ALGORITHM 72
COMPOSITION GENERATOR

L. Hellerman and S. Ogden
IBM-Product Development Laboratory, Poughkeepsie, N. Y.

```
procedure comp (c, k); value k; integer array c;
  integer k;
comment Given a k-part composition c of the positive integer n,
  comp generates a consequent composition if there is one. If
  comp operates on each consequent composition after it is found,
  all compositions will be generated, provided that 1, 1, . . ., 1,
  n−k+1 is the initial c. If c is of the form n−k+1, 1, 1, . . . , 1,
  there is no consequent, and c will be replaced by a k vector of
  0's. Reference: John Riordan, An Introduction to Combi-
  natorial Analysis, John Wiley and Sons, Inc., New York, 1958,
  Chapter 6;
begin integer j; integer array d [1:k];
  if k = 1 then go to last;
  for j := 1 step 1 until k do d [j] := c [j] − 1;
test: if d[j]>0 then go to set;
      j := j−1;
      go to if j = 1 then last else test;
set:  d [j] := 0;
      d [j − 1] := d [j − 1] + 1;
      d [k] := c [j] − 2;
      for j := 1 step 1 until k do c [j] := d[j] + 1;
      go to exit;
last: for j := 1 step 1 until k do c [j] := 0;
exit: end comp
```

CERTIFICATION OF ALGORITHM 42
INVERT (T. C. Wood, *Comm. ACM*, Apr., 1961)

Anthony W. Knapp and Paul Shaman
Dartmouth College, Hanover, N. H.

INVERT was hand-coded for the LGP-30 using machine language and the 24.0 floating-point interpretive system, which carries 24 bits of significance for the fractional part of a number and five bits for the exponent. The following changes were found necessary:

```
(a) if j = n+1 then a[i, j] := 1.0 else a[i, j] := 0.0;
    should be
    if j = n+i then a[i, j] := 1.0 else a[i, j] := 0.0;

(b) for k := j step 1 until 2 × n do
      a[i, k] := a[i, k]/a[i, j];
    should be
    for k := 2 × n step −1 until i do
      a[i, k] := a[i, k]/a[i, i];

(c) if l ≠ i then for k := 1 step 1 until 2 × n do
      a[l, k] := a[l, k] − a[i, k] × a[l, j];
    should be
    if l ≠ i then for k := 2 × n step −1 until i do
      a[l, k] := a[l, k] − a[i, k] × a[l, i];
```

Given these changes, j becomes superfluous in the second i loop, and the other references to j may be changed to references to i.

INVERT obtained the following results:

The computer inverted a 17-by-17 matrix whose elements were integers less than ten in absolute value. When the matrix and its inverse were multiplied together, the largest nondiagonal element in the product was −.00003. Most nondiagonal elements were less than .00001 in absolute value.

INVERT was tested using finite segments of the Hilbert matrix. The following results were obtained in the 4 × 4 case:

16.005	−120.052	240.125	−140.082
−120.052	1200.584	−2701.407	1680.917
240.126	−2701.411	6483.401	−4202.217
−140.082	1680.920	−4202.219	2801.446

The exact inverse is:

16	−120	240	−140
−120	1200	−2700	1680
240	−2700	6480	−4200
−140	1680	−4200	2800

INVERT was also coded for the LGP-30 in machine language and the 24.1 extended range interpretive system. This system, which uses 30 significant bits for the fraction, obtained the following as the inverse of the 4 × 4 Hilbert matrix:

16.000	−120.001	240.001	−140.001
−120.001	1200.006	−2700.015	1680.010
240.001	−2700.016	6480.037	−4200.024
−140.001	1680.010	−4200.024	2800.016

The program coded in the 24.0 interpretive system successfully inverted a matrix consisting of ones on the minor diagonal and zeros everywhere else.

REMARK ON ALGORITHM 52
A SET OF TEST MATRICES (John R. Herndon, *Comm. ACM*, Apr. 1961)

G. H. Dubay
University of St. Thomas, Houston, Tex.

In the assignment statement

c := t×(t + 1)×(t + t − 5)/6;   (a)

the t is undefined. A suitable definition would be provided by preceding (a) with t := n; .

CERTIFICATION OF ALGORITHM 68
AUGMENTATION (H. G. Rice, *Comm. ACM*, Aug. 1961)

L. M. Breed
Stanford University, Stanford, Calif.

AUGMENTATION was transliterated into BALGOL for the Burroughs 220, and proved successful in a number of test cases. However, the following algorithm has exactly the same effect and is considerably simpler:

```
real procedure Aug(x, y); value x, y; integer x, y;
begin if x<0 then L : go to L else Aug := x+y end Aug
```