APL
PRESS
Box 27, Swarthmore, PA 19081

3

2 1

1977

APL NEWS

APL PRESS

BOX 27, SWARTHMORE, PA 19081 (215) 328-9771

Our mail continues to bring in requests for information on the use of APL in a wide variety of areas. Recent examples include commercial applications, population genetics, critical path analysis, computer science and mathematics at the high school level, APL implementations on micro-computers, and civil engineering.

Many correspondents ask to be put in touch with others working in the same area. We would be pleased to pass on or publish statements of work in progress.

The APL 76 Conference Proceedings are now available from ACM, Box 12105, Church Street Station, New York 10249. Although commercial data processing forms the main theme, the proceedings touch on a variety of other topics, including the organization of APL libraries, programming techniques, and text editing, as well as certain language questions.

Jeff Shallit

Parallel to the self-reproducing function discussed in NL2 is the self-reproducing expression (SRE), defined informally by Phil Abrams as "an expression which when typed in from a terminal outputs what was typed in". Use of the execute function permits the following formal definition: A character vector *C* is an SRE if:

(1) ∧/*C*=⍎*C*, and (2) ∧/~*C*∊'*ABC...Z∆ABC...Z∆*'.

For example, if *FAKE*←'*FAKE*', then ∧/*FAKE*=⍎*FAKE* results in a 1, but *FAKE* is not an SRE because of condition 2. Three examples of SRE's follow, the last of which is the shortest known:

```
'''[1 1,26⍴⍳13]'[1 1,26⍴⍳13]

1⌽26⍴14⍴13⍴'''1⌽26⍴14⍴13⍴'

1⌽22⍴11⍴'''1⌽22⍴11⍴'''
```

The following questions present themselves: Is there a longest SRE? Are there only a finite number of SRE's? We will show that the answer to both these questions is "no".

In any particular implementation of APL there are only finite number of SRE's because there is only a finite amount of space available. In the APL language, however, where we are not concerned with such limitations as ⎕WA and ⎕PP, we can find an infinite number of SRE's. The following function produces a different SRE (of length 50+8×N) for each positive integer argument N:

```
    ∇ Z←AGEN N
[1]   Z←1⌽(50+8×N)⍴(25+4×N)⍴'''1⌽(42+8×⍴',(⍕1,N⍴1),')⍴(21+
      4×⍴',(⍕1,N⍴1),')⍴''' ∇
```

The method of construction used in AGEN is best seen in examples (with ⎕PW←60):

```
      AGEN 1
1⌽(42+8×⍴1 1)⍴(21+4×⍴1 1)⍴'''1⌽(42+8×⍴1 1)⍴(21+4×⍴1 1)⍴'''

      ⍎AGEN 1
1⌽(42+8×⍴1 1)⍴(21+4×⍴1 1)⍴'''1⌽(42+8×⍴1 1)⍴(21+4×⍴1 1)⍴'''

      AGEN 2
1⌽(42+8×⍴1 1 1)⍴(21+4×⍴1 1 1)⍴'''1⌽(42+8×⍴1 1 1)⍴(21+4×⍴1 1
      1)⍴'''
      AGEN 3
1⌽(42+8×⍴1 1 1 1)⍴(21+4×⍴1 1 1 1)⍴'''1⌽(42+8×⍴1 1 1 1)⍴(21+4
      ×⍴1 1 1 1)⍴'''

      ⍴X←AGEN 1000
8050
      ∧/X=⍎X
```

Problems:

(1) Can you find an SRE C such that 22>⍴C ?

(2) Can you find a character vector C that is not immediately self-reproducing, but which generates itself after two or more applications of the execute function? For example, is there a C such that ∧/C=⍎⍎C but that ~∧/C=⍎C ?

(3) Is there an SRE of odd length?

Answers to problems posed in NL2.

Problem 1 (Mastermind). Line 2 defines a matrix TABLE of dimension 1296 4 consisting of all possible four-digit arrangements of the digits 1,2,3,4,5,6. This table represents all possible choices of the six colored pegs in four slots. This is a typical example of the use of the encode function in dealing with arrangements.

