```
DECLARE (N, R) FIXED BINARY(31);
DECLARE YES BIT(1) INITIAL ('1'B), NO BIT(1) INITIAL ('0'B);
DECLARE PRIME BIT(1);

DO WHILE (YES);
   GET LIST(N);
   IF N <= 1 THEN
      PUT EDIT ('ILLEGAL INPUT N =', N, ' <= 1')
            (SKIP, X(10), A, F(5), A);
   ELSE DO;
      PRIME = YES;
      IF N > 2 & MOD(N, 2) = 0 THEN
         PRIME = NO;
      DO R = 3 TO SQRT(N) BY 2 WHILE (PRIME = YES);
         IF MOD(N, R) = 0 THEN
            PRIME = NO;
      END;
      IF PRIME THEN
         PUT EDIT (N, ' IS A PRIME NUMBER') (SKIP, F(15), A);
      ELSE
         PUT EDIT (N, ' IS NOT A PRIME NUMBER') (SKIP, F(15), A);
   END;
END;
```

This version is slightly longer, but markedly easier to understand. And it has no duplicated PUT statement either.

Don't strain to re-use code; reorganize instead.

A faster-running program is often the by-product of clear, straightforward code. As an example, this program computes $n!$ for $n = 3, 5, ..., 49$.

```
C FACTORIAL PROGRAM
      DOUBLE PRECISION FACTOR, X
      NAMELIST /OUT/I,FACTOR
      DO 100 I =3,50,2
      FACTOR = I
      J =I-1
      DO 200 K=1,J
      X=K
  200 FACTOR = FACTOR*X
  100 WRITE (6,OUT)
      STOP
      END
```

Admittedly one does not compute a table of odd factorials very often, but this program is needlessly complicated and wasteful, because it recomputes $n!$ from scratch for each n, instead of just multiplying the previous value by $n\times(n-1)$. Here's the simpler version: