In twenty-five words or less, this article says that the assessment program did its computation without checking that its input data made sense. There is a lesson here that every programmer must learn, usually the hard way (how would you like to be the programmer who wrote the assessment program?) and often several times: NEVER TRUST ANY DATA. Input prepared by people or even by other programs will contain errors. A good program tests its input for validity (the letter "P" is not a digit) and, in critical cases like tax computations, for plausibility. (An automobile assessed at seven million dollars is not plausible.)

Many introductory programming texts contain variants of the problem "Write a program to read three numbers A, B, and C, representing the sides of a triangle, and compute the area of the triangle." Here is one solution:

```
      READ (5, 23) A, B, C
   23 FORMAT (3F10.0)
      S = (A + B + C) / 2.0
      AREA = SQRT(S * (S - A) * (S - B) * (S - C))
      WRITE (6, 17) A, B, C, AREA
   17 FORMAT (1P4E16.7)
      STOP
      END
```

In most ways this is a good program: it is well-formatted, and copies its input data to the output for visual inspection. The output format is well chosen: it uses the E format to handle very large or very small answers without losing significant digits, and the scale factor 1P to print one position before the decimal point, so the answers are in the form most familiar to readers. Our only minor complaint concerns the apparently random statement numbers.

But what happens if we test the program on the "triangle" (3, 1, 1)? Most probable is an unexpected termination with a diagnostic like "negative argument in SQRT," certainly an indirect way to report that the input does not represent a triangle. (And try it on −1, −1, −1.)

You should always "launder" your input: after S has been computed, verify that no side is too big (or negative). One programmer used PL/I's built-in function ANY to test each element of an array T which contains the lengths of the three sides:

```
      S = SUM(T)/2;
IF ANY(T <=0)|ANY(T>S) THEN   it's not a triangle ...

IF ANY(T=S) THEN   it's a straight line ...
...
```

The test for a straight line may not always be reliable, for reasons we shall explore in Chapter 6, but the generally suspicious approach is commendable. This checking is easy in PL/I. In Fortran, the same tests require more code, which may explain why they are not often made.

Test input for validity and plausibility.
