ-1↑V

```

```

.
[1]   R←V,(0[N-ρV)ρ-1↑V

```

```

7
[1]   R←V,(0[N-ρV)ρ-1↑V

```

```

26
[1]   R←V,(0[N-ρV)ρ-1↑V

```

```

26
[1]   R←V,(0[N-ρV)ρ-1↑V

```

```

.....
[1]   R←V,(0[N-ρV)ρ-1↑V

```

ITEM 4.....

FILLS

REPLACE VACANT ELEMENTS [CFORMAT CONFORM STRUCT Δ]

SYNTAX:

$R \leftarrow A \text{ FILLS } B$

- THE STRUCTURE A, WHICH MAY BE SCALAR, WILL APPEAR IN VACANT SPACE OF B. IN A NUMERIC STRUCTURE ZERO SIGNIFIES VACANCY. DISPARATE STRUCTURES WILL BE MADE TO CONFORM. UNLESS OFFSET, THE FIRST ELEMENT OF A WILL MAP INTO THE FIRST ELEMENT OF B. IF ONE, BUT NOT BOTH OF THE OPERANDS, IS NUMERIC, IT WILL BE CONVERTED TO CHARACTER FORM.
- USES: ∇ CFORMAT ∇ CONFORM

FUNCTIONS:

$\nabla R \leftarrow A \text{ FILLS } B$
[1] CFORMAT
[2] CONFORM
[3] $R \leftarrow (\rho B) \rho (B = 1 \uparrow 0 \rho B) \ominus B, [\square IO - 0.5] A$
 ∇

EXAMPLES:

EX
X X
X X
X
X X
X X
QUADX
 $\square \square$
 $\square \square$
 \square
 $\square \square$
 $\square \square$
NULLX
.
.
.
.
.
.
.
.
.
.
(' ', ' ', NULLX) FILLS (' ', QUADX) FILLS EX
X \square . X \square .
X \square X \square .
X \square .
X \square X \square .
X \square . X \square .

∇ CONFORM; J; K; R
[1] $R \leftarrow (J \leftarrow \rho \rho A) [K \leftarrow \rho \rho B$
[2] $\rightarrow 0 \text{ IF } 0 = J \times K$
[3] $A \leftarrow R \text{ STRUCT } A \Delta B \leftarrow R \text{ STRUCT } B$
[4] $A \leftarrow R \uparrow A \Delta B \leftarrow (R \leftarrow (\rho A) [\rho B]) \uparrow B$
 ∇
 ∇ X \leftarrow D STRUCT A
[1] ρ D IS DESIRED RANK (DIMENSIONS)
[2] $X \leftarrow ((-D) \uparrow (D \rho 1), \rho A) \rho A$
 ∇
 ∇ R \leftarrow A Δ B
[1] ρ THE SEPARATOR. A AND B MUST RETURN VALUES.
[2] $R \leftarrow A$
 ∇
 ∇ CFORMAT
[1] ρ ASSUMES A AND B HAVE BEEN LOCALIZED
[2] $\& 'A \leftarrow \nabla A \Delta B \leftarrow \nabla B' \text{ IF } (\text{CHARACTER } A) \neq \text{CHARACTER } B$
 ∇

ANALYSIS:

CFORMAT WILL FORCE BOTH A AND B INTO CHARACTER FORM IF ONLY ONE IS SO.
CONFORM WILL PAD THE SMALLER ARRAY TO THE SHAPE OF THE LARGER,
UNLESS EITHER ONE IS SCALAR.
STRUCT REDEFINES THE RANK OF ITS OPERAND.

LOC

LOCATE STRUCTURED DATA

SYNTAX: $I \leftarrow P \text{ LOC } A$

- RESULT IS A MATRIX OF THE STARTING LOCATIONS $\square IO \leftarrow 0$ IF THE ENTIRE STRUCTURE WAS FOUND AT LEAST ONCE.
P IS THE SEARCH ARGUMENT. (SEE ∇ ONESIN)
- USES: $\nabla \Delta$

FUNCTION:

$\nabla I \leftarrow P \text{ LOC } A; \square IO$
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 ∇

EXAMPLE:

'TOP SECRET' LOC 'STOP SECRETARY'

1

ANALYSIS:

5 6 LOC 17
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$ IN CLEAR WS, $\square IO \leftarrow 1$
 1 2 3 4 5 6 7
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$ LOCALLY, $\square IO \leftarrow 0$
 7
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 7
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 0 1 2 3 4 5 6
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 5 6
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 0 0 0 0 1 0 0
 0 0 0 0 0 1 0
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 0 1
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 0 0 0 0 1 0 0
 0 0 0 0 1 0 0
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 4
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 7
 [1] $I \leftarrow (\rho A) \tau(\wedge f(\iota \rho, P) \phi(, P) \circ . =, A) / \iota \times / \rho A \Delta \square IO \leftarrow 0$
 4

LOGICAL MISCELLANEOUS [INTEGER FLOATING EMPTY]

SYNTAX: T ← LOGICAL A

- RETURN 1 IF THE STRUCTURE SATISFIES CONDITION,
OTHERWISE, 0.

FUNCTIONS:

```

      ∇ T ← LOGICAL A
[1]   T ← ∧ / ( , A ) ∈ 1 0
      ∇
      ∇ T ← INTEGER A
[1]   → ( CHARACTER A ) / T ← 0
[2]   T ← 0 ∧ . = 1 | , A
      ∇
      ∇ T ← FLOATING A
      ρ DEF'N: FLOATING = 1, AS USED HERE, MEANS AT LEAST ONE
      ρ MEMBER OF THE ARGUMENT IS NOT AN INTEGER.
[1]   T ← ( ~ INTEGER A ) ∧ ( ~ LOGICAL A ) ∧ ~ CHARACTER A
      ∇
      ∇ T ← EMPTY A
[1]   T ← 0 = ρ , A
      ∇

```

EXAMPLES:

```

      LOGICAL 1 4
0
      LOGICAL 1 □ IO ← 1
1
      EMPTY 1 0
1
      INTEGER 1 1
1      (ONES AND ZEROS ARE INTEGERS)
      FLOATING 1 1
0
      FLOATING 0 . 1
1
      CHARACTER 1
0
      CHARACTER ' 1 '
1

```

NUMBLANKCOLS COUNTS BLANK COLUMNS AT SIDES OF STRUCTURE

SYNTAX: Z←NUMBLANKCOLS A

◦ RETURNS A TWO-ELEMENT VECTOR REPRESENTING COLUMNS OF
SUCCESSIVE BLANKS ON THE LEFT AND RIGHT SIDE OF STRUCTURE.

FUNCTION:

```

      ▽ Z←NUMBLANKCOLS A
[1]   A←⌵⌵' '=▽MATRIX A
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
      ▽

```

EXAMPLES:

```

      NUMBLANKCOLS 3 4ρ' '
0 0
      NUMBLANKCOLS ' ',3 4ρ'K'
1 0
      NUMBLANKCOLS 3 4ρ' K '
2 1

```

ANALYSIS: NUMBLANKCOLS 2 5ρ' AB '

```

[1]   A←⌵⌵' '=▽MATRIX A
      AB
      AB
[1]   A←⌵⌵' '=▽MATRIX A

```

```

1 0 0 1 1
1 0 0 1 1
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
1 1 0 0 1
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
      FROM THE RIGHT

```

```

3
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
2
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
2 3
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
      ORIGIN INDEPENDENT

```

```

-1
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
1 2
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
5
[2]   Z←(ρA)|(-□IO)+(A⌵0),(ϕA)⌵0
      ZEROS IF ALL BLANK

```

1 2

ONESIN

LOCATE ONES IN NUMERIC STRUCTURE

SYNTAX:

R←ONESIN A

AN ARRAY OF ONES AND ZEROS MAY HAVE BEEN THE RESULT OF A TEST OF ANOTHER ARRAY. THIS FUNCTION WILL CONVERT THE ONES TO THEIR OWN LOCATIONS ($\square IO \leftarrow 0$) BY COLUMNS, THAT CAN READILY BE USED TO GENERATE SUBSCRIPTS THAT RELATE TO THE SOURCE.

FUNCTION:

```

      ▽ R←ONESIN A;  $\square IO$ 
[1]   R←( $\rho A$ )↑R/1ρR←,1=A+ $\square IO \leftarrow 0$ 
      ▽
```

EXAMPLE:

```

      A←2 3 4ρ15
      A
0  1  2  3
4  0  1  2
3  4  0  1

2  3  4  0
1  2  3  4
0  1  2  3
      ONESIN A
0  0  0  1  1
0  1  2  1  2
1  2  3  0  1
```

THRU GENERATE INDICES OR OTHER EQUAL INCREMENTS BETWEEN LIMITS

SYNTAX: R←F THRU TB

- TO PRODUCE NUMERIC VECTORS WITH INTEGRAL OR FRACTIONAL INCREMENTS OR DECREMENTS

FUNCTION:

∇ R←F THRU TB;□IO;B
 [1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0
 [2] A GENERATES EQUAL INTERVALS BETWEEN LIMITS F (A SCALAR) AND 1+TB
 [4] A 1+TB←THE DESIRED INTERVAL,E.G.,1,01,0.125,360,ETC.

EXAMPLE:

6 THRU 11 2
 6 8 10

ANALYSIS: 47 THRU 43 0.5

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

0.5

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

47

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

-4

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

-8

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

SIGN CAPTURED

8

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

FOR FRACTIONS

8

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

FOR END-POINT

9

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

0 1 2 3 4 5 6 7 8

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

SCALE

0 0.5 1 1.5 2 2.5 3 3.5 4

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

SIGN APPLIED

-1

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

0 -0.5 -1 -1.5 -2 -2.5 -3 -3.5 -4

[1] R←F+(×R)×B×1+L|R←(TB[0]-F)÷B←|TB[1]+□IO←0

47 46.5 46 45.5 45 44.5 44 43.5 43

SYNTAX:

Z←TABLE TLU ARGS

- RETURNS A MATRIX OF SUBSTITUTIONS CORRESPONDING TO A MATRIX OF ARGUMENTS. THE SUBSTITUTIONS ARE FOUND IN A TABLE WHOSE INITIAL COLUMNS WILL BE MATCHED AGAINST ANY NUMBER OF ARGUMENTS, IN ANY ORDER.
- THE ARGUMENTS ARE USUALLY PRESENTED AS A MATRIX, BUT A SINGLE ARGUMENT MAY BE VECTOR OR SCALAR.
- UNDISCOVERED FUNCTIONS WILL BE RETURNED AS BLANKS (OR ZEROS).
- THE UNMATCHED ARGUMENTS WILL BE REPORTED AT THE TERMINAL.
- IF THE ARGUMENT PORTION OF THE TABLE IS NOT UNIQUE, THE FUNCTION OF THE FIRST OCCURRENCE OF THE ARGUMENT IN THE TABLE WILL BE RETURNED.
- USES: ∇HANG, WHICH PRESERVES THE STACK FOR ANALYSIS.
 ∇FIRSTM TO REMOVE DUPLICATES FROM TABLE.
 ∇IS TO CHECK WHETHER TABLE AND ARGUMENT ARE
 EITHER BOTH NUMERIC, OR BOTH CHARACTER.
 ∇MATRIX ∇IF ∇ON

FUNCTIONS:

```

∇ Z←TABLE TLU ARGS;W;R;L
[1] 'ARGS AND TABLE DISAGREE' HANG~TABLE IS ARGS←MATRIX ARGS
[2] TABLE←(FIRSTM TABLE[;1↑ρARGS])↑TABLE
[3] L←∇/R←ARGS∧.=QTABLE[;1W←1↑ρARGS]
[4] Z←L∧(0,W)↑TABLE[(,R)/,(ρR)ρ1↑ρR;]
[5] →0 IF∧/L
[6] 'NOT FOUND:' ON(~L)↑ARGS
[7] ''
∇
∇ R←A IS B
[1] A TRUE, IF BOTH NUMERIC.
[2] A OR IF BOTH CHARACTER.
[3] R←(0≠0\0ρA)=0≠0\0ρB
∇

```

EXAMPLES:

ARGS	TABLE	SARGS
D03	D01EDUCATION	D03
D01	D02SYSTEMS SUPP	D01
D4A	D03MKTG SERV	D4A
D02	D4AMARKETING	XXX
D01		D02
D03		D01
D4A		D03
D02		

TABLE TLU ARGS	TABLE TLU SARGS
MKTG SERV	NOT FOUND:
EDUCATION	XXX
MARKETING	
SYSTEMS SUPP	MKTG SERV
EDUCATION	EDUCATION
MKTG SERV	MARKETING
MARKETING	
SYSTEMS SUPP	SYSTEMS SUPP
	EDUCATION
	MKTG SERV

EXPAND LOGICAL VECTOR

- A BINARY INDICATION OF A COMPRESSED DATA STRUCTURE WILL BE TRANSFORMED INTO AN EXPANSION VECTOR THAT CAN INJECT W SPACES (OR W ZEROS IN A NUMERIC STRUCTURE) AHEAD OF THE FIELD OR GROUP TO BE SHIFTED.
- SINCE THE EXPANSION CAN BE MADE ALONG ANY AXIS, THE LENGTH OF THE BINARY VECTOR, B, MUST EQUAL THE LENGTH OF THE AXIS.

```

      ▽ R←W XVEC B
[1]   R←(1-1↑R+~□IO)∈R←(1pB)++\W×B←~B
[2]   A B IS A LOGICAL VECTOR, WITH ZEROS INDICATING THE BEGINNING
[3]   A OF EACH FIELD, BEFORE WHICH Wp0 WILL BE INSERTED.
[4]   A THE ORIGINAL ZEROS WILL BE CONVERTED TO ONES.

```

```

      A←□←'  'SHAPE'TOM DICK HARRY'
TOM
DICK
HARRY
      B←□←1 XVEC 1 0 1
1 0 1 1
      B\A
TOM
DICK
HARRY

```

[illegible]

APPENDIX

BIBLIOGRAPHY

IBM Publications

APL Language, GC26-3847

APL Shared Variables (APLSV) Version 3 User's Guide, SH20-9087

APL Shared Variables (APLSV) Version 3 Operations Guide, SH20-9088

VS APL General Information, GH20-9064

VS APL for CMS: Writing Auxiliary Processors, SH20-9068

VS APL for CMS: Terminal User's Guide, SH20-9067

VS APL for VSPC: Terminal User's Guide, SH20-9066

VS APL for TSO: Yale University Terminal User's Guide, SH20-1872

VSPC Installation Reference Material, SH20-9072

OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838

An Introduction to the IBM 3270 Data Analysis APL Feature, GA27-2788

VS TSIO Guide and Reference, SH20-9107

Non-IBM Publications

Iverson, K. E., *A Programming Language*, John Wiley & Sons, New York, 1962

Falkoff, A. D., Iverson, K. E. and Sussenguth, E. H., *A Formal Description of System/360*, IBM Systems Journal, Vol. 3, Nos. 2 & 3, 1964

Polivka, R.P. and Pakin, S., *APL: The Language and its Usage*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1975 (Contains an *extensive APL bibliography*)

Gilman, L. and Rose, A.J., *APL - An Interactive Approach*, Second Edition Revised Reprinting, John Wiley & Sons, New York, 1976

Harms, E. and Zabinski, M. P., *Introduction to APL and Computer Programming*, John Wiley & Sons, New York 1977

KWIC INDEX

Suppose you require a technique to solve a particular problem. You suspect that within the handbook there is a function which can help, but you do not know its name. How do you locate it?

Scan the keywords for a subject reference. When you find it, you will see (within the same abstract) the name of the **APL** function you need.

Conversely, you may determine the purpose of a function if you know only its name. Use the function name as a keyword to yield the appropriate abstract.

∇ADD	MULTIPRECISION INTEGER ADDITION	
ADD COLUMNS TO A MATRIX VECTOR OR SCALAR		∇ADDCOLS
ADD ROWS TO A MATRIX VECTOR OR SCALAR		∇ADDDROWS
∇ADDCOLS	ADD COLUMNS TO A MATRIX VECTOR OR SCALAR	
ADDITION	∇ADD	MULTIPRECISION INTEGER
ADDITION	∇FADD	MULTIPRECISION FLOATING POINT
∇ADDDROWS	ADD ROWS TO A MATRIX VECTOR OR SCALAR	
ADJACENT ELEMENTS [UNSCAN]	∇DIFF	DIFFERENCES BETWEEN
∇ADJUSTDOWN	EXTEND THE ' ' IN REPORT FORMATTING [ROWINDICES]	
∇ADJUSTUP	EXTENDS ' ' IN REPORT FORMATTING	
∇ALPREC	ALTER PRECISION OF A SCALAR OR MULTIPRECISION NUMBER	
ALT] ∇TIME	RUNNING TIME AND NEW SPACE FOR AN APL STATEMENT [
ALTER PRECISION OF A SCALAR OR MULTIPRECISION NUMBER		∇ALPREC
∇AMORTIZE	MORTGAGE CALCULATION BY MONTHS	
APL STATEMENT [ALT] ∇TIME	RUNNING TIME AND NEW SPACE FOR AN	
APPEARANCE IN MATRIX [FIRSTV]	∇FIRSTM	SELECT FIRST OR ONLY
ARBITRARY PRECISION	∇PI	COMPUTE PI TO
ARBITRARY SCALAR UNIT∇DROUNDS		DISTRIBUTIVE ROUNDING OF A VECTOR TO
ARGUMENTS [IS]	∇TLU	TABLE LOOK-UP OF STRUCTURED
ARRAY	∇REPL	REPLACE ALL OCCURRENCES OF ELEMENT IN
ARRAY [DLB RJUST DL]	∇LJUST	LEFT JUSTIFY ANY
ARRAY TO NUMERIC PATTERN	∇CHAR	BUILD CHARACTER
ARRAY [MATRIX CHARACTER]	∇FRAME	FRAME AN
ARRAYS	∇CITED	EXTRACT CITED STRINGS FROM CHARACTER
ARRAYS WITH BLANKS OR ZEROS	∇PAD	PADS
ASCENDING ROW INDICES [AV NFORM LJNFORM]	∇GRADEUP	GENERATE
ASTRONOMERS [MOONPHASE]	∇DAYNO	DAY NUMBER FOR
ASTRONOMERS' DAY NUMBER	∇DATE	COMPUTE NORMAL DATE FROM
AT SIDES OF STRUCTURE	∇NUMBLANKCOLS	COUNTS BLANK COLUMNS
AV NFORM LJNFORM]	∇GRADEUP	GENERATE ASCENDING ROW INDICES [
∇BARGRAPH	PLOT HORIZONTAL INTEGER BARGRAPHS	
BARGRAPHS	∇BARGRAPH	PLOT HORIZONTAL INTEGER
BASE [DIGITS CONFRAC]	∇CONV	CONVERT DECIMAL VALUES TO ANY
BELONG TO A	∇INDEX	COLUMN INDEX IN MATRIX B WHOSE MEMBERS ALL
∇BESIDE	PRESENTS TWO STRUCTURES SIDE BY SIDE IN REPORT FORMAT	
∇BLANK	DELETE SPECIFIC STRING FROM STRUCTURE [LIM]	
BLANK COLUMNS AT SIDES OF STRUCTURE	∇NUMBLANKCOLS	COUNTS
BLANKS OR ZEROS	∇PAD	PADS ARRAYS WITH
BUILD CHARACTER ARRAY TO NUMERIC PATTERN	∇CHAR	
BUILD COMPRESSION OR LOGICAL VECTOR	∇CVEC	
BY COLUMNS [VERTAB CFORMAT CMATRIX ROWFORM]	∇CCAT	CATENATE

BY IN FROM]	∇TO	NUMERIC VECTORS IN EQUAL INCREMENTS [
BY MONTHS		∇AMORTIZE MORTGAGE CALCULATION
BY ROW OF A MATRIX [ESCAPE ESCAPEX]	∇VFORM	VARIABLE FORMAT
BY ROWS [COLFORM CHARACTER VERT]	∇RCAT	CATENATE STRUCTURES
BY SIDE IN REPORT FORMAT	∇BESIDE	PRESENTS TWO STRUCTURES SIDE
CALCULATION BY MONTHS		∇AMORTIZE MORTGAGE
CALCULATIONS [DATES NDATES PAYDAY]		∇DAYS DATE
∇CAN	EDIT MULTIPRECISION INTEGERS INTO CANONICAL FORMAT	
CANONICAL FORMAT	∇CAN	EDIT MULTIPRECISION INTEGERS INTO
CATENATE ANY STRUCTURES		∇ON CONFORM AND
CATENATE BY COLUMNS [VERTAB CFORMAT CMATRIX ROWFORM]		∇CCAT
CATENATE STRUCTURES BY ROWS [COLFORM CHARACTER VERT]		∇RCAT
CATENATES TWO STRUCTURES [CENTER]		∇CENTERON CENTERS AND
∇CCAT	CATENATE BY COLUMNS [VERTAB CFORMAT CMATRIX ROWFORM]	
CDIV]	∇ZDIV	ZERO TOLERANT DIVISION [
CENTER HEADINGS OVER FORMATTED COLUMNS [CNTR DMZ NEXTA]		∇OUTPUT
CENTER]	∇CENTERON	CENTERS AND CATENATES TWO STRUCTURES [
∇CENTERON	CENTERS AND CATENATES TWO STRUCTURES [CENTER]	
CENTERS AND CATENATES TWO STRUCTURES [CENTER]		∇CENTERON
CFORMAT CMATRIX ROWFORM]	∇CCAT	CATENATE BY COLUMNS [VERTAB
CFORMAT CONFORM STRUCT Δ]	∇FILLS	REPLACE VACANT ELEMENTS [
∇CHAR	BUILD CHARACTER ARRAY TO NUMERIC PATTERN	
CHARACTER ARRAY TO NUMERIC PATTERN		∇CHAR BUILD
CHARACTER ARRAYS	∇CITED	EXTRACT CITED STRINGS FROM
CHARACTER MATRIX EXPAND RESULT [V2M]		∇M2V COMPRESS
CHARACTER MATRIX [USCORE]	∇VLINE	UNDERLINE SPECIFIED ROWS OF
CHARACTER STRING		∇SHAPE SHAPE MATRIX FROM
CHARACTER STRING	∇FORMAT	CONVERT MULTIPRECISION NUMBER TO
CHARACTER STRUCTURE [DLTMB]	∇ERECT	ERECT WORD MATRIX FROM
CHARACTER STRUCTURE [DTMB]		∇WORD SELECT NTH WORD IN
CHARACTER STRUCTURE [SEDIT]	∇EDIT	EDIT LATENT EXPRESSION OR
CHARACTER VERT]	∇RCAT	CATENATE STRUCTURES BY ROWS [COLFORM
CHARACTER]		∇FRAME FRAME AN ARRAY [MATRIX
CHARACTER]	∇LOGICAL	MISCELLANEOUS [INTEGER FLOATING EMPTY
CHARACTERISTICS OR CONTENTS OF VARS SELECTIVELY	∇VARS	DISPLAY
CHARACTERS	∇TCC	SYSTEM INDEPENDENT TERMINAL CONTROL
CHARACTERS OR VALUES	∇DT	DELETE TRAILING INSIGNIFICANT
CHECK TERMINAL ENTRY OR DEFAULT		∇PROMPT PROMPT AND
CHECKS A MATRIX FOR FRAMING		∇FRAMETEST
CHI SQUARE PROBABILITY FUNCTION		∇QPROBF COMPUTE
∇CITED	EXTRACT CITED STRINGS FROM CHARACTER ARRAYS	

CITED STRINGS FROM CHARACTER ARRAYS	VCITED	EXTRACT
CMATRIX ROWFORM] VCCAT	CATENATE BY COLUMNS [VERTAB CFORMAT	
CNTR DMZ NEXTA] VOUTPUT	CENTER HEADINGS OVER FORMATTED COLUMNS [
CODES VCOLLECT	COLLECT AND SUMMARIZE COEFFICIENTS OF COMMON	
COEFFICIENTS OF COMMON CODES VCOLLECT	COLLECT AND SUMMARIZE	
COLFORM CHARACTER VERT] VRCAT	CATENATE STRUCTURES BY ROWS [
VCOLLECT	COLLECT AND SUMMARIZE COEFFICIENTS OF COMMON CODES	
COLLECT AND SUMMARIZE COEFFICIENTS OF COMMON CODES VCOLLECT		
COLNO] VTABS	COMPARE REQUIRED TAB SETTINGS TO EXISTING ONES [
COLUMN INDEX IN MATRIX B WHOSE MEMBERS ALL BELONG TO A VINDEX		
