A-14-A

GENERAL ELECTRIC

DIAL COMM	8-264-4207	DATE	MARCH 10, 1970	COPIES
DEPT.	CISL/LSD			
ADDRESS	575 TECHNOLOGY SQ., CAMBRIDGE, MASS. 02139			
SUBJECT	PROPOSED MULTICS APL			

TO: C. T. CLINGEN
F. J. CORBATÓ
R. C. DALEY
R. A. FREIBURGHOUSE
J. W. GINTELL
J. M. GROCHOW
J. H. SALTZER ✓
T. H. VAN VLECK

FROM: J. D. MILLS
M. G. SMITH

The attached document describes an implementation of the APL programming language proposed for Multics. It is sent to you for your information and review. Please direct your criticisms and suggestions, preferably in writing, to J. D. Mills or M. G. Smith by Wednesday, March 18. Shortly thereafter a revised proposal will be distributed, incorporating any adopted alterations.

James D. Mills / Maxim G. Smith

JAMES D. MILLS/MAXIM G. SMITH

/1

(enclosures)

PROPOSED MULTICS APL

A project has just started which will lead to an implementation of APL in Multics. APL is a programming language, rich in operators and matrix operations, originally developed by K. E. Iverson. The name comes from the initial letters of the title of his book, A Programming Language (New York: Wiley, 1962), which documents an early, pedagogical version of the language.

An implementation of APL, called APL\360, has become fairly popular at MIT and elsewhere. It is an interactive interpreter running on remote-access IBM 360 computers.

Multics APL is planned to be a nearly exact duplicate of the programming language portion of APL\360. Also, most of the APL\360 system commands and editing functions will be present in Multics APL, though some will look different to the user. In addition, several new functions unique to the Multics implementation will be added, to allow APL users to take advantage of some of the power of Multics.

This document attempts to describe briefly what capabilities Multics APL will provide. Inasmuch as our implementation is to be so similar to APL\360, the most convenient way to specify Multics APL is to list the differences between it and APL\360. Hence, the reader is presumed to be familiar with APL\360. A good introduction to APL\360 can be found in APL\360 Primer, IBM form number GH20-0689. This primer is organized as follows:

Chapter 1 doesn't say anything;
Chapter 2 discusses dialing up and getting connected to APL\360;
Chapters 3-5, 8-13, and 16-25 describe the programming language itself;
Chapters 7 and 8 explain the editing provided to enter and correct programs; and
Chapters 14, 15, and 26 discuss control and maintenance functions.

A simple console session with APL\360 is included with this document as Appendix A. This should give some of the flavor of the language.

1. The Programming Language

As stated above, Multics APL will retain the programming language portion of APL\360 practically without change. Referencing the primer cited above, this implies that chapters 3-5, 8-13, and 16-25 all will be accurately implemented in Multics APL with only these two changes, dictated by the floating-point hardware of the GE 645:

a. The largest and smallest numbers which can be handled by Multics APL will be approximately 1e38 and 1e-38, respectively, instead of 1e75 and 1e-75 for APL\360.

b. Multics APL will retain approximately 19 decimal digits of precision, as opposed to 16 for APL\360.

2. Editing

The editing of APL\360 cannot be duplicated exactly under Multics as it is today. This is because APL\360 editing relies on the use of the quit signal to edit a line, while Multics discards any partial input line when quit is detected. In addition, it is felt that APL\360 editing is cumbersome at best, so that something better is in order anyway. Hence, Multics APL will provide two editing modes, Multics mode and APL\360 mode. APL\360 mode will be as much like APL\360 editing as is possible without using the quit signal. It is provided to make it easier for persons familiar with APL\360 to change over to Multics APL with a minimum of hurdles. Multics editing mode will be practically identical to EDM. Most users will no doubt switch to this mode as soon as they discover how superior it is to APL\360 mode.

this is a bug in Multics not something to design around

Chapters 7 and 8 of the primer describe the APL\360 mode of editing in Multics APL with the following changes:

a. The quit signal (referred to as the attention button in the primer, as it is oriented toward IBM terminals) will not be used for line editing. Instead, erase, kill, and escape characters will be defined for the APL character set which will function like the corresponding Multics characters.

b. Deletion of lines from stored files will not use the quit signal either. Lines will be deleted by replacing them with null lines (null lines are illegal in APL).

