As a spectacular example of what floating point errors can lead to, when compounded at a million instructions per second, let us look at these two programs, from consecutive pages of a text, each of which produces a table of natural logarithms:

```
Program 1:
  PUT EDIT ((LOG(A) DO A=1 TO 9.99 BY .01))(F(10,6));

Program 2:
  DECLARE A FIXED DECIMAL (4,2);

  DO A=1 TO 9.9 BY .1;
    PUT EDIT (A)(F(5,2));
    PUT EDIT ((LOG(A+B) DO B=0 TO .09 BY .01))(X(5),10 F(9,5));
  END;
```

We would hope that these produce identical tables, save for formatting. Since the text reproduces some of the output of these programs, let us show the values obtained for the logarithms of 3.00, 3.01, 3.02, 3.03:

```
Program 1:
  1.098563  1.101891  1.105208  1.108513  ...

Program 2:
  1.09861   1.10193   1.10525   1.10856  ...
```

A third of the way through the table, these differ by five parts in a hundred thousand, an uncomfortable error. Why are the answers so different? As a wise programmer once said, "Floating point numbers are like sandpiles: every time you move one, you lose a little sand and you pick up a little dirt." And after a few computations, things can get pretty dirty.

One of the first lessons that must be learned about floating point numbers is that tests for exact equality between two computed floating point numbers are almost certain to fail. For example,

```
C      RIGHT TRIANGLES
       LOGICAL RIGHT, DATA
       DO 1 K = 1,100
       READ (2,10) A, B, C
C     CHECK FOR NEGATIVE OR ZERO DATA
       DATA = A.GT.0. .AND. B.GT.0. .AND. C.GT.0.
       IF(.NOT.DATA) GO TO 2
C     CHECK FOR RIGHT-TRIANGLE CONDITION
       A = A**2
       B = B**2
       C = C**2
       RIGHT = A.EQ.B+C .OR. B.EQ.A+C .OR. C.EQ.A+B
1      WRITE(3,11) K, RIGHT
       CALL EXIT
C    ERROR MESSAGE
2      WRITE(1,12)
       STOP
10     FORMAT(3F10.4)
11     FORMAT(I6,L12)
12     FORMAT(11H DATA ERROR)
       END
```