COLUMNS AT SIDES OF STRUCTURE VNUMBLANKCOLS	COUNTS BLANK	
COLUMNS TO A MATRIX VECTOR OR SCALAR VADDCOLS	ADD	
COLUMNS [CNTR DMZ NEXTA] VOUTPUT	CENTER HEADINGS OVER FORMATTED	
COLUMNS [VERTAB CFORMAT CMATRIX ROWFORM] VCCAT	CATENATE BY	
VCOMB	ALL COMBINATIONS OF ELEMENTS [DEBLANK UNIQ]	
COMBINATIONS OF ELEMENTS [DEBLANK UNIQ] VCOMB	ALL	
COMMON CODES VCOLLECT	COLLECT AND SUMMARIZE COEFFICIENTS OF	
COMPARE REQUIRED TAB SETTINGS TO EXISTING ONES [COLNO] VTABS		
COMPARE] VHEADERON	PUTS A HEADING ON A REPORT [
COMPRESS CHARACTER MATRIX EXPAND RESULT [V2M] VM2V		
COMPRESSION OR LOGICAL VECTOR VCVEC	BUILD	
COMPUTE CHI SQUARE PROBABILITY FUNCTION VQPROBF		
COMPUTE NORMAL DATE FROM ASTRONOMERS' DAY NUMBER VDATE		
COMPUTE PI TO ARBITRARY PRECISION VPI		
COMPUTE THE DATE OF EASTER VEASTER		
CONFORM AND CATENATE ANY STRUCTURES VON		
CONFORM STRUCT A] VFILLS	REPLACE VACANT ELEMENTS [CFORMAT	
CONFRAC] VCONV	CONVERT DECIMAL VALUES TO ANY BASE [DIGITS	
CONTENTS OF VARS SELECTIVELY VVARS	DISPLAY CHARACTERISTICS OR	
CONTROL CHARACTERS VTCC	SYSTEM INDEPENDENT TERMINAL	
CONTROLLED FORMAT [HANG] VTABULATE	NUMERIC STRUCTURES IN	
VCONV	CONVERT DECIMAL VALUES TO ANY BASE [DIGITS CONFRAC]	
CONVERT DECIMAL VALUES TO ANY BASE [DIGITS CONFRAC] VCONV		
CONVERT INTEGER TO ROMAN NUMERALS VROMAN		
CONVERT MULTIPRECISION NUMBER TO CHARACTER STRING VFORMAT		
CONVERT TO DECIMAL VDEC		
CONVERT TO MULTIPRECISION FLOATING POINT [SCALE] VFLOAT		
CONVERT TO MULTIPRECISION INTEGER VFIX		
COUNTS BLANK COLUMNS AT SIDES OF STRUCTURE VNUMBLANKCOLS		
CURRENT SESSION AND WORKSPACE STATUS [NOW] VSTATUS		
VCVEC	BUILD COMPRESSION OR LOGICAL VECTOR	

DATA		VLLOC	LOCATE STRUCTURED
VDATE	COMPUTE NORMAL DATE FROM ASTRONOMERS' DAY NUMBER		
DATE CALCULATIONS [DATES NDATES PAYDAY]		VDATE	COMPUTE NORMAL
DATE FROM ASTRONOMERS' DAY NUMBER		VEASTER	COMPUTE THE
DATE OF EASTER		VDATE	DATE CALCULATIONS [
DATES NDATES PAYDAY]		VDATE	COMPUTE NORMAL DATE FROM ASTRONOMERS'
DAY NUMBER		VDATE	DAYNO
DAY NUMBER FOR ASTRONOMERS [MOONPHASE]		VDATE	
VDAYNO	DAY NUMBER FOR ASTRONOMERS [MOONPHASE]		
VDATE	DATE CALCULATIONS [DATES NDATES PAYDAY]		
DEBLANK UNIQ]	VCOMB		ALL COMBINATIONS OF ELEMENTS [
VDEC	CONVERT TO DECIMAL		
DECIMAL		VDEC	CONVERT TO
DECIMAL VALUES TO ANY BASE [DIGITS CONFRAC]	VCONV		CONVERT
DEFAULT	VPROMPT		PROMPT AND CHECK TERMINAL ENTRY OR
DELETE SPECIFIC STRING FROM STRUCTURE [LIM]		VBLANK	
DELETE TRAILING INSIGNIFICANT CHARACTERS OR VALUES		VDT	
VDIFF	DIFFERENCES BETWEEN ADJACENT ELEMENTS [UNSCAN]		
DIFFERENCES BETWEEN ADJACENT ELEMENTS [UNSCAN]		VDIFF	
DIGITS	VTRUNC		TRUNCATE HIGHER AND LOWER ORDER
DISPLAY CHARACTERISTICS OR CONTENTS OF VARS SELECTIVELY	VVARS		
DISTRIBUTIVE ROUNDING OF A VECTOR TO ARBITRARY SCALAR UNIT	VDROUNDS		
VDIV	MULTIPRECISION INTEGER DIVISION		
DIVISION		VDIV	MULTIPRECISION INTEGER
DIVISION	VFDIV		MULTIPRECISION FLOATING POINT
DIVISION [CDIV]		VZDIV	ZERO TOLERANT
DL]	VLJUST		LEFT JUSTIFY ANY ARRAY [DLB RJUST
DLB RJUST DL]	VLJUST		LEFT JUSTIFY ANY ARRAY [
DLTMB]	VERECT		ERECT WORD MATRIX FROM CHARACTER STRUCTURE [
DMZ NEXTA]	VOUTPUT		CENTER HEADINGS OVER FORMATTED COLUMNS [CNTR
VDREP	SELECT UNIQUE ELEMENTS FROM ANY STRUCTURE		
VDROUNDS	DISTRIBUTIVE ROUNDING OF A VECTOR TO ARBITRARY SCALAR UNIT		
VDT	DELETE TRAILING INSIGNIFICANT CHARACTERS OR VALUES		
DTMB]	VWORD		SELECT NTH WORD IN CHARACTER STRUCTURE [
EASTER		VEASTER	COMPUTE THE DATE OF
VEASTER	COMPUTE THE DATE OF EASTER		
VEDIT	EDIT LATENT EXPRESSION OR CHARACTER STRUCTURE [SEDIT]		
EDIT LATENT EXPRESSION OR CHARACTER STRUCTURE [SEDIT]		VEDIT	
EDIT MATRIX		VMEDIT	
EDIT MULTIPRECISION INTEGERS INTO CANONICAL FORMAT		VCAN	
EDITING [POSTEDIT]	VPREEDIT		PREPARE MATRIX FOR FUNCTION-LIKE

ELEMENT IN ARRAY	∇REPL	REPLACE ALL OCCURRENCES OF
ELEMENTS [DEBLANK UNIQ]	∇COMB	ALL COMBINATIONS OF
ELEMENTS [SORTDA KFORM]	∇FREQ	FREQUENCY DISTRIBUTION OF
ELEMENTS [UNSCAN]	∇DIFF	DIFFERENCES BETWEEN ADJACENT
ELEMENTS FROM ANY STRUCTURE	∇DREP	SELECT UNIQUE
ELEMENTS [CFORMAT CONFORM STRUCT Δ]	∇FILLS	REPLACE VACANT
EMPTY CHARACTER]	∇LOGICAL	MISCELLANEOUS [INTEGER FLOATING
∇ENC		GENERATE SUFFICIENT ENCODING POSITIONS
ENCODING POSITIONS	∇ENC	GENERATE SUFFICIENT
ENTRY OR DEFAULT	∇PROMPT	PROMPT AND CHECK TERMINAL
EQUAL INCREMENTS BETWEEN LIMITS	∇THRU	GENERATE INDICES OR OTHER
EQUAL INCREMENTS [BY IN FROM]	∇TO	NUMERIC VECTORS IN
∇ERECT		ERECT WORD MATRIX FROM CHARACTER STRUCTURE [DLTMB]
ERECT WORD MATRIX FROM CHARACTER STRUCTURE [DLTMB]	∇ERECT	
ESCAPE ESCAPEX]	∇VFORM	VARIABLE FORMAT BY ROW OF A MATRIX [
ESCAPEX]	∇VFORM	VARIABLE FORMAT BY ROW OF A MATRIX [ESCAPE
EXISTING ONES [COLNO]	∇VTABS	COMPARE REQUIRED TAB SETTINGS TO
EXPAND LOGICAL VECTOR		∇XVEC
EXPAND RESULT [V2M]	∇M2V	COMPRESS CHARACTER MATRIX
EXPONENTIAL FUNCTION	∇FEXP	MULTIPRECISION FLOATING POINT
EXPRESSION OR CHARACTER STRUCTURE [SEDIT]	∇EDIT	EDIT LATENT
∇EXTEND		EXTEND VECTOR WITH LAST VALUE
EXTEND THE ' ' IN REPORT FORMATTING [ROWINDICES]		∇ADJUSTDOWN
EXTEND VECTOR WITH LAST VALUE		∇EXTEND
EXTENDS ' ' IN REPORT FORMATTING		∇ADJUSTUP
EXTRACT CITED STRINGS FROM CHARACTER ARRAYS		∇CITED
∇FADD		MULTIPRECISION FLOATING POINT ADDITION
∇FDIV		MULTIPRECISION FLOATING POINT DIVISION
∇FEXP		MULTIPRECISION FLOATING POINT EXPONENTIAL FUNCTION
∇FILLS		REPLACE VACANT ELEMENTS [CFORMAT CONFORM STRUCT Δ]
FIRST OR ONLY APPEARANCE IN MATRIX [FIRSTV]	∇FIRSTM	SELECT
∇FIRSTM		SELECT FIRST OR ONLY APPEARANCE IN MATRIX [FIRSTV]
FIRSTV]	∇FIRSTM	SELECT FIRST OR ONLY APPEARANCE IN MATRIX [
∇FIX		CONVERT TO MULTIPRECISION INTEGER
∇FLOAT		CONVERT TO MULTIPRECISION FLOATING POINT [SCALE]
FLOATING EMPTY CHARACTER]	∇LOGICAL	MISCELLANEOUS [INTEGER
FLOATING POINT ADDITION	∇FADD	MULTIPRECISION
FLOATING POINT DIVISION	∇FDIV	MULTIPRECISION
FLOATING POINT EXPONENTIAL FUNCTION	∇FEXP	MULTIPRECISION
FLOATING POINT MULTIPLICATION	∇FMUL	MULTIPRECISION
FLOATING POINT SQUARE ROOT	∇FSQRT	MULTIPRECISION

FLOATING POINT SUBTRACTION	VFLOAT	MULTIPRECISION
FLOATING POINT [SCALE]	VFLOAT	CONVERT TO MULTIPRECISION
VFLOAT	MULTIPRECISION FLOATING POINT MULTIPLICATION	
VFLOAT	CONVERT MULTIPRECISION NUMBER TO CHARACTER STRING	
FORMAT	VCAN	EDIT MULTIPRECISION INTEGERS INTO CANONICAL
FORMAT	VBESIDE	PRESENTS TWO STRUCTURES SIDE BY SIDE IN REPORT
FORMAT BY ROW OF A MATRIX [ESCAPE ESCAPEX]	VFFORM	VARIABLE
FORMAT [HANG]	VTABULATE	NUMERIC STRUCTURES IN CONTROLLED
FORMATTED COLUMNS [CNTR DMZ NEXTA]	VOUTPUT	CENTER HEADINGS OVER
FORMATTED MATRIX		WIDTH MEASURE
FORMATTING	VADJUSTUP	EXTENDS 'I' IN REPORT
FORMATTING [IF]	VPREPARE	STANDARDIZE STRUCTURE FOR REPORT
FORMATTING [ROWINDICES]	VADJUSTDOWN	EXTEND THE 'I' IN REPORT
VFRAME	FRAME AN ARRAY [MATRIX CHARACTER]	
FRAME AN ARRAY [MATRIX CHARACTER]		VFRAME
VFRAMETEST	CHECKS A MATRIX FOR FRAMING	
FRAMING	VFRAMETEST	CHECKS A MATRIX FOR
VFREQ	FREQUENCY DISTRIBUTION OF ELEMENTS [SORTDA KFORM]	
FREQUENCY DISTRIBUTION OF ELEMENTS [SORTDA KFORM]	VFREQ	
FROM ANY STRUCTURE	VDREP	SELECT UNIQUE ELEMENTS
FROM ASTRONOMERS' DAY NUMBER	VDATE	COMPUTE NORMAL DATE