The Multics mode of editing will be added. It does not exist at all in APL\360.

c. There will be a system command, ")EDIT", and a built-in function, "⌶EDIT", to select the editing mode. Initially upon invoking APL, the system will be in APL\360 mode. To change to Multics mode, the user will issue the system command ")EDIT MULTICS" where "MULTICS" is any word beginning with "M" or execute the built-in function "⌶EDIT 'MULTICS'", where "'MULTICS'" is any expression having the value of a character array with the initial letter "M". The mode can be changed back via ")EDIT APL" or "⌶EDIT 'APL'", where again only the "A" is significant. The system will respond "WAS MULTICS" or "WAS APL" to the system command, and will return the character vector "'MULTICS'" or "'APL'" as the value of the built-in function.

d. Definition mode is entered and left with the nabla character, whether APL\360 or Multics editing is in effect. When definition mode is entered with Multics editing, the system will respond "INPUT" or "EDIT", accordingly as the program mentioned by the user does not or does exist. The user may switch from input to

d. (Continued)

edit and vice-versa by typing a line consisting of only a period. A nabla typed anywhere except between quotes terminates definition mode; i.e., exits from the editor; whether typed during input or edit. The program itself, rather than a temporary copy of it, will be edited as requests are typed, so there is no need to "write" it out, as in EDM.

e. The editing requests will be those of EDM, except that there will be no quit request and the write requests will always require a segment name. An APL source program will not be stored as a separate segment by the APL interpreter, so there is no default segment name which could be applied on write requests. Note that the APL program itself is edited as requests are issued, so there is no need to issue write requests unless one desires a copy of his program as an independent segment.

f. In Multics editing mode, there will be no hypothetical null line in front of the program. The first line of the program will be its header line, which can be edited as any other line. It is considered to have line number zero. Successive lines have numbers one, two, ... unless some editing has been done, in which case there may be missing or fractional line numbers.

g. When the nabla is typed to exit from definition mode, the header line will be checked for validity, all null lines will be removed, and the lines will be renumbered sequentially by one's.
Note: the line numbers are not apparent when editing unless the "=" is used.

3. Character Set

The APL character set (see Appendix B) is far removed from the Multics standard. In fact, it contains more than 128 distinct characters so there is no hope of establishing a one-to-one mapping onto the Multics set. Hence, Multics APL will have its own eight-bit-in-a-nine-bit-field character set. This set will be identical to the Multics set where the graphics overlap or can by some stretch of the imagination be made to correspond (which will be about 87 cases), but will be arbitrary elsewhere.

Another consequence of this is that it will be convenient to use Multics APL only from a terminal equipped with APL graphics. Terminals with conventional Multics graphics will be usable in an emergency, but the user will have to pay the price of occasional escapes or far-fetched correspondences. The only terminals which can presently be equipped with APL graphics are IBM 1050's, 2740's and 2741's. However, rumors are about that General Electric may supply an APL belt for the Terminet 300, and that Teletype may supply APL pallets for the model 37. The pallet-box of a model 37 Teletype is fairly easy to change--a less than one minute operation; the change of belt in a Terminet is definitely not a do-it-yourself job.

Upper/lower case is a nuisance

The converse operation, using Multics from a terminal equipped with APL graphics, will not be difficult. Multics is less demanding of its character set, and almost all Multics activity occurs within the 87-member common subset of characters.

Consequently, users will observe the following points:

a. Multics APL will install its own DIM (or else a table to drive the standard DIM if it is flexible enough) when invoked. The DIM for IBM terminals will assume that the user has an APL typing element; the DIM for other terminals will assume that they have Multics graphics. IBM terminal users can use Multics APL as if it

a. (Continued)

were APL\360. Other terminal users will necessarily resort to escape sequences more or less often.

b. There are occasions when, within APL (i.e., through the APL DIM), one would like to type lines to Multics (example: the execute request of EDM). To do this, the user will have to be aware of the correspondence between APL characters and Multics characters. As noted above, this will not be difficult. The 26 unshifted alphabetic characters of APL will be mapped by the DIM into the Multics lower-case alphabetics. The 26 alphabetics underscored of APL will be mapped into the upper-case alphabetics. The correspondences for the numerics and the 23 special characters common to APL and Multics are obvious. Other arbitrary correspondences will be adopted for the remaining 10 Multics graphics.

c. Inasmuch as the internal codes used to represent most APL characters within Multics will correspond to the Multics standard codes, the collating sequence of characters in Multics APL will differ from that of APL\360.

4. Entering and Leaving APL

APL\360 is a stand-alone system, but Multics APL will be implemented as a command (hence also as a subroutine) on Multics. Therefore, Chapter 2 of the APL\360 primer does not apply to Multics APL.

a. Multics APL will be entered by issuing the command "apl". Possible arguments to the command may include options for obtaining input lines from a segment instead of the console, writing output to a segment instead of the console, and loading a particular workspace initially.

