Answers to Quiz on Programmer-Defined Control Operations

See Icon Analyst 55, page 16, for the questions.

1.

(a)

```
procedure ExchangePDCO(L)
  local i
  while i := @L[1] do
    suspend @L[1] | i
end
```

(b)

```
procedure CumulativePDCO(L)
  local i
  i := 0
  while i +:= @L[1] do
    suspend i
end
```

(c)

```
procedure IntegerPDCO(L)
  local x
  while x := @L[1] do
    if type(x) == "integer" then suspend x
end
```

Note: The problem was poorly phrased. A procedure that filters *out* non-integer values should be named IntegerPDCO{} as above, not NonintegerPDCO{}. Notice that the version given above only passes through values of type integer and does not attempt to convert values of other types. The code for the latter interpretation is

```
while x := @L[1] do
  suspend integer(x)
```

(d)

```
procedure ModnPDCO(L)
  local i, j
  every i := seq() do {
    j :=  @L[1]  | fail
    suspend j % i
    }
end
```

None of these PDCOs has a problem with infinite sequences, *per se*. However, IntegerPDCO{} stops producing values but doesn't terminate if an infinite sequence stops producing integer values.

2.

(a) Term-wise sum of the integers and the Fibonacci numbers: 2, 3, 5, 7, 10, 14, 20, 29, 43, 65, 100, 156, 246, 391, 625, 1003, …

(b) Alternating sums and differences of the primes and Fibonacci numbers: 3, 2, 7, 4, 16, 5, 30, -2, 57, -26, 120, -107, 274, -334, 657, -934, 1656, -2523, 4248, -6694, 11019, -17632, …

(c) The integers *i* repeated *i* times: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, …

(d) Differences of successive primes: 1, 2, 2, 4, 2, 4, 2, 4, 6, 2, 6, 4, 2, 4, 6, 6, 2, 6, 4, 2, 6, 4, 6, 8, 4, 2, 4, 2, 4, 14, 4, 6, 2, 10, …

(e) The Fibonacci numbers with insertions, where necessary, to make terms alternate between odd and even. The inserted terms are underscored: 1, 2, 1, 2, 3, 4, 5, 8, 13, 14, 21, 34, 55, 56, 89, 144, 233, 234, 377, 610, 987, 988, 1597, 2584, 4181, 4182, 6765, 10946, 17711, 17712, …

(f) The Fibonaci numbers and the primes interleaved and then reduced modulo 8: 1, 2, 1, 3, 2, 5, 3, 7, 5, 3, 0, 5, 5, 1, 5, 3, 2, 7, 7, 5, 1, 7, 0, 5, 1, 1, 1, 3, 2, 7, …

(g) The Fibonacci numbers interleaved with the primes taken mod 8: 1, 2, 1, 3, 2, 5, 3, 7, 5, 3, 8, 5, 13, 1, 21, 3, 34, 7, 55, 5, 89, 7, 144, 5, 233, 1, 377, 3, 610, 7, …

(h) The sizes (numbers of digits) of the Fibonacci numbers: 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 13, 13, 13, 13, …

(i) The individual digits of the Fibonacci numbers (since ! is a generator, each application runs through all the characters (digits) of the term to which it is applied): 1, 1, 2, 3, 5, 8, 1, 3, 2, 1, 3, 4, 5, 5, 8, 9, 1, 4, 4, 2, 3, 3, 3, 7, 7, 6, 1, 0, 9, 8, 7, 1, 5, 9, 7, 2, 5, 8, 4, 4, 1, 8, 1, 6, 7, 6, 5, 1, 0, 9, 4, 6, 1, 7, 7, 1, 1, 2, 8, 6, 5, 7, 4, 6, 3, 6, …

3.

(a) The integers *i* repeated *p* times by the corresponding primes *p*:

```
ReplPDCO{seq(), prime()}
```

(b) The integers *i* repeated by *i* repeated *i* times (as in solution 2(c)):

```
ReplPDCO{seq(), ReplPDCO{seq(), seq()}}
```

(c) The primes interleaved with the primes plus 3:

```
InterPDCO{primeseq(), primeseq() + 3}
```

(d) The primes made into an odd-even sequence:

```
OddEven{primesseq()}
```

4.

(a) Puzzle1PDCO{*e1*, *e2*, …, *en*} produces results from its argument expressions selected at random.

(b) Puzzle2PDCO{*e1*, *e2*} skips the number of terms in *e1* given by *e2*. For example,

```
Puzzle2PDCO{seq(), primeseq()}
```

produces

1, 4, 8, 14, 22, 34, 48, 66, 86, 110, 140, 172, 210, 252, 296, 344, 398, 458, 520, 588, 660, 734, 814, 898, 988, 1086, 1188, 1292, 1400, 1510, 1624, 1752, 1884, 2022, 2162, …

(c) Puzzle3PDCO{*e*} fills in *e* with runs of consequetive integers as necessary. For example,

```
Puzzle3PDCO{
  InterleavePDCO{primeseq(), seq()}
  }
```

produces

2, 1, 2, 3, 2, 3, 4, 5, 4, 3, 4, 5, 6, 7, 6, 5, 4, 5, 6, 7, 8, 9, 10, 11, 10, 9, 8, 7, 6, 5, 6, 7, 8, 9, 10, 11, 12, 13, 12, 11, 10, 9, 8, 7, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, …