FROM CHARACTER ARRAYS	VCITED	EXTRACT CITED STRINGS
FROM CHARACTER STRING	VSHAPE	SHAPE MATRIX
FROM CHARACTER STRUCTURE [DLTMB]	VERECT	ERECT WORD MATRIX
FROM STRUCTURE [LIM]	VBLANK	DELETE SPECIFIC STRING
FROM]	VTO	NUMERIC VECTORS IN EQUAL INCREMENTS [BY IN
VFSQRT	MULTIPRECISION FLOATING POINT SQUARE ROOT	
VFSUB	MULTIPRECISION FLOATING POINT SUBTRACTION	
FUNCTION	VQPROBF	COMPUTE CHI SQUARE PROBABILITY
FUNCTION	VFEXP	MULTIPRECISION FLOATING POINT EXPONENTIAL
FUNCTION IN STANDARD FORM	VLISTFN	LISTS A
FUNCTION-LIKE EDITING [POSTEDIT]	VPREEDIT	PREPARE MATRIX FOR
GENERATE ASCENDING ROW INDICES [AV NFORM LJNFORM]	VGRADEUP	
GENERATE INDICES OR OTHER EQUAL INCREMENTS BETWEEN LIMITS	VTHRU	
GENERATE SUFFICIENT ENCODING POSITIONS	VENC	
VGRADEUP	GENERATE ASCENDING ROW INDICES [AV NFORM LJNFORM]	
HANG]	VTABULATE	NUMERIC STRUCTURES IN CONTROLLED FORMAT [
VHEADERON	PUTS A HEADING ON A REPORT [COMPARE]	
HEADING ON A REPORT [COMPARE]	VHEADERON	PUTS A
HEADINGS OVER FORMATTED COLUMNS [CNTR DMZ NEXTA]	VOUTPUT	CENTER
HIGHER AND LOWER ORDER DIGITS	VTRUNC	TRUNCATE

HORIZONTAL INTEGER BARGRAPHS	VBARGRAPH	PLOT
IF]	VPREPARE	STANDARDIZE STRUCTURE FOR REPORT FORMATTING [
INCREMENTS BETWEEN LIMITS VTHRU		GENERATE INDICES OR OTHER EQUAL
INCREMENTS [BY IN FROM]	VTO	NUMERIC VECTORS IN EQUAL
INDEPENDENT TERMINAL CONTROL CHARACTERS	VTCC	SYSTEM
VINDEX	COLUMN INDEX IN MATRIX B WHOSE MEMBERS ALL BELONG TO A	
INDEX IN MATRIX B WHOSE MEMBERS ALL BELONG TO A	VINDEX	COLUMN
INDICES [AV NFORM LJNFORM]	VGRADEUP	GENERATE ASCENDING ROW
INDICES OR OTHER EQUAL INCREMENTS BETWEEN LIMITS VTHRU		GENERATE
INSIGNIFICANT CHARACTERS OR VALUES	VDT	DELETE TRAILING
INTEGER	VFIX	CONVERT TO MULTIPRECISION
INTEGER ADDITION	VADD	MULTIPRECISION
INTEGER BARGRAPHS	VBARGRAPH	PLOT HORIZONTAL
INTEGER DIVISION	VDIV	MULTIPRECISION
INTEGER FLOATING EMPTY CHARACTER]	VLOGICAL	MISCELLANEOUS [
INTEGER MULTIPLICATION	VMUL	MULTIPRECISION
INTEGER SQUARE ROOT	VSQRT	MULTIPRECISION
INTEGER SUBTRACTION	VSUB	MULTIPRECISION
INTEGER TO ROMAN NUMERALS	VROMAN	CONVERT
INTEGERS INTO CANONICAL FORMAT	VCAN	EDIT MULTIPRECISION
INTO CANONICAL FORMAT	VCAN	EDIT MULTIPRECISION INTEGERS
IOTA	VRIOTA	MATRIX ROW
IS]	VTLU	TABLE LOOK-UP OF STRUCTURED ARGUMENTS [
JUSTIFY ANY ARRAY	[DLB RJUST DL]	VLJUST
KFORM]	VFREQ	FREQUENCY DISTRIBUTION OF ELEMENTS [SORTDA
LAST VALUE	VEXTEND	EXTEND VECTOR WITH
LATENT EXPRESSION OR CHARACTER STRUCTURE [SEDIT]	VEDIT	EDIT
LEFT JUSTIFY ANY ARRAY	[DLB RJUST DL]	VLJUST
LIM]	VBLANK	DELETE SPECIFIC STRING FROM STRUCTURE [
LIMITS VTHRU		GENERATE INDICES OR OTHER EQUAL INCREMENTS BETWEEN
VLISTFN	LISTS A FUNCTION IN STANDARD FORM	
LISTS A FUNCTION IN STANDARD FORM		VLISTFN
LJNFORM]	VGRADEUP	GENERATE ASCENDING ROW INDICES [AV NFORM
VLJUST	LEFT JUSTIFY ANY ARRAY	[DLB RJUST DL]
VLOC	LOCATE STRUCTURED DATA	
LOCATE ONES IN NUMERIC STRUCTURE		VONESIN
LOCATE STRUCTURED DATA		VLOC
VLOGICAL	MISCELLANEOUS [INTEGER FLOATING EMPTY CHARACTER]	
LOGICAL VECTOR	VCVEC	BUILD COMPRESSION OR
LOGICAL VECTOR	VXVEC	EXPAND
LOWER ORDER DIGITS	VTRUNC	TRUNCATE HIGHER AND

MATRIX		VMEDIT	EDIT
MATRIX		WIDTH	MEASURE FORMATTED
MATRIX	EXPAND RESULT [V2M]	VM2V	COMPRESS CHARACTER
MATRIX	B WHOSE MEMBERS ALL BELONG TO A	VINDEX	COLUMN INDEX IN
MATRIX	CHARACTER]	VFRAME	FRAME AN ARRAY [
MATRIX	FOR FRAMING	VFRAMETEST	CHECKS A
MATRIX	FOR FUNCTION-LIKE EDITING [POSTEDIT]	VPREEDIT	PREPARE
MATRIX	FROM CHARACTER STRING	VSHAPE	SHAPE
MATRIX	FROM CHARACTER STRUCTURE [DLTMB]	VERECT	ERECT WORD
MATRIX	ROW IOTA		VRIOTA
MATRIX	VECTOR OR SCALAR	VADDCOLS	ADD COLUMNS TO A
MATRIX	VECTOR OR SCALAR	VADDDROWS	ADD ROWS TO A
MATRIX	[ESCAPE ESCAPEX]	VVFORM	VARIABLE FORMAT BY ROW OF A
MATRIX	[FIRSTV]	VFIRSTM	SELECT FIRST OR ONLY APPEARANCE IN
MATRIX	[USCORE]	VULINE	UNDERLINE SPECIFIED ROWS OF CHARACTER
MATRIX	MEASURE FORMATTED MATRIX		WIDTH
VMEDIT	EDIT MATRIX		
MATRIX	MEMBERS ALL BELONG TO A	VINDEX	COLUMN INDEX IN MATRIX B WHOSE
MATRIX	MISCELLANEOUS [INTEGER FLOATING EMPTY CHARACTER]		VLOGICAL
MATRIX	MONTHS	VAMORTIZE	MORTGAGE CALCULATION BY
MATRIX	MOONPHASE]	VDAYNO	DAY NUMBER FOR ASTRONOMERS [
MATRIX	MORTGAGE CALCULATION BY MONTHS		VAMORTIZE
VMUL	MULTIPRECISION INTEGER MULTIPLICATION		
MATRIX	MULTIPLICATION	VMUL	MULTIPRECISION INTEGER
MATRIX	MULTIPLICATION	VMUL	MULTIPRECISION FLOATING POINT
MATRIX	MULTIPRECISION FLOATING POINT ADDITION		VFADD
MATRIX	MULTIPRECISION FLOATING POINT DIVISION		VFDIV
MATRIX	MULTIPRECISION FLOATING POINT EXPONENTIAL FUNCTION		VFEXP
MATRIX	MULTIPRECISION FLOATING POINT [SCALE]	VFLOAT	CONVERT TO
MATRIX	MULTIPRECISION FLOATING POINT MULTIPLICATION		VMUL
MATRIX	MULTIPRECISION FLOATING POINT SQUARE ROOT		VFSQRT
MATRIX	MULTIPRECISION FLOATING POINT SUBTRACTION		VFSUB
MATRIX	MULTIPRECISION INTEGER	VFIX	CONVERT TO
MATRIX	MULTIPRECISION INTEGER ADDITION		VADD
MATRIX	MULTIPRECISION INTEGER DIVISION		VDIV
MATRIX	MULTIPRECISION INTEGER MULTIPLICATION		VMUL
MATRIX	MULTIPRECISION INTEGER SQUARE ROOT		VSQRT
MATRIX	MULTIPRECISION INTEGER SUBTRACTION		VSUB
MATRIX	MULTIPRECISION INTEGERS INTO CANONICAL FORMAT	VCAN	EDIT
MATRIX	MULTIPRECISION NUMBER	VALPREC	ALTER PRECISION OF A SCALAR OR
MATRIX	MULTIPRECISION NUMBER TO CHARACTER STRING	VFORMAT	CONVERT

VM2V	COMPRESS CHARACTER MATRIX	EXPAND RESULT [V2M]
NDATES PAYDAY]	VDAYS	DATE CALCULATIONS [DATES
NEW SPACE FOR AN APL STATEMENT [ALT]	VTIME	RUNNING TIME AND
NFORM LJNFORM]	VGRADEUP	GENERATE ASCENDING ROW INDICES [AV
NORMAL DATE FROM ASTRONOMERS' DAY NUMBER	VDATE	COMPUTE
NOW]	VSTATUS	CURRENT SESSION AND WORKSPACE STATUS [
NTH WORD IN CHARACTER STRUCTURE [DTMB]	VWORD	SELECT
NUMBER	VDATE	COMPUTE NORMAL DATE FROM ASTRONOMERS' DAY
NUMBER	VALPREC	ALTER PRECISION OF A SCALAR OR MULTIPRECISION
NUMBER FOR ASTRONOMERS [MOONPHASE]	VDAYNO	DAY
NUMBER TO CHARACTER STRING	VFORMAT	CONVERT MULTIPRECISION
VNUMBLANKCOLS	COUNTS BLANK COLUMNS AT SIDES OF STRUCTURE	
NUMERALS	VROMAN	CONVERT INTEGER TO ROMAN
NUMERIC PATTERN	VCHAR	BUILD CHARACTER ARRAY TO
NUMERIC STRUCTURE	VONESIN	LOCATE ONES IN
NUMERIC STRUCTURES IN CONTROLLED FORMAT [HANG]	VTABULATE	
NUMERIC VECTORS IN EQUAL INCREMENTS [BY IN FROM]	VTO	
OCCURRENCES OF ELEMENT IN ARRAY	VREPL	REPLACE ALL
ONES [COLNO] VTABS		COMPARE REQUIRED TAB SETTINGS TO EXISTING
ONES IN NUMERIC STRUCTURE	VONESIN	LOCATE
VONESIN	LOCATE ONES IN NUMERIC STRUCTURE	
ONLY APPEARANCE IN MATRIX [FIRSTV]	VFIRSTM	SELECT FIRST OR
ORDER DIGITS	VTRUNC	TRUNCATE HIGHER AND LOWER
OTHER EQUAL INCREMENTS BETWEEN LIMITS VTHRU		GENERATE INDICES OR
VOUTPUT	CENTER HEADINGS OVER FORMATTED COLUMNS [CNTR DMZ NEXTA]	
OVER FORMATTED COLUMNS [CNTR DMZ NEXTA]	VOUTPUT	CENTER HEADINGS
VPAD	PADS ARRAYS WITH BLANKS OR ZEROS	
PADS ARRAYS WITH BLANKS OR ZEROS	VPAD	
PATTERN	VCHAR	BUILD CHARACTER ARRAY TO NUMERIC
PAYDAY]	VDAYS	DATE CALCULATIONS [DATES NDATES
VPI	COMPUTE PI TO ARBITRARY PRECISION	
PI TO ARBITRARY PRECISION	VPI	COMPUTE
PLOT HORIZONTAL INTEGER BARGRAPHS		VBARGRAPH
POINT ADDITION	VFADD	MULTIPRECISION FLOATING
POINT DIVISION	VFDIV	MULTIPRECISION FLOATING