4. (Continued)

b. A workspace will be a segment. Any place that the APL syntax requires a workspace name, the Multics APL user will be able to specify a path-name. Normal Multics access and search rules will be applicable. The Multics file-system will take the place of APL\360 "libraries".

c. When APL attains control, it will type "APL" and indent six characters (unless either input or output is from or to a segment), and a clear workspace will be in effect (unless the initial workspace argument was specified), in execution mode.

d. APL will establish its own quit handler as soon as it is invoked (this is so that it can give high-level response and debugging aid to looping programs, and so that it can properly reset the DIM before exiting back to Multics). Any time the user presses the quit button, APL will regain control, type "APL" on the console, and read the console for input. This implies that, under APL, the quit button cannot be used to return to Multics command level. The only way to cause APL to return is to issue a quit system request, ")Q" or ")QQ", or to execute a quit built-in function, "IQ" or "IQQ".

If APL has been invoked recursively so that several instances of the interpreter are in execution, the quit signal will be accepted by the most recent invocation. To return control to a previous invocation (and hence discard the suspended states of more recent invocations in the process), the user may issue the quit request ")Q" as many times as necessary. The request ")QQ" will return directly to Multics command level, across any number of invocations of APL. The built-in functions "IQ" and "IQQ" behave correspondingly.

e. The APL interpreter will be callable recursively from within APL in a number of ways. One way is via the "IAPL" built-in function. Execution of "z<-IAPL x", where "x" is any expression having the value of a character array, will cause APL to interpret the input lines read from "x" (in row-major order, NL characters must be in "x" in the proper places, rank and dimensions are ignored), and place the output in the character vector "z".

5. System Commands

Control and maintenance requests issued to the APL\360 system are called "system commands" in the primer. Chapters 14, 15, and 26 of the primer will apply to Multics APL as amended by these points:

a. The error messages which Multics APL will emit are completely undesigned. It is unknown how much they will or will not resemble APL\360 error messages.

b. Entering and leaving APL will be done as discussed in section 4, above.

c. Saved workspaces will be segments in the user's working directory. The ")LIB" system command will be implemented with Multics "list". "Libraries" will not exist in Multics APL; standard Multics path-names, access rules, and search rules will apply to accessing workspaces.

d. The system commands to communicate with the computer operator will not be implemented.

what is it? e. The notion of a "protected" function will not be available in Multics APL.

5. (Continued)

There may need to be an option to prevent this for limited users.

f. The system command ")E x" will be added. The text "x" will be passed to Multics as a command line to be executed.
A corresponding built-in function "⌶E x" will accept any expression "x" having the value of a character array.

g. The "⌶APL" built-in function will be provided as discussed in point 4.e, above. Note that the APL interpreter can also be called recursively using: the ")E" system command; the "⌶E" built-in function; the "E̲" request when in Multics definition mode; or when called by any program entered by any of the above ways (including the shell's command-level entry!).

h. The "⌶RSEG" and "⌶WSEG" built-in functions will be added to permit APL programs to read and write segments. Execution of "z <- ⌶RSEG x" where the value of "x" is a path-name, causes the entire segment of that name to become the value of the character vector "z", successive characters of the segment being assigned to successive elements of "z". If the segment "x" cannot be found, a diagnostic occurs. In addition, the "⌶RSEG" function can accept a left operand: execution of "z <- y ⌶RSEG x", where the value of "y" is a vector of integers, causes the line numbers mentioned in "y" to be read from segment "x" into "z". This assumes that NL characters will be found in "x"; if none are, "x" consists of only one line. Any lines not found contribute no input to "z".

Note the difference in operation of "⌶RSEG" with no left operand as opposed to a null left operand: "z <- ⌶RSEG x" reads the entire segment; "z <- ι0 ⌶RSEG x" reads nothing ("z" will be null).

h. (Continued)

Execution of "x ⌶WSEG y", where the values of "x" and "y" are character arrays, writes the characters in "y" out as a segment named "x". The letters "R", "E", "W" and "A" in any combination may follow the name in "x" after a blank, and the segment will be given that mode after creation (otherwise, RWA mode will be assigned).

Note: Another way to read and write segments from within APL is to use the "⊥", "W", and "W̲" editing requests.

i. The "⌶19" built-in function of APL\360 (cumulative keyboard-unlocked time) will not be implemented, as there is no way to obtain this information under Multics.

I don't believe this statement.

