Simple tests save later grief. Does the program read a parameter to define an array size? Then test that it does not exceed the array bounds. Here is part of a sort program:

```
      DIMENSION X(300)
C     READ NUMBERS TO BE SORTED.
      READ 1,N,(X(I),I=1,N)
    1 FORMAT(I3/(F5.1))
C     INITIALIZE TO MAKE N-1 COMPARISONS ON FIRST PASS.
      K=N-1
C     INITIALIZE TO BEGIN COMPARISONS WITH THE FIRST 2 NUMBERS.
    6 J=1
C     L IS USED TO RECORD THE FACT THAT AN INTERCHANGE OCCURS.
   19 L=0
C     MAKE COMPARISONS.
      DO 2 I=J,K
      IF(X(I)-X(I+1)) 2,2,3
    3 ...
```

There are minor flaws, such as the random statement numbers (again) and the use of an arithmetic `IF` where a logical `IF` would be more readable, but we will defer discussion of the full text until Chapter 7. For now, let us concentrate on the input statement.

Suppose `N` exceeds 300. Parts of storage outside the array `X` may be overwritten. Whatever happens after that will not be good, nor will it tell the user unequivocally what he did wrong. The program may run to completion, but if the user does not look carefully at the output, he may not even notice that the program failed.

Some compilers (WATFIV, PL/I with `SUBSCRIPTRANGE` enabled, for instance) allow a check during execution that subscripts do not exceed array dimensions. This is a help, but not sufficient. First, many programmers do not use such compilers because "They're not efficient." (Presumably this means that it is vital to get the wrong answers quickly.) Second, subscript range checking will not detect the other deficiency in this code. Suppose that the value of `N` is one. Then the program compares `X(1)` with `X(2)`, which is not defined and hence garbage. If the garbage happens to be less than `X(1)`, `X(1)` is gone forever, since it is sorted out of its position.

If we write a precise description of the *exact* input data for which the sort program works, we find, of course, that it fails for `N` outside the range 2 to 300. At that point, we might be embarrassed into making it do something sensible for all values of `N`:

```
      DIMENSION X(300)
      READ 10, N
   10 FORMAT(I3)
      IF (N.LT.1 .OR. N.GT.300)    take error action
      READ 20, (X(I), I=1,N)
   20 FORMAT(F5.0)
      IF (N.EQ.1)    leave routine, since in order
      ...
```

It may be easier to redo a program than to describe *exactly* what cases it works for. In any case, writing the description should point to bugs, and to areas for improvement. (The author of this sort program came close — his second and third