POINT EXPONENTIAL FUNCTION	VFEXP	MULTIPRECISION FLOATING
POINT MULTIPLICATION	VMUL	MULTIPRECISION FLOATING
POINT SQUARE ROOT	VFSQRT	MULTIPRECISION FLOATING
POINT SUBTRACTION	VFSUB	MULTIPRECISION FLOATING
POINT [SCALE]	VFLOAT	CONVERT TO MULTIPRECISION FLOATING
POSITIONS	VENC	GENERATE SUFFICIENT ENCODING

POSTEDIT] ∇PREEDIT PREPARE MATRIX FOR FUNCTION-LIKE EDITING [
 PRECISION ∇PI COMPUTE PI TO ARBITRARY
 PRECISION OF A SCALAR OR MULTIPRECISION NUMBER ∇ALPREC ALTER
 ∇PREEDIT PREPARE MATRIX FOR FUNCTION-LIKE EDITING [POSTEDIT]
 ∇PREPARE STANDARDIZE STRUCTURE FOR REPORT FORMATTING [IF]
 PREPARE MATRIX FOR FUNCTION-LIKE EDITING [POSTEDIT] ∇PREEDIT
 PRESENTS TWO STRUCTURES SIDE BY SIDE IN REPORT FORMAT ∇BESIDE
 PROBABILITY FUNCTION ∇QPROBF COMPUTE CHI SQUARE
 ∇PROMPT PROMPT AND CHECK TERMINAL ENTRY OR DEFAULT
 PROMPT AND CHECK TERMINAL ENTRY OR DEFAULT ∇PROMPT
 PUTS A HEADING ON A REPORT [COMPARE] ∇HEADERON
 ∇QPROBF COMPUTE CHI SQUARE PROBABILITY FUNCTION
 ∇RCAT CATENATE STRUCTURES BY ROWS [COLFORM CHARACTER VERT]
 ∇REPL REPLACE ALL OCCURRENCES OF ELEMENT IN ARRAY
 REPLACE ALL OCCURRENCES OF ELEMENT IN ARRAY ∇REPL
 REPLACE VACANT ELEMENTS [CFORMAT CONFORM STRUCT Δ] ∇FILLS
 REPORT [COMPARE] ∇HEADERON PUTS A HEADING ON A
 REPORT FORMAT ∇BESIDE PRESENTS TWO STRUCTURES SIDE BY SIDE IN
 REPORT FORMATTING ∇ADJUSTUP EXTENDS '|' IN
 REPORT FORMATTING [IF] ∇PREPARE STANDARDIZE STRUCTURE FOR
 REPORT FORMATTING [ROWINDICES] ∇ADJUSTDOWN EXTEND THE '|' IN
 REQUIRED TAB SETTINGS TO EXISTING ONES [COLNO] ∇TABS COMPARE
 ∇RIOTA MATRIX ROW IOTA
 RJUST DL] ∇LJUST LEFT JUSTIFY ANY ARRAY [DLB
 ∇ROMAN CONVERT INTEGER TO ROMAN NUMERALS
 ROMAN NUMERALS ∇ROMAN CONVERT INTEGER TO
 ROOT ∇SQRT MULTIPRECISION INTEGER SQUARE
 ROOT ∇FSQRT MULTIPRECISION FLOATING POINT SQUARE
 ROUNDING ∇ROUNDS SELECTIVE SYMMETRICAL
 ROUNDING OF A VECTOR TO ARBITRARY SCALAR UNIT ∇DROUNDS DISTRIBUTIVE
 ∇ROUNDS SELECTIVE SYMMETRICAL ROUNDING
 ROW INDICES [AV NFORM LJNFORM] ∇GRADEUP GENERATE ASCENDING
 ROW IOTA ∇RIOTA MATRIX
 ROW OF A MATRIX [ESCAPE ESCAPEX] ∇VFORM VARIABLE FORMAT BY
 ROWFORM] ∇CCAT CATENATE BY COLUMNS [VERTAB CFORMAT CMATRIX
 ROWINDICES] ∇ADJUSTDOWN EXTEND THE '|' IN REPORT FORMATTING [
 ROWS OF CHARACTER MATRIX [USCORE] ∇ULINE UNDERLINE SPECIFIED
 ROWS TO A MATRIX VECTOR OR SCALAR ∇ADDDROWS ADD
 ROWS [COLFORM CHARACTER VERT] ∇RCAT CATENATE STRUCTURES BY
 RUNNING TIME AND NEW SPACE FOR AN APL STATEMENT [ALT] ∇TIME
 SCALAR ∇ADDDROWS ADD ROWS TO A MATRIX VECTOR OR

SCALAR	∇ADDCOLS	ADD COLUMNS TO A MATRIX VECTOR OR
SCALAR OR MULTIPRECISION NUMBER	∇ALPREC	ALTER PRECISION OF A
SCALAR UNIT∇DROUNDS		DISTRIBUTIVE ROUNDING OF A VECTOR TO ARBITRARY
SCALE]	∇FLOAT	CONVERT TO MULTIPRECISION FLOATING POINT [
SEEDIT]	∇EDIT	EDIT LATENT EXPRESSION OR CHARACTER STRUCTURE [
SELECT FIRST OR ONLY APPEARANCE IN MATRIX [FIRSTV]		∇FIRSTM
SELECT NTH WORD IN CHARACTER STRUCTURE [DTMB]		∇WORD
SELECT UNIQUE ELEMENTS FROM ANY STRUCTURE		∇DREP
SELECTIVE SYMMETRICAL ROUNDING		∇ROUNDS
SELECTIVELY ∇VARS		DISPLAY CHARACTERISTICS OR CONTENTS OF VARS
SESSION AND WORKSPACE STATUS [NOW]		∇STATUS CURRENT
SETTINGS TO EXISTING ONES [COLNO] ∇TABS		COMPARE REQUIRED TAB
∇SHAPE		SHAPE MATRIX FROM CHARACTER STRING
SHAPE MATRIX FROM CHARACTER STRING		∇SHAPE
SIDE BY SIDE IN REPORT FORMAT	∇BESIDE	PRESENTS TWO STRUCTURES
SIDE IN REPORT FORMAT	∇BESIDE	PRESENTS TWO STRUCTURES SIDE BY
SIDES OF STRUCTURE	∇NUMBLANKCOLS	COUNTS BLANK COLUMNS AT
SORTDA KFORM]	∇FREQ	FREQUENCY DISTRIBUTION OF ELEMENTS [
SPACE FOR AN APL STATEMENT [ALT]	∇TIME	RUNNING TIME AND NEW
SPECIFIC STRING FROM STRUCTURE [LIM]		∇BLANK DELETE
SPECIFIED ROWS OF CHARACTER MATRIX [USCORE]	∇ULINE	UNDERLINE
∇SQRT		MULTIPRECISION INTEGER SQUARE ROOT
SQUARE PROBABILITY FUNCTION		∇QPROBF COMPUTE CHI
SQUARE ROOT	∇SQRT	MULTIPRECISION INTEGER
SQUARE ROOT	∇FSQRT	MULTIPRECISION FLOATING POINT
STANDARD FORM	∇LISTFN	LISTS A FUNCTION IN
STANDARDIZE STRUCTURE FOR REPORT FORMATTING [IF]		∇PREPARE
STATEMENT [ALT]	∇TIME	RUNNING TIME AND NEW SPACE FOR AN APL
∇STATUS		CURRENT SESSION AND WORKSPACE STATUS [NOW]
STATUS [NOW]	∇STATUS	CURRENT SESSION AND WORKSPACE
STRING	∇SHAPE	SHAPE MATRIX FROM CHARACTER
STRING	∇FORMAT	CONVERT MULTIPRECISION NUMBER TO CHARACTER
STRING FROM STRUCTURE [LIM]		∇BLANK DELETE SPECIFIC
STRINGS FROM CHARACTER ARRAYS		∇CITED EXTRACT CITED
STRUCT Δ]	∇FILLS	REPLACE VACANT ELEMENTS [CFORMAT CONFORM
STRUCTURE		∇ONESIN LOCATE ONES IN NUMERIC
STRUCTURE	∇DREP	SELECT UNIQUE ELEMENTS FROM ANY
STRUCTURE	∇NUMBLANKCOLS	COUNTS BLANK COLUMNS AT SIDES OF
STRUCTURE FOR REPORT FORMATTING [IF]		∇PREPARE STANDARDIZE
STRUCTURE [DLTMB]	∇ERECT	ERECT WORD MATRIX FROM CHARACTER
STRUCTURE [DTMB]	∇WORD	SELECT NTH WORD IN CHARACTER

STRUCTURE [LIM]	VBANK	DELETE SPECIFIC STRING FROM
STRUCTURE [SEDIT]	VEDIT	EDIT LATENT EXPRESSION OR CHARACTER
STRUCTURED ARGUMENTS [IS]	VTLU	TABLE LOOK-UP OF
STRUCTURED DATA	VLOC	LOCATE
STRUCTURES	VON	CONFORM AND CATENATE ANY
STRUCTURES BY ROWS [COLFORM CHARACTER VERT]	VRCAT	CATENATE
STRUCTURES IN CONTROLLED FORMAT [HANG]	VTABULATE	NUMERIC
STRUCTURES SIDE BY SIDE IN REPORT FORMAT	VBESIDE	PRESENTS TWO
STRUCTURES [CENTER]	VCENTERON	CENTERS AND CATENATES TWO
VSUB	MULTIPRECISION INTEGER SUBTRACTION	
SUBTRACTION	VSUB	MULTIPRECISION INTEGER
SUBTRACTION	VFSUB	MULTIPRECISION FLOATING POINT
SUFFICIENT ENCODING POSITIONS	VENC	GENERATE
SUMMARIZE COEFFICIENTS OF COMMON CODES	VCOLLECT	COLLECT AND
SYMMETRICAL ROUNDING	VROUNDS	SELECTIVE
SYSTEM INDEPENDENT TERMINAL CONTROL CHARACTERS	VTCC	
TAB SETTINGS TO EXISTING ONES [COLNO]	VTABS	COMPARE REQUIRED
TABLE LOOK-UP OF STRUCTURED ARGUMENTS [IS]	VTLU	
VTABS	COMPARE REQUIRED TAB SETTINGS TO EXISTING ONES [COLNO]	
VTABULATE	NUMERIC STRUCTURES IN CONTROLLED FORMAT [HANG]	
VTCC	SYSTEM INDEPENDENT TERMINAL CONTROL CHARACTERS	
TERMINAL CONTROL CHARACTERS	VTCC	SYSTEM INDEPENDENT
TERMINAL ENTRY OR DEFAULT	VPROMPT	PROMPT AND CHECK
THE DATE OF EASTER	VEASTER	COMPUTE
THE ' ' IN REPORT FORMATTING [ROWINDICES]	VADJUSTDOWN	EXTEND
VTHRU	GENERATE INDICES OR OTHER EQUAL INCREMENTS BETWEEN LIMITS	
VTIME	RUNNING TIME AND NEW SPACE FOR AN APL STATEMENT [ALT]	
TIME AND NEW SPACE FOR AN APL STATEMENT [ALT]	VTIME	RUNNING
VTLU	TABLE LOOK-UP OF STRUCTURED ARGUMENTS [IS]	
VT0	NUMERIC VECTORS IN EQUAL INCREMENTS [BY IN FROM]	
TO A	VINDEX	COLUMN INDEX IN MATRIX B WHOSE MEMBERS ALL BELONG
TO A MATRIX VECTOR OR SCALAR	VADDCOLS	ADD COLUMNS
TO A MATRIX VECTOR OR SCALAR	VADDDROWS	ADD ROWS
TO ANY BASE [DIGITS CONFRAC]	VCONV	CONVERT DECIMAL VALUES
TO ARBITRARY PRECISION	VPI	COMPUTE PI
TO ARBITRARY SCALAR UNIT	VDROUNDS	DISTRIBUTIVE ROUNDING OF A VECTOR
TO CHARACTER STRING	VFORMAT	CONVERT MULTIPRECISION NUMBER
TO DECIMAL	VDEC	CONVERT
TO EXISTING ONES [COLNO]	VTABS	COMPARE REQUIRED TAB SETTINGS
TO MULTIPRECISION FLOATING POINT [SCALE]	VFLOAT	CONVERT
TO MULTIPRECISION INTEGER	VFIX	CONVERT

TO NUMERIC PATTERN	▽CHAR	BUILD CHARACTER ARRAY
TO ROMAN NUMERALS	▽ROMAN	CONVERT INTEGER
TOLERANT DIVISION [CDIV]	▽ZDIV	ZERO