MGS/1
3/9/70

Appendix A

SAMPLE TERMINAL SESSION

```
)1776
010)  19.32.36 07/03/68  JANET

   A P L \ 3 6 0
                                        FUNDAMENTALS

      3×4                               Entry automatically indented
12                                      Response not indented
      X←3×4                             X is assigned value of
                                             the expression
      X
12                                      Value of X typed out
      Y←¯5                              Negative sign for negative
                                             constants
      X+Y
7
      144E¯2                            Exponential form of constant
1.44
      P←1  2  3  4                      Four-element vector
      P×P                               Functions apply element by element
1   4   9   16
      P×Y                               Scalar applies to all elements
¯5  ¯10  ¯15  ¯20
      Q←'CATS'                          Character constant (4-element
      Q                                      vector)
CATS
      YZ←5                              Multi-character names
      YZ̲1←5
      YZ+YZ̲1
10
      3+4×5+6                           Correction by backspace
         v                                  and linefeed
         +5+6
18
      X←3
      Y←4
      (X×Y)+4
16
      X×Y+4                             Executed from right to left
24
```

```
      X Y                          Entry of invalid expression
SYNTAX ERROR                       Shows type of error committed
      X Y                          Retypes invalid statement with
      ^                                caret where execution stopped
      XY                           Multi-character name (not X×Y)
VALUE ERROR
      XY                           XY had not been assigned a value
      ^
                                   SCALAR FUNCTIONS

      4×3⌈5.1                      Dyadic maximum
20.4
      (4×3)⌈5.1
12
      4×⌈5.1                       Monadic ceiling
24
      X←⍳5                         Index generator function
      X
1  2  3  4  5
      ⍳0                           Empty vector
                                       prints as a blank line
      Y←5-X                        All scalar functions extend
      Y                                to vectors
4  3  2  1  0
      X⌈Y
4  3  3  4  5
      X≤Y                          Relations produce
1  1  0  0  0                          logical (0 1) results
      ○1                           Pi×1
3.141592654
      ○÷1 2                        Pi÷1 2
3.141592654  1.570796327
      X←45 90
      ○X÷180                       Conversion of X to radians
0.7853981634  1.570796327
      1○1                          Sin 1
0.8418709848
      2○1 2                        Cos 1 2
0.5403023059  ¯0.4161468365
      3○1                          Tan 1
1.557407725
      ¯3○1                         Arctan 1
0.7853981634
      3○¯3○⍳7                      Tan Arctan 1 2 3 4 5 6 7
1  2  3  4  5  6  7
      Y←1 2
      4○Y                          (1+Y*2)*.5
1.414213562  2.236067977
      0○÷Y                         (1-÷Y*2)*.5
0  0.8660254038
      7○1 2                        Tanh 1 2
0.761594156  0.9640275801
      ¯7○7○1 2                     Arctanh Tanh 1 2
1  2
```

DEFINED FUNCTIONS

```
          ∇Z←X F Y                     Header (2 args and result)
[1]       Z←((X*2)+Y*2)*.5             Function body
[2]       ∇                            Close of definition
          3 F 4                        Execution of dyadic function F
5
          P←7
          Q←(P+1)F P-1                 Use of F with expressions
          Q                                 as arguments
10
          4×3 F 4
20
          ∇B←G A                       G is the signum function
[1]       B←(A>0)-A<0                  A and B are local variables
[2]       ∇
          G 4
1
          G ¯6
¯1
          X←¯6
          G X
¯1
          ∇H A                         Like G but has no explicit result
[1]       P←(A>0)-A<0                  P is a global variable
[2]       ∇
          H¯6
          P
¯1
          Y←H¯6                        H has no explicit result
VALUE ERROR                                 and hence produces a value
          Y←H ¯6                            error when used to right
            ∧                               of assignment
          ∇Z←FAC N;I                   FAC is the factorial function
[1]       Z←1
[2]       I←0
[3]       L1:I←I+1                     L1 becomes 3 at close of def
[4]       →0×⍳I>N                      Branch to 0 (out) or to next
[5]       Z←Z×I
[6]       →L1                          Branch to L1 (that is, 3)
[7]       ∇
          FAC 3
6
          FAC 5
120
          T∆FAC←3 5                    Set trace on lines 3 and 5 of FAC
          X←FAC 3
FAC[3] 1                               Trace of FAC
FAC[5] 1
FAC[3] 2
FAC[5] 2
FAC[3] 3
FAC[5] 6
FAC[3] 4
          T∆FAC←0                      Reset trace control
```