The user enters in line 9 the number of black pins (to indicate the number of pins correct in both color and position), followed by the number of white pins (to indicate the number of pegs correct in color, but incorrect in position). Thus in line 11 the inner product *TABLE*+.=*GUESS* determines how many pegs are correct in both color and position for each entry in *TABLE*.

In line 12 we consider the quantity +/*INPUT*. This number corresponds to the number of pegs correct in color, without regard to position. The expressions +/[2]*TABLE*∘.=⍳1↑ρ*COLORS* and +⌿*GUESS*∘.=⍳1↑ρ*COLORS* form "frequency tables" indicating how many pegs of each color occur in any arrangement. The +.⌊ inner product then "matches" the number of colors identical in both arrangements.

<u>Problem</u> 2 (Golden Mean). *G*←.5+.5×5*.5 and *G̲*←-÷.5+.5×5*.5 or *G̲*←.5-.5×5*.5.

<u>Problem</u> 3 (Representations of name lists).

```
      ∇ Z←CANON X;A
[1]     →(1=ρρX)/VECTOR
[2]     A←(¯1+(' '=⌽X)⊥1)⌽X
[3]     Z←(∧\∨⌿≠' ')/A
[4]     →0
[5]    VECTOR:A←' '≠X←' ',X
[6]     Z←1↓(A∨1⌽A)/X ∇

      ∇ Z←CONVERT X;I;D;⎕IO
[1]     ⎕IO←1
[2]     →(1=ρρX)/VECTOR
[3]     Z←¯1↓(,(1++/X≠' ')∘.≥⍳1+¯1↑ρX)/,X,' '
[4]     →0
[5]    VECTOR:I←(X=' ')/⍳ρX
[6]     D←(I,1+ρX)-1+0,I
[7]     Z←((ρD),⌈/D)ρ(,D∘.≥⍳⌈/D)\,(X≠' ')/X ∇
```

<u>Problem</u> 4 (Self-reproducing functions). G.L. Stoddard of Interstate Electronics Corp. points out that the shortest self-reproducing function has the canonical representation 2 1ρ'*BA*', where the global variable *A* is pre-specified as *A*←3 7ρ'   ∇ *B*[1]   *A*   ∇'. However, if we disallow the use of global variables, then the shortest known function is:

```
    ∇ S
[1]    (3 5 ρ'    ∇[1]  '), 3 ¯36↑⎕CR'S'
    ∇
```

AN αω COMPILER

Rick Mayforth
Control Data Corp.

