Consider the program fragment

```
      DO 14 I=1,N
      DO 14 J=1,N
   14 V(I,J)=(I/J)*(J/I)
```

A modest familiarity with Fortran tells us that this doubly nested DO loop assigns something to each element of an N by N matrix V. What are the values assigned? I and J are positive integer variables and, in Fortran, integer division truncates toward zero. Thus when I is less than J, (I/J) is zero; conversely, when J is less than I, (J/I) is zero. When I equals J, both factors are one. So (I/J)*(J/I) is one if and only if I equals J; otherwise it is zero. The program fragment puts ones on the diagonal of V and zeros everywhere else. (V becomes an identity matrix.) How clever!

Or is it?

Suppose you encountered this fragment in a larger program. If your knowledge of Fortran is sufficiently deep, you may have enjoyed the clever use of integer division. Possibly you were appalled that two divisions, a multiplication, and a conversion from integer to floating point were invoked when simpler mechanisms are available. More likely, you were driven to duplicating the reasoning we gave above to understand what is happening. Far more likely, you formed a vague notion that something useful is being put into an array and simply moved on. Only if motivated strongly, perhaps by the need to debug or to alter the program, would you be likely to go back and puzzle out the precise meaning.

A better version of the fragment is

```
C  MAKE V AN IDENTITY MATRIX
      DO 14 I = 1,N
         DO 12 J = 1,N
   12       V(I,J) = 0.0
   14    V(I,I) = 1.0
```

This zeros each row, then changes its diagonal element to one. The intent is now reasonably clear, and the code even happens to execute faster. Had we been programming in PL/I, we could have been more explicit:

1