MECHANICS OF FUNCTION DEFINITION

```
        ∇G←M GCD N               Greatest common divisor
[1]     G←N                           function based on the
[2]     M←M|N                         Euclidean algorithm
[3]     →4×M≠0
[4]     [1]G←M                   Correction of line 1
[2]     [4]N←G                   Resume with line 4
[5]     [1⎕]                     Display line 1
[1]     G←M
[1]     [⎕]                      Display entire GCD Function
      ∇ G←M GCD N
[1]     G←M
[2]     M←M|N
[3]     →4×M≠0
[4]     N←G
      ∇                          Close of display, not close of def
[5]     →1                       Enter line 5
[6]     ∇                        Close of definition
        36 GCD 44                Use of GCD
4                                4 is GCD of 36 and 44
        ∇GCD                     Reopen def (Use ∇ and name only)
[6]     [4.1]M,N                 Insert between 4 and 5
[4.2]   [⎕]                      Display entire function
      ∇ G←M GCD N
[1]     G←M
[2]     M←M|N
[3]     →4×M≠0
[4]     N←G
[4.1]   M,N                      Fraction stays until close of def
[5]     →1
      ∇                          End of display
[6]     ∇                        Close of definition
        36 GCD 44
8   36                           Iterations printed by
4   8                                 line 5 (was line 4.1)
4                                Final result
        ∇GCD[⎕]∇                 Reopen, display, and close GCD
      ∇ G←M GCD N
[1]     G←M
[2]     M←M|N
[3]     →4×M≠0
[4]     N←G
[5]     M,N                      Line numbers have been
[6]     →1                            reassigned as integers
      ∇                          Close (Even number of ∇'s in all)
        ∇GCD                     Reopen definition of GCD
[7]     [5]                      Delete line 5 by linefeed
           ^
        ∇                        Close definition
        ∇Z←ABC X                  A function to show line editing
[1]     Z←(33×Q+(R×5)-6           A line to be corrected
[2]     [1⎕9]                     Initiate edit of line 1
[1]     Z←(33×Q+(R×5)-6           Types line, stops ball under 9
           /    1 /1              Slash deletes, digit inserts spaces
[1]     Z←(3×Q)+(T×5)-6           Ball stops at first new
[2]     ∇                              space. Then enter )  T
        FAC 5                     FAC still defined
120
        )ERASE FAC                Erase function FAC
        FAC 5                     Function FAC no longer exists
SYNTAX ERROR
        FAC 5
        ∧
        ∇Z←BIN N                  An (erroneous) function for
[1]     LA:Z←(Z,0)+0,Z                 binomial coefficients
[2]     →LA×N≥⍴Z∇
        BIN 3
VALUE ERROR
BIN[1] LA:Z←(Z,0)+0,Z             Suspended execution
                             ∧
        Z←1                       Assign value to Z
        →1                        Resume execution
1   3   3   1                     Binomial coefficients of order 3
        BIN 4
VALUE ERROR                       Same error (local variable Z
BIN[1] L1:Z←(Z,0)+0,Z                  does not retain its value)
                             ∧
        ∇BIN[.1]Z←1∇              Insert line to initialize Z
        )SI                       Display state indicator
BIN[1]  *                         Suspended on line 1 of BIN
        →1                        Resume execution (BIN now correct)
1   4   6   4   1
        ∇BIN[⎕]∇                  Display revised function
     ∇ Z←BIN N                         and close definition
[1]     Z←1
[2]     LA:Z←(Z,0)+0,Z
[3]     →LA×N≥⍴Z
     ∇
        SΔBIN←2                   Set stop on line 2
        Q←BIN 3                   Execute BIN

BIN[2]                            Stop due to stop control
        Z                         Display current value of Z
1
        →2                        Resume execution

BIN[2]                            Stop again on next iteration
        →2                        Resume

BIN[2]                            Stop again
        →0                        Branch to 0 (terminate)
```