The following functions provide a solution to the problem posed in the first newsletter, fixing the definition of any function presented in the αω form defined in Iverson, Elementary Analysis. Name conflicts will not occur if the user avoids names ending in 9, and any name assigned a value within the function defined will be localized. The function *DEF* is used as follows:

```
      DEF                              DEF
PLUS:α+ω                         B:(Z,0)+0,Z←B ω-1:ω=0:1
      3 PLUS 4                         B 5
7                                1 5 10 10 5 1
```

```
      ∇DEF[⎕]∇
    ∇ DEF;B9;E9;⎕IO;ALPH9;X9
[1]   ⎕IO←1
[2]   ⍎(1 0 ∧.='αω'∊X9←,⍞)/'X9[(X9=''α'')/⍳⍴⍴X9]←''ω'''
[3]   →((∨/∧/' '=X9)∨~∨/ 2 4 ∊1↑⍴X9←':' MAT9 ' ',B9←,X9)/M9
[4]   ALPH9←'ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTU
      VWXYZ0123456789'
[5]   E9←'Z9←',(('α'∊B9)/' X9 '),X9[1;],('ω'∊B9)/' Y9 '
[6]   E9←E9,,((0 1 ↓B9)∨.≠' ')⌿B9←(0 0 ⍴'') SPECIFY9⌽B9
[7]   →(2=1↑⍴X9)/L9
[8]   E9←E9 RJN9 'Z9←',X9[4;]
[9]   E9←E9 RJN9 '→(',X9[3;],')/0'
[10] L9:E9←E9 RJN9 'Z9←',X9[2;]
[11]  →(0=⍴⍴X9←⎕FX ':' MAT9 REPLACE9 ¯1↓,E9,':')/M9
[12]  →0
[13] M9:'NOT DONE' ∇
    ∇ Z←A MAT9 B;C;D;E
[1]   E←⍴D←¯1↓(C,0)-0,C←(+/~C)↓⍋C←(~≠\''''=B)∧A=B←,B,A
[2]   Z←(+/∧\' '=Z)⌽Z← 0 ¯1 ↓(E,C)⍴(,⌽D∘.≥⍳C←⌈/D)\B ∇
    ∇ Z←REPLACE9 X;A;B;C;D
[1]   Z←' ',⍉' ',[1] X,[0.5] ' '
[2]   Z[B←(C∧X∊D)/⍳⍴X;]←(2 4 ⍴' Y9  X9 ')[((C←(~≠\''''=X)∧A≤⍴
      D)/A←(D←'ωα')⍳X;]
[3]   C←(⍴Z)⍴ 0 0 1 0 ·
[4]   C[B; 1 2 4]←1=1
[5]   Z←(,C)/,Z ∇
    ∇ R←A RJN9 B;X
[1]   X←(¯1↑⍴A←(¯2↑ 1 1 ,⍴A)⍴A)⌈¯1↑⍴B←(¯2↑ 1 1 ,⍴B)⍴B
[2]   R←(((1↑⍴A),X)↑A),[1]((1↑⍴B),X)↑B ∇
    ∇ Z←A SPECIFY9 B;C;D;E
[1]   Z← 0 0 ⍴''
[2]   →(0=⍴B←(B⍳'←')↓B)/0
[3]   C←';',⌽(¯1+C⍳' ')↑C←(+/∧\' '=C)↓C←(+/∧\B∊ALPH9)↑B
[4]   Z←(~∨/(((1↑⍴A),E)↑A)∧.=(E←(1↓⍴A)⌈⍴C)↑C)⌿(1,⍴C)⍴C
[5]   Z←Z RJN9(A RJN9 C) SPECIFY9 B ∇
```

APL DOMAIN

R. H. Lathwell
IBM Corp.

APL functions take arrays of numbers or characters as arguments and return arrays as corresponding results. More precisely, an APL function is a relationship between a domain of permissible arguments and a range of corresponding results. Functions are classified according to their domains. For example: boolean functions have domains and a range consisting solely of 0 and 1, arithmetic functions map numbers to numbers, division excludes 0 as a right argument.

The domains of all scalar functions consist of numbers, and I'd like to suggest a small set which is representative of the real number system.

First, choose a few interesting numbers:

```
S←2,(*1),2.5,○1
```

\- an even number, e, a rational number, and pi; an integer, a number with a decimal part between zero and .5, a decimal part of .5, and a decimal part between .5 and 1.

There are as many numbers between zero and 1 as there are greater than 1. Extending *S* by

```
S←(⌽÷S),1,S
```

results in a set with a representative distribution about 1. Real numbers are also distributed symmetrically about zero:

```
S←(⌽-S),0,S
```

The resulting 19-element vector contains numbers with 30 or so interesting properties, the boolean set, a subset of integers, and so on, and is representative of the set of real numbers. It proves adequate for determining the domain and range of all APL functions.

COMPLEX PRIMES

E. E. McDonnell
IBM Corp.

Algorithms have appeared [1,2], written in Algol, which produce the complex primes within a given range. These algorithms are moderately complicated, each over thirty statements long. The present note gives a sequence of eight APL statements for producing a representation of a specified range of complex primes.

