ALGORITHM 67
CRAM
JOHN CAFFREY
Director of Research, Palo Alto Unified School District, Palo Alto, California

```
procedure CRAM (n, r, a) Result: (f); value n, r; integer
  n, r; real array a, f;
comment CRAM stores, via an unspecified input procedure
READ, the diagonal and superdiagonal elements of a square sym-
metric matrix e, of order n, as a pseudo-array of dimension
1:n(n + 1)/2. READ (u) puts one number into u. Elements e[i, j]
are addressable as a[c + j], where c = (2n - i)(i - 1)/2 and c[i + 1]
may be found as c[i] + n - i. Since c[1] = 0, it is simpler to develop
a table of the c[i] by recursion, as shown in the sequence labelled
"table". Further manipulation of the elements so stored is illus-
trated by premultiplying a rectangular matrix f, of order n, r, by
the matrix e, replacing the elements of f with the new values, re-
quiring a temporary storage array v of dimension 1:n;
begin integer i, j, k, m; real array v[1:n]; real s;
  integer array c[1:n];
table: j := - n; k := n + 1; for i := 1 step 1 until n do
       begin
       j := j + k - i; c[i] := j end;
load: for i := 1 step 1 until n do
      begin for j := i step 1 until n do READ (v[j]); m :=
      c[i];
      for k := i step 1 until n do a[m + k] := v[k] end;
premult: for j := 1 step 1 until r do
         begin for i := 1 step 1 until n do
               begin s := 0.0;
                 for k := 1 step 1 until i do
                 begin m := c[k]; s := s + a[m + i]
                   ×f[k, j] end;
                 for k := i + 1 step 1 until n do
                     s := s + a[m + k] × f[k, j]; v[i] = s
               end;
               for k := 1 step 1 until n do f[k, j] = v[k]
         end
end CRAM
```

CERTIFICATION OF ALGORITHM 67
CRAM (J. Caffrey, *Comm. ACM 4* (July 1961), 322)
A. P. RELPH
Atomic Power Div., The English Electric Co., Whetstone, England

CRAM was translated using the DEUCE ALGOL compiler with the following corrections:

V[i] = S *was changed to* V[i] := S
f[k,j] = V[k] *was changed to* f[k,j] := V[k]

It is quicker not to use the table of the C[i] in the "load" sequence and instead use the following sequence:

```
load: m := n × (n+1)/2;
      for i := 1 step 1 until m do READ (a[i]);
```