INPUT AND OUTPUT

```
      ∇MULTDRILL N;Y;X                A multiplication drill
[1]   Y←?N                            ρN random integers
[2]   Y                               Print the random factors
[3]   X←⎕                             Keyboard input
[4]   →0×⍳X='S'                       Stop if entry is the letter S
[5]   →⍳X=×/Y                         Repeat if entry is correct product
[6]   'WRONG, TRY AGAIN'              Prints if preceding branch fails
[7]   →3∇                             Branch to 3 for retry
      MULTDRILL 12 12                 Drill for pairs in range 1 to 12
2  10
⎕:                                    Indicates that keyboard entry
      37                                   is awaited
WRONG, TRY AGAIN
⎕:
      20
6  7
⎕:
      'S'                             Entry of letter S stops drill
      ∇Z←ENTERTEXT                    Example of character (⍞) input
[1]   Z←''                            Make Z an empty vector
[2]   D←ρZ                            D is the length of Z
[3]   Z←Z,⍞                           Append character keyboard entry
[4]   →2×D≠ρZ                         Branch to 2 if length increased
[5]   ∇                                    (i.e., entry was not empty)
      Q←ENTERTEXT
THIS IS ALL                           Keyboard
 CHARACTER INPUT                           entries
                                      Empty input to terminate
      Q                               Display Q
THIS IS ALL CHARACTER INPUT
      N←5
      'NOTE: ⍳';N;' IS ';⍳N           Mixed output statement
NOTE:⍳5 IS 1  2  3  4  5
```

RECTANGULAR ARRAYS

```
      P←2 3 5 7
      ρP                              Dimension of P
4
      T←'OH MY'                       Character vector
      ρT
5
      P,P                             Catenation
2  3  5  7  2  3  5  7
      T,T
OH MYOH MY
      T,P
DOMAIN ERROR                          Characters cannot be catenated
      T,P                                  with numbers
      ^
      M←2 3ρ2 3 5 7 11 13          Reshape to produce a 2×3 matrix
      M                            Display of an array of rank >1
                                        is preceded by a blank line
 2   3   5
 7  11  13
      2 4ρT                        A 2×4 matrix of characters

OH M
YOH
      6ρM                          A matrix reshaped to a vector
2  3  5  7  11  13
      ,M                           Elements in row-major order
2  3  5  7  11  13
      P←,M
      P[3]                         Indexing (third element of P)
5
      P[1 3 5]                     A vector index
2  5  11
      P[ι3]                        The first three elements of P
2  3  5
      P[ρP]                        Last element of P
13
      M[1;2]                       Element in row 1, column 2 of M
3
      M[1;]                        Row 1 of M
2  3  5
      M[1 1;3 2]                   Rows 1 and 1, columns 3 2

  5  3
  5  3
      A←'ABCDEFGHIJKLMNOPQ'        The alphabet to Q
      A[M]                         A matrix index produces
                                        a matrix result
BCE
GKM
      A[M[1 1;3 2]]

EC
EC
      M[1;]←15 3 12                Respecifying the first row of M
      M

 15   3  12
  7  11  13
```

```
        Q←3 1 5 2 4 6                     A permutation vector
        P[Q]                              Permutation of P
 5  2  11  3  7  13
        Q[Q]                              A new permutation
 5  3  4  1  2  6
        P[3]                              Present index origin is 1
 5
        )ORIGIN 0                         Set index origin to 0
WAS 1
        P[3]
 7
        P[0 1 2]                          First three elements of P
 2  3  5
        ι5                                Result of index generator
 0  1  2  3  4                                 begins at origin
        )ORIGIN 1
WAS 0
        ι5
 1  2  3  4  5
```