Fermat showed that a real prime of the form 1+4×*N* has a unique representation as the sum of two squares (see, for example, reference [3]). For example, 5=2 1+.*2 and 13=2 3+.*2. This representation is related to the decomposition of a real prime of this kind into complex prime factors. In other words, 5 and 13 are not prime in the complex domain. Their decompositions (using an orthography proposed in [4]) are 5=2*I*1×2*I*¯1 and 13=2*I*3×2*I*¯3. This decomposition is unique to within associates (an associate of a number is a number differing from the given number by a unit factor; the complex units are 1, ¯1, 0*I*1, and 0*I*¯1). The complex primes are exhausted by these primes, the real primes of the form 3+4×*N* (which remain prime in the complex domain), and the complex prime divisors of 2.

The complex prime divisors of 2 are 1*I*1 and its associates, 1*I*¯1, ¯1*I*1, and ¯1*I*¯1. Just as 2 and its associate ¯2 are unique among the real primes, in being the only even primes, so 1*I*1 and its associates are unique among the complex primes in having real and imaginary parts with the same parity.

The construction given here provides a table *T* indicating by an entry of 1 at *T*[*R*;*I*] that the complex integer with real part 2×*R* and imaginary part ¯1+2×*I* is prime, and by an entry of 0 that it is composite. The table shows one of the primes; there are eight in all formed using the same *R* and *I*: they are the number found in the table and its associates, and the conjugate of the number found in the table and its associates. Since a complex prime (other than 1*I*1 and its associates) always has real and imaginary parts of opposite parity, and always has either an associate or an associate of its conjugate in the first quadrant, we may choose as representative of eight different complex primes the one having even, positive real part, and odd, positive imaginary part.

To construct a table which shows which are prime among the complex integers with real part less than or equal to *M*, and imaginary part less than or equal to *N*, we proceed as follows. First, we need a list of the real primes up to

(*M*,*N*)+.*2. Such a list may be obtained by using the functions *P* and *Q*, expressed in the αω-notation [5,6]:

```
P:2:L←ω:Q 2
Q:Q ω,(∧⌿0≠ω∘.|V)/V←K+⍳K:L≤K←⌈/ω:ω
```

Q depends on Bertrand's postulate (proven by Chebyshev) [that a prime number will always be found between *X* and 2×*X*, for *X* greater than 3.

Next, proceed as follows:

```
A←(0=2|⍳M)/⍳M
B←(1=2|⍳N)/⍳N
E←A*2
O←B*2
S←E∘.+O
T←S∊P (M,N)+.*2
```

The information in *T* may be displayed in a compact form by forming a companion table *U*, in which the 0's and 1's in *T* are replaced by spaces and +'s. From this, a table *V* may be formed, with row and column labels added to make reading easier:

```
U←' +'[1+T]
V←(' ',[1]2 0⍕((⍴A),1)⍴2×⍳⍴A),(⍉2 0⍕10
1⍴((⍴B),1)⍴¯1+2×⍳⍴B),[1]U
```

Such a table for a *T* of 6 rows and 10 columns (covering complex integers with even real part from 2 to 12, and odd imaginary part from 1 to 19, is shown in the figure:

```
       11111
  1357913579
 2++++   +++
 4+ + ++ +
 6+ +  +   +
 8 +++  + +
10++ ++ + ++
12    +  + +
```

References

[1] Dunham, K. B., "An Algorithm to Produce Complex Primes, CSIEVE," CACM 13, 1, January 1970

[2] Bratley, Paul, "An Improved Algorithm to Produce Complex Primes," CACM 13, 11, November 1970

[3] Davenport, H., The Higher Arithmetic, Harper & Bros., New York, 1960

[4] McDonnell, E. E., "Complex Floor," APL Congress 73, Gjerlov et al eds., North Holland, Amsterdam, 1973

[5] Iverson, K. E., Elementary Analysis, APL Press, Swarthmore, PA, 1976

[6] Orth, D. L., Calculus in a New Key, APL Press, Swarthmore, PA, 1976

[7] Dickson, L. E., History of the Theory of Numbers, volume I, Chelsea, New York, 1952