TRAILING INSIGNIFICANT CHARACTERS OR VALUES	▽DT	DELETE
▽TRUNC	TRUNCATE HIGHER AND LOWER ORDER DIGITS	
TRUNCATE HIGHER AND LOWER ORDER DIGITS	▽TRUNC	
▽ULINE	UNDERLINE SPECIFIED ROWS OF CHARACTER MATRIX [USCORE]	
UNDERLINE SPECIFIED ROWS OF CHARACTER MATRIX [USCORE]	▽ULINE	
UNIQ]	▽COMB	ALL COMBINATIONS OF ELEMENTS [DEBLANK
UNIQUE ELEMENTS FROM ANY STRUCTURE	▽DREP	SELECT
UNIT▽DROUNDS	DISTRIBUTIVE ROUNDING OF A VECTOR TO ARBITRARY SCALAR	
UNSCAN]	▽DIFF	DIFFERENCES BETWEEN ADJACENT ELEMENTS [
USCORE]	▽ULINE	UNDERLINE SPECIFIED ROWS OF CHARACTER MATRIX [
VACANT ELEMENTS [CFORMAT CONFORM STRUCT Δ]	▽FILLS	REPLACE
VALUE	▽EXTEND	EXTEND VECTOR WITH LAST
VALUES	▽DT	DELETE TRAILING INSIGNIFICANT CHARACTERS OR
VALUES TO ANY BASE [DIGITS CONFRAC]	▽CONV	CONVERT DECIMAL
VARIABLE FORMAT BY ROW OF A MATRIX [ESCAPE ESCAPEX]	▽VFORM	
▽VARS	DISPLAY CHARACTERISTICS OR CONTENTS OF VARS SELECTIVELY	
VARS SELECTIVELY	▽VARS	DISPLAY CHARACTERISTICS OR CONTENTS OF
VECTOR	▽CVEC	BUILD COMPRESSION OR LOGICAL
VECTOR	▽XVEC	EXPAND LOGICAL
VECTOR OR SCALAR	▽ADDCOLS	ADD COLUMNS TO A MATRIX
VECTOR OR SCALAR	▽ADDDROWS	ADD ROWS TO A MATRIX
VECTOR TO ARBITRARY SCALAR UNIT▽DROUNDS	DISTRIBUTIVE ROUNDING OF A	
VECTOR WITH LAST VALUE	▽EXTEND	EXTEND
VECTORS IN EQUAL INCREMENTS [BY IN FROM]	▽TO	NUMERIC
VERT]	▽RCAT	CATENATE STRUCTURES BY ROWS [COLFORM CHARACTER
VERTAB CFORMAT CMATRIX ROWFORM]	▽CCAT	CATENATE BY COLUMNS [
▽VFORM	VARIABLE FORMAT BY ROW OF A MATRIX [ESCAPE ESCAPEX]	
V2M]	▽M2V	COMPRESS CHARACTER MATRIX EXPAND RESULT [
WHOSE MEMBERS ALL BELONG TO A	▽INDEX	COLUMN INDEX IN MATRIX B
▽WIDTH	MEASURE FORMATTED MATRIX	
WITH BLANKS OR ZEROS	▽PAD	PADS ARRAYS
WITH LAST VALUE	▽EXTEND	EXTEND VECTOR
▽WORD	SELECT NTH WORD IN CHARACTER STRUCTURE [DTMB]	
WORD IN CHARACTER STRUCTURE [DTMB]	▽WORD	SELECT NTH
WORD MATRIX FROM CHARACTER STRUCTURE [DLTM]	▽VERECT	ERECT
WORKSPACE STATUS [NOW]	▽STATUS	CURRENT SESSION AND
▽XVEC	EXPAND LOGICAL VECTOR	
▽ZDIV	ZERO TOLERANT DIVISION [CDIV]	
ZERO TOLERANT DIVISION [CDIV]	▽ZDIV	
ZEROS	▽PAD	PADS ARRAYS WITH BLANKS OR
Δ]	▽FILLS	REPLACE VACANT ELEMENTS [CFORMAT CONFORM STRUCT

Primary Function Names vs. Abstract Sorted by Abstract

ADDCOLS	ADD COLUMNS TO A MATRIX VECTOR OR SCALAR
ADDROWS	ADD ROWS TO A MATRIX VECTOR OR SCALAR
COMB	ALL COMBINATIONS OF ELEMENTS [DEBLANK UNIQ]
ALPREC	ALTER PRECISION OF A SCALAR OR MULTIPRECISION NUMBER
CHAR	BUILD CHARACTER ARRAY TO NUMERIC PATTERN
CVEC	BUILD COMPRESSION OR LOGICAL VECTOR
CCAT	CATENATE BY COLUMNS [VERTAB CFORMAT CMATRIX ROWFORM]
RCAT	CATENATE STRUCTURES BY ROWS [COLFORM CHARACTER VERT]
OUTPUT	CENTER HEADINGS OVER FORMATTED COLUMNS [CNTR DMZ NEXTA]
CENTERON	CENTERS AND CATENATES TWO STRUCTURES [CENTER]
FRAMETEST	CHECKS A MATRIX FOR FRAMING
COLLECT	COLLECT AND SUMMARIZE COEFFICIENTS OF COMMON CODES
INDEX	COLUMN INDEX IN MATRIX B WHOSE MEMBERS ALL BELONG TO A
TABS	COMPARE REQUIRED TAB SETTINGS TO EXISTING ONES [COLNO]
M2V	COMPRESS CHARACTER MATRIX EXPAND RESULT [V2M]
QPROBF	COMPUTE CHI SQUARE PROBABILITY FUNCTION
DATE	COMPUTE NORMAL DATE FROM ASTRONOMERS' DAY NUMBER
PI	COMPUTE PI TO ARBITRARY PRECISION
EASTER	COMPUTE THE DATE OF EASTER
ON	CONFORM AND CATENATE ANY STRUCTURES
CONV	CONVERT DECIMAL VALUES TO ANY BASE [DIGITS CONFRAC]
ROMAN	CONVERT INTEGER TO ROMAN NUMERALS
FORMAT	CONVERT MULTIPRECISION NUMBER TO CHARACTER STRING
DEC	CONVERT TO DECIMAL
FIX	CONVERT TO MULTIPRECISION INTEGER
FLOAT	CONVERT TO MULTIPRECISION FLOATING POINT [SCALE]
NUMBLANKCOLS	COUNTS BLANK COLUMNS AT SIDES OF STRUCTURE
STATUS	CURRENT SESSION AND WORKSPACE STATUS [NOW]
DAYS	DATE CALCULATIONS [DATES NDATES PAYDAY]
DAYNO	DAY NUMBER FOR ASTRONOMERS [MOONPHASE]
BLANK	DELETE SPECIFIC STRING FROM STRUCTURE [LIM]
DT	DELETE TRAILING INSIGNIFICANT CHARACTERS OR VALUES
DIFF	DIFFERENCES BETWEEN ADJACENT ELEMENTS [UNSCAN]
VARS	DISPLAY CHARACTERISTICS OR CONTENTS OF VARS SELECTIVELY
DROUNDS	DISTRIBUTIVE ROUNDING OF A VECTOR TO ARBITRARY SCALAR UNIT
EDIT	EDIT LATENT EXPRESSION OR CHARACTER STRUCTURE [SEDIT]
MEDIT	EDIT MATRIX
CAN	EDIT MULTIPRECISION INTEGERS INTO CANONICAL FORMAT
ERECT	ERECT WORD MATRIX FROM CHARACTER STRUCTURE [DLTM]
XVEC	EXPAND LOGICAL VECTOR
ADJUSTDOWN	EXTEND THE ' ' IN REPORT FORMATTING [ROWINDICES]
EXTEND	EXTEND VECTOR WITH LAST VALUE
ADJUSTUP	EXTENDS ' ' IN REPORT FORMATTING
CITED	EXTRACT CITED STRINGS FROM CHARACTER ARRAYS
FRAME	FRAME AN ARRAY [MATRIX CHARACTER]
FREQ	FREQUENCY DISTRIBUTION OF ELEMENTS [SORTDA KFORM]
GRADEUP	GENERATE ASCENDING ROW INDICES [AV NFORM LJNFORM]
THRU	GENERATE INDICES OR OTHER EQUAL INCREMENTS BETWEEN LIMITS
ENC	GENERATE SUFFICIENT ENCODING POSITIONS
LJUST	LEFT JUSTIFY ANY ARRAY [DLB RJUST DL]
LISTFN	LISTS A FUNCTION IN STANDARD FORM
ONESIN	LOCATE ONES IN NUMERIC STRUCTURE
LOC	LOCATE STRUCTURED DATA
RIOTA	MATRIX ROW IOTA
WIDTH	MEASURE FORMATTED MATRIX

LOGICAL	MISCELLANEOUS [INTEGER FLOATING EMPTY CHARACTER]
AMORTIZE	MORTGAGE CALCULATION BY MONTHS
ADD	MULTIPRECISION INTEGER ADDITION
DIV	MULTIPRECISION INTEGER DIVISION
FADD	MULTIPRECISION FLOATING POINT ADDITION
FDIV	MULTIPRECISION FLOATING POINT DIVISION
FEXP	MULTIPRECISION FLOATING POINT EXPONENTIAL FUNCTION
FMUL	MULTIPRECISION FLOATING POINT MULTIPLICATION
FSQRT	MULTIPRECISION FLOATING POINT SQUARE ROOT
FSUB	MULTIPRECISION FLOATING POINT SUBTRACTION
MUL	MULTIPRECISION INTEGER MULTIPLICATION
SQRT	MULTIPRECISION INTEGER SQUARE ROOT
SUB	MULTIPRECISION INTEGER SUBTRACTION
TABULATE	NUMERIC STRUCTURES IN CONTROLLED FORMAT [HANG]
TO	NUMERIC VECTORS IN EQUAL INCREMENTS [BY IN FROM]
PAD	PADS ARRAYS WITH BLANKS OR ZEROS
BARGRAPH	PLOT HORIZONTAL INTEGER BARGRAPHS
PREEDIT	PREPARE MATRIX FOR FUNCTION-LIKE EDITING [POSTEDIT]
BESIDE	PRESENTS TWO STRUCTURES SIDE BY SIDE IN REPORT FORMAT
PROMPT	PROMPT AND CHECK TERMINAL ENTRY OR DEFAULT
HEADERON	PUTS A HEADING ON A REPORT [COMPARE]
REPL	REPLACE ALL OCCURRENCES OF ELEMENT IN ARRAY
FILLS	REPLACE VACANT ELEMENTS [CFORMAT CONFORM STRUCT Δ]
TIME	RUNNING TIME AND NEW SPACE FOR AN APL STATEMENT [ALT]
FIRSTM	SELECT FIRST OR ONLY APPEARANCE IN MATRIX [FIRSTV]
WORD	SELECT NTH WORD IN CHARACTER STRUCTURE [DTMB]
DREP	SELECT UNIQUE ELEMENTS FROM ANY STRUCTURE
ROUNDS	SELECTIVE SYMMETRICAL ROUNDING
SHAPE	SHAPE MATRIX FROM CHARACTER STRING
PREPARE	STANDARDIZE STRUCTURE FOR REPORT FORMATTING [IF]
TCC	SYSTEM INDEPENDENT TERMINAL CONTROL CHARACTERS
TLU	TABLE LOOK-UP OF STRUCTURED ARGUMENTS [IS]
TRUNC	TRUNCATE HIGHER AND LOWER ORDER DIGITS
ULINE	UNDERLINE SPECIFIED ROWS OF CHARACTER MATRIX [USCORE]
VFORM	VARIABLE FORMAT BY ROW OF A MATRIX [ESCAPE ESCAPEX]
ZDIV	ZERO TOLERANT DIVISION [CDIV]

Primary Function Names vs. Abstract Sorted by Function Name

ADD	MULTIPRECISION INTEGER ADDITION
ADDCOLS	ADD COLUMNS TO A MATRIX VECTOR OR SCALAR
ADDDROWS	ADD ROWS TO A MATRIX VECTOR OR SCALAR
ADJUSTDOWN	EXTEND THE ' ' IN REPORT FORMATTING [ROWINDICES]
ADJUSTUP	EXTENDS ' ' IN REPORT FORMATTING
ALPREC	ALTER PRECISION OF A SCALAR OR MULTIPRECISION NUMBER
AMORTIZE	MORTGAGE CALCULATION BY MONTHS