FUNCTIONS ON ARRAYS

```
        V←?3ρ9                            Vector of 3 random integers (1-9)
        M←?3 3ρ9                          Random 3 by 3 matrix
        N←?3 3ρ9                          Random 3 by 3 matrix
        V
 2  1  7
        M

   7  9  4
   5  8  1
   1  5  7
        N
   1  4  1
   4  7  6
   9  8  5
        M+N                               Sum (element-by-element)

   8  13   5
   9  15   7
  10  13  12
```

```
          M⌈N                               Maximum

   7  9  4
   5  8  6
   9  8  7
          M≤N                               Comparison

 0 0 0
 0 0 1
 1 1 0
          +/V                               Sum-reduction of V
10
          ×/V                               Product-reduction
14
          +/[1]M                            Sum over first coordinate of M
13   22   12                                     (down columns)
          +/[2]M                            Sum over second coordinate of M
                                                 (over rows)
20   14   13
          +/M                               Sum over last coordinate
20   14   13
          ⌈/M                               Maximum over last coordinate
9   8   7
          X←1.5
          +/(1 2○X)*2                       Sin squared plus Cos squared
1
          ○/1 2,X                           Sin Cos X
0.07067822453
          Y←○/0 2,X                         (1-(COS X)*2)*.5
          Y
0.9974949866
          Y=1○X                             An identity
1
        M+.×N                               Ordinary matrix (+.× inner)
                                                 product
   79   123    81
   46    84    58
   84    95    66
        M+.≤N                               An inner product
  1   1   1
  1   1   1
  2   3   2
        M+.×V                               +.× inner product with vector
51   25   56                                     right argument
```

```
          V
   2  1   7
          V∘.×⍳5                         Outer product (times)

     2    4    6    8   10
     1    2    3    4    5
     7   14   21   28   35
          V∘.≤⍳9                         Outer product

   0  1  1  1  1  1  1  1  1
   1  1  1  1  1  1  1  1  1
   0  0  0  0  0  0  1  1  1
          V∘.×M                          An outer product of rank 3

    14   18    8
    10   16    2
     2   10   14
                                         A blank line between planes
     7    9    4
     5    8    1
     1    5    7

    49   63   28
    35   56    7
     7   35   49


                                         MIXED FUNCTIONS

         Q←?10⍴5                         A random 10 element vector
         Q                                    (range 1 to 5)
   1   4   3   4   5   4   2   1   4   2
         +/[1]Q∘.=⍳5                     Ith element of result is number
   2   2   1   4   1                          of occurences of the
                                              value I in Q
         2 1⍉M                           Ordinary transpose of M

     7   5   1
     9   8   5
     4   1   7
         ⍉M                              Ordinary transpose of M (monadic)

     7   5   1
     9   8   5
     4   1   7
```

```
         T←2 3 4ρι24                          An array of rank 3
         T

    1    2    3    4
    5    6    7    8
    9   10   11   12

   13   14   15   16
   17   18   19   20
   21   22   23   24

         3 1 2⍉T                              Transpose of T (dimension
                                                   of result is 3  4  2)
    1   13
    2   14
    3   15
    4   16

    5   17
    6   18
    7   19
    8   20

    9   21
   10   22
   11   23
   12   24
         1 1⍉M                                Diagonal of M
 7  8    7
         1 1 2⍉T                              Diagonal section in first
                                                   two coordinates of T

    1    2    3    4
   17   18   19   20
         X←○(0,ι5)÷6
         )DIGITS 4                            Set number of output digits to 4
WAS 10
         ⍉1 2 3∘.○X

   0.000E0     1.000E0      0.000E0          Table of sines, cosines,and
   5.000E¯1    8.660E¯1     5.774E¯1              tangents in intervals
   8.660E¯1    5.000E¯1     1.732E0               of 30 degrees
   1.000E0     1.744E¯16    5.734E15
   8.660E¯1   ¯5.000E¯1    ¯1.732E0
   5.000E¯1   ¯8.660E¯1    ¯5.774E¯1
```

```
          Q
1    4    3    4    5    4    2    1    4    2
          3⌽Q                                      Rotate to left by 3 places
4    5    4    2    1    4    2    1    4    3
          ¯3⌽Q                                     Rotate to right by 3 places
1    4    2    1    4    3    4    5    4    2
          0 1 2⌽[1]M                               Rotate columns by
                                                        different amounts
   7    8    7
   5    5    4
   1    9    1
          ¯2⌽[2]M                                  Rotation of rows all
                                                        by 2 to right
   9    4    7
   8    1    5
   5    7    1
          1 2 3⌽M                                  Rotation of rows

   9    4    7
   1    5    8
   1    5    7
          ⌽Q                                       Reversal of Q
2    4    1    2    4    5    4    3    4    1

          ⌽[1]M                                    Reversal of M along
                                                        first coordinate
   1    5    7
   5    8    1
   7    9    4
          ⌽M                                       Reversal along last coordinate

   4    9    7
   1    8    5
   7    5    1
```

```
      U←Q>4
      U
0  0  0  0  1  0  0  0  0  0
      U/Q                          Compression of Q by logical
5                                      vector U
      (~U)/Q                       Compression by not U
1  4  3  4  4  2  1  4  2
      +/U/Q
5
      1 0 1/[1]M                   Compression along first
                                       coordinate of M
  7  9  4
  1  5  7
      1 0 1/M                      Compression along last
                                       coordinate
  7  4
  5  1
  1  7
      (,M>5)/,M                    ,M is  7 9 4 5 8 1 1 5 7
7  9  8  7                         All elements of M which exceed 5
      V←1 0 1 0 1
      V\⍳3                         Expansion of iota 3
1  0  2  0  3
      V\M                          Expansion of rows of M

  7  0  9  0  4
  5  0  8  0  1
  1  0  5  0  7
      V\'ABC'                      Expansion of literal vector
A B C                                  inserts spaces
      10⊥1 7 7 6                   Base 10 value of vector 1 7 7 6
1776
      8⊥1 7 7 6                    Base 8 value of 1 7 7 6
1022
      (4⍴10)⊤1776                  4 digit base 10 representation
1  7  7  6                             of number 1776
      (3⍴10)⊤1776                  3 digit base 10 representation
7  7  6                                of 1776
      10 10⊤1776
7  6
      10⊤1776
6
      24 60 60⊥1 3 25              Mixed base value of 1 3 25
3805                                   (time radix)
      24 60 60⊤3805                Representation of number 3805
1  3  25                               in time radix
      2⊥1 0 1 1 0                  Base 2 value
22
```

```
      M
  7  9  4
  5  8  1
  1  5  7
      )ORIGIN 0
WAS 1
      M[2;0]                         Indexing of matrix in 0-origin.
1                                        Note relation to indexing of
      (,M)[(⍴M)⊥2,0]                     ravel of M
1
      )ORIGIN 1                      Restore 1-origin
WAS 0
      P
2  3  5  7  11  13
      P⍳7                            Index of 7 in vector P
4                                    7 is 4th element of P
      P⍳6                            6 does not occur in P, hence
7                                        result is 1+⍴P
      P⍳4 5 6 7
7  3  7  4
      Q←5 1 3 2 4                    A permutation vector
      R←Q⍳⍳⍴Q                        R is the permutation inverse to Q
      R
2  4  3  5  1
      Q[R]
1  2  3  4  5
      A←'ABCDEFGHIJKLMNOPQ'
      A←A,'RSTUVWXYZ'
      A                              A is the alphabet
ABCDEFGHIJKLMNOPQRSTUVWXYZ
      A⍳'C'                          Rank of letter C in alphabet is 3
3
      J←A⍳'CAT'
      J
3  1  20
A[J]
CAT
```

```
      M←3 5⍴'THREESHORTWORDS'    A matrix of characters
      M

THREE
SHORT
WORDS
      J←A⍳M                      Ranking of M produces a matrix
      J
  20   8  18   5   5
  19   8  15  18  20
  23  15  18   4  19
      A[J]                       Indexing by a matrix produces
                                      a matrix
THREE
SHORT
WORDS
      3?5                        Random choice of 3 out of 5
5  1  2                               without replacement
      6?5
DOMAIN ERROR
      6?5
       ^
      X←8?8                      A random permutation vector
      X
4  6  7  2  5  1  8  3
      ⍋X                         Grading of X
6  4  8  1  5  2  3  7
      X[⍋X]                      Arrange in ascending order
1  2  3  4  5  6  7  8
      X[⍒X]                      Arrange in descending order
8  7  6  5  4  3  2  1
      U←A∊'NOW IS THE TIME'      Membership
      '01'[1+U]
00001001100011100011001000
      U/A
EHIMNOSTW
      (⍳8)∊3 7 5
0  0  1  0  1  0  1  0
```

The APL graphic character set consists of:

26 alphabetics

A B C D E F G H I J
K L M N O P Q R S T
U V W X Y Z

26 alphabetics underscored (overstrikes)

$\underline{A}$ $\underline{B}$ $\underline{C}$ $\underline{D}$ $\underline{E}$ $\underline{F}$ $\underline{G}$ $\underline{H}$ $\underline{I}$ $\underline{J}$
$\underline{K}$ $\underline{L}$ $\underline{M}$ $\underline{N}$ $\underline{O}$ $\underline{P}$ $\underline{Q}$ $\underline{R}$ $\underline{S}$ $\underline{T}$
$\underline{U}$ $\underline{V}$ $\underline{W}$ $\underline{X}$ $\underline{Y}$ $\underline{Z}$

10 numerics

0 1 2 3 4 5 6 7 8 9

18 special characters common to Multics, 37ttys, and 963 2741s

< = > - + ? * _ ' (
) | ; : , . /
blank

4 special characters common to Multics and 37ttys, but not 963 2741s

~ [ ] \

25 special characters not in Multics

¯ ≤ ≥ ≠ ∨ ∧ ÷ × ∊ ⍴
↑ ↓ ⍳ ○ → ← ⌈ ⌊ ∇ ∆
∘ ⎕ ∩ ⊥ ⊤

6 special characters not in Multics, but appearing only as data in APL

¨ ⍵ ⍺ ⊂ ⊃ ∪

16 overstrike combinations

⍱ ⍲ ⊖ ⌿ ⍫ ⍟ ! ⌽ ⍉ ⍎
⍒ ⍋ ⍝ ⍞ ⌶ ⍀

131 graphic characters (NL, BS, HT not included)