BARGRAPH	PLOT HORIZONTAL INTEGER BARGRAPHS
BESIDE	PRESENTS TWO STRUCTURES SIDE BY SIDE IN REPORT FORMAT
BLANK	DELETE SPECIFIC STRING FROM STRUCTURE [LIM]
CAN	EDIT MULTIPRECISION INTEGERS INTO CANONICAL FORMAT
CCAT	CATENATE BY COLUMNS [VERTAB CFORMAT CMATRIX ROWFORM]
CENTERON	CENTERS AND CATENATES TWO STRUCTURES [CENTER]
CHAR	BUILD CHARACTER ARRAY TO NUMERIC PATTERN
CITED	EXTRACT CITED STRINGS FROM CHARACTER ARRAYS
COLLECT	COLLECT AND SUMMARIZE COEFFICIENTS OF COMMON CODES
COMB	ALL COMBINATIONS OF ELEMENTS [DEBLANK UNIQ]
CONV	CONVERT DECIMAL VALUES TO ANY BASE [DIGITS CONFRAC]
CVEC	BUILD COMPRESSION OR LOGICAL VECTOR
DATE	COMPUTE NORMAL DATE FROM ASTRONOMERS' DAY NUMBER
DAYNO	DAY NUMBER FOR ASTRONOMERS [MOONPHASE]
DAYS	DATE CALCULATIONS [DATES NDATES PAYDAY]
DEC	CONVERT TO DECIMAL
DIFF	DIFFERENCES BETWEEN ADJACENT ELEMENTS [UNSCAN]
DIV	MULTIPRECISION INTEGER DIVISION
DREP	SELECT UNIQUE ELEMENTS FROM ANY STRUCTURE
DROUNDS	DISTRIBUTIVE ROUNDING OF A VECTOR TO ARBITRARY SCALAR UNIT
DT	DELETE TRAILING INSIGNIFICANT CHARACTERS OR VALUES
EASTER	COMPUTE THE DATE OF EASTER
EDIT	EDIT LATENT EXPRESSION OR CHARACTER STRUCTURE [SEDIT]
ENC	GENERATE SUFFICIENT ENCODING POSITIONS
ERECT	ERECT WORD MATRIX FROM CHARACTER STRUCTURE [DLTMB]
EXTEND	EXTEND VECTOR WITH LAST VALUE
FADD	MULTIPRECISION FLOATING POINT ADDITION
FDIV	MULTIPRECISION FLOATING POINT DIVISION
FEXP	MULTIPRECISION FLOATING POINT EXPONENTIAL FUNCTION
FILLS	REPLACE VACANT ELEMENTS [CFORMAT CONFORM STRUCT Δ]
FIRSTM	SELECT FIRST OR ONLY APPEARANCE IN MATRIX [FIRSTV]
FIX	CONVERT TO MULTIPRECISION INTEGER
FLOAT	CONVERT TO MULTIPRECISION FLOATING POINT [SCALE]
FMUL	MULTIPRECISION FLOATING POINT MULTIPLICATION
FORMAT	CONVERT MULTIPRECISION NUMBER TO CHARACTER STRING
FRAME	FRAME AN ARRAY [MATRIX CHARACTER]
FRAMETEST	CHECKS A MATRIX FOR FRAMING
FREQ	FREQUENCY DISTRIBUTION OF ELEMENTS [SORTDA KFORM]
FSQRT	MULTIPRECISION FLOATING POINT SQUARE ROOT
FSUB	MULTIPRECISION FLOATING POINT SUBTRACTION
GRADEUP	GENERATE ASCENDING ROW INDICES [AV NFORM LJNFORM]
HEADERON	PUTS A HEADING ON A REPORT [COMPARE]
INDEX	COLUMN INDEX IN MATRIX B WHOSE MEMBERS ALL BELONG TO A
LISTFN	LISTS A FUNCTION IN STANDARD FORM
LJUST	LEFT JUSTIFY ANY ARRAY [DLB RJUST DL]
LOC	LOCATE STRUCTURED DATA
LOGICAL	MISCELLANEOUS [INTEGER FLOATING EMPTY CHARACTER]
MEDIT	EDIT MATRIX

MUL	MULTIPRECISION INTEGER MULTIPLICATION
M2V	COMPRESS CHARACTER MATRIX EXPAND RESULT [V2M]
NUMBLANKCOLS	COUNTS BLANK COLUMNS AT SIDES OF STRUCTURE
ON	CONFORM AND CATENATE ANY STRUCTURES
ONESIN	LOCATE ONES IN NUMERIC STRUCTURE
OUTPUT	CENTER HEADINGS OVER FORMATTED COLUMNS [CNTR DMZ NEXTA]
PAD	PADS ARRAYS WITH BLANKS OR ZEROS
PI	COMPUTE PI TO ARBITRARY PRECISION
PREEDIT	PREPARE MATRIX FOR FUNCTION-LIKE EDITING [POSTEDIT]
PREPARE	STANDARDIZE STRUCTURE FOR REPORT FORMATTING [IF]
PROMPT	PROMPT AND CHECK TERMINAL ENTRY OR DEFAULT
QPROBF	COMPUTE CHI SQUARE PROBABILITY FUNCTION
RCAT	CATENATE STRUCTURES BY ROWS [COLFORM CHARACTER VERT]
REPL	REPLACE ALL OCCURRENCES OF ELEMENT IN ARRAY
RIOTA	MATRIX ROW IOTA
ROMAN	CONVERT INTEGER TO ROMAN NUMERALS
ROUNDS	SELECTIVE SYMMETRICAL ROUNDING
SHAPE	SHAPE MATRIX FROM CHARACTER STRING
SQRT	MULTIPRECISION INTEGER SQUARE ROOT
STATUS	CURRENT SESSION AND WORKSPACE STATUS [NOW]
SUB	MULTIPRECISION INTEGER SUBTRACTION
TABS	COMPARE REQUIRED TAB SETTINGS TO EXISTING ONES [COLNO]
TABULATE	NUMERIC STRUCTURES IN CONTROLLED FORMAT [HANG]
TCC	SYSTEM INDEPENDENT TERMINAL CONTROL CHARACTERS
THRU	GENERATE INDICES OR OTHER EQUAL INCREMENTS BETWEEN LIMITS
TIME	RUNNING TIME AND NEW SPACE FOR AN APL STATEMENT [ALT]
TLU	TABLE LOOK-UP OF STRUCTURED ARGUMENTS [IS]
TO	NUMERIC VECTORS IN EQUAL INCREMENTS [BY IN FROM]
TRUNC	TRUNCATE HIGHER AND LOWER ORDER DIGITS
ULINE	UNDERLINE SPECIFIED ROWS OF CHARACTER MATRIX [USCORE]
VARS	DISPLAY CHARACTERISTICS OR CONTENTS OF VARS SELECTIVELY
VFORM	VARIABLE FORMAT BY ROW OF A MATRIX [ESCAPE ESCAPEX]
WIDTH	MEASURE FORMATTED MATRIX
WORD	SELECT NTH WORD IN CHARACTER STRUCTURE [DTMB]
XVEC	EXPAND LOGICAL VECTOR
ZDIV	ZERO TOLERANT DIVISION [CDIV]

INDEX

Function		Page
ADD	47
ADDCOLS	1
ADDROWS	2
ADJUSTDOWN	24
ADJUSTUP	25
ALPREC	48
ALT	45
AMORTIZE	63
AV	11
BARGRAPH	26
BESIDE	27
BLANK	28
BY	76
CAN	49
CCAT	3
CDIV	78
CENTER	29
CENTERON	29
CFORMAT	84
CHAR	4
CHARACTER	9
CITED	30
CMATRIX	3
CNTR	36
COLFORM	16
COLLECT	31
COLNO	43
COMB	79
COMPARE	33
CONFORM	84
CONFRAC	64
CONV	64
CVEC	80
DATE	65
DATES	67
DAYNO	66
DAYS	67
DEBLANK	79
DEC	68
DIFF	5
DIV	50
DL	34
DLB	34
DLTMB	7
DMZ	36
DREP	32
DROUNDS	69
DT	81
DTMB	23
EASTER	82

INDEX (cont.)

Function	Page
<i>FDIT</i>	6
<i>EMPTY</i>	86
<i>ENC</i>	70
<i>FRECT</i>	7
<i>ESCAPE</i>	21
<i>ESCAPEX</i>	21
<i>EXTEND</i>	83
<i>FADD</i>	51
<i>FDIV</i>	52
<i>FEXP</i>	53
<i>FILLS</i>	84
<i>FIRSTM</i>	8
<i>FIRSTV</i>	8
<i>FIX</i>	54
<i>FLOAT</i>	55
<i>FLOATING</i>	86
<i>FMUL</i>	56
<i>FORMAT</i>	57
<i>FRAME</i>	9
<i>FRAMETEST</i>	10
<i>FREQ</i>	71
<i>FROM</i>	76
<i>FSQRT</i>	58
<i>FSUB</i>	59
<i>GRADEUP</i>	11
<i>HANG</i>	39
<i>HEADERON</i>	33
<i>IF</i>	38
<i>IN</i>	76
<i>INDEX</i>	12
<i>INTEGER</i>	86
<i>IS</i>	90
<i>KFORM</i>	71
<i>LIM</i>	28
<i>LISTFN</i>	40
<i>LJNFORM</i>	11
<i>LJUST</i>	34
<i>LOC</i>	85
<i>LOGICAL</i>	86
<i>MATRIX</i>	9
<i>MEDIT</i>	13
<i>MOONPHASE</i>	66
<i>MUL</i>	60
<i>M2V</i>	14
<i>NDATES</i>	67
<i>NEXTA</i>	36
<i>NFORM</i>	11
<i>NOW</i>	42
<i>NUMBLANKCOLS</i>	87
<i>ON</i>	35

INDEX (cont.)

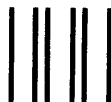
Function	Page
ONESIN	88
OUTPUT	36
PAD	37
PAYDAY	67
PI	72
POSTEDIT	15
PREEDIT	15
PREPARE	38
PROMPT	41
QPROBF	73
RCAT	16
REPL	17
RIOTA	18
RJUST	34
ROMAN	74
ROUNDS	75
ROWFORM	3
ROWINDICES	24
SCALE	55
SEDIT	6
SHAPE	19
SORTDA	71
SQRT	61
STATUS	42
STRUCT	84
SUB	62
TABS	43
TABULATE	39
TCC	44
THRU	89
TIME	45
TLU	90
TO	76
TRUNC	77
ULINE	20
UNIQ	79
UNSCAN	5
USCORE	20
VARS	46
VERT	16
VERTAB	3
VFORM	21
V2M	14
WIDTH	22
WORD	23
XVEC	91
ZDIV	78
Δ	84
<u>DIGITS</u>	64

Reader's Comment Form

Fold and tape

Please Do Not Staple

Fold and tape



BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 40

ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 824
1133 Westchester Avenue
White Plains, New York 10604

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



Fold and tape

Please Do Not Staple

Fold and tape

Cut or Fold Along Line

The APL Handbook of Techniques Printed in U.S.A. S320-5996-0



S320-5996-0

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed.*

Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape

Please Do Not Staple

Fold and tape



BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 40

ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 824
1133 Westchester Avenue
White Plains, New York 10604

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



Fold and tape

Please Do Not Staple

Fold and tape

The APL Handbook of Techniques Printed in U.S.A. S320-5996-0



S320-5996-0

The APL Handbook of Techniques Printed in U.S.A. S320-5996-0

IBM