```
C     TEST FOR NEGATIVE VALUE OF X.
      IF(XST) 5,5,3
      ...
    5 PRINT 11
   11 FORMAT(1H041HTHE VALUE OF X MUST BE GREATER THAN ZERO.)
```

the comment is certainly not correct, even assuming that X refers to XST. Fortunately this case is sufficiently obvious that it is not likely to mislead.

The more common situation is that the comment is correct but the code it describes is not:

```
      /* THIS TIME WE SHALL TEST FOR      */
      /* ODD NUMBERS.                     */
   IF MOD(X,2)=0 THEN
      DO; SUM = SUM + X;
      ODDNO = ODDNO + 1;
      END;
```

The comment tells us that we are testing for odd numbers, the name ODDNO encourages us to believe it, but the test still selects *even* numbers.

The trouble with comments that do not accurately reflect the code is that they may well be believed subconsciously, so the code itself is not examined critically. A programmer shaky in his understanding of the MOD function might accept this comment at face value, especially since the mnemonic identifier ODDNO provides confirmation. (To avoid this subconscious acceptance, in *The Psychology of Computer Programming* Weinberg suggests that comments should be written on the right side of the page and code on the left, so the comments can be *covered* during debugging.)

Make sure comments and code agree.

Comments should also convey new information:

```
C     NEXT TWO STATEMENTS TEST FOR XMAX, IF LESSTHAN 10**-8,GO TO 1000
C
      EPSI=1.E-8
      IF(XMAX.LE.EPSI) GO TO 1000
```

This contains the same boundary error we saw above — the branch on equality is not what the comment says it is. But even if the comment were true, it would be useless. A meaningful comment would explain the reason for the test instead of merely repeating it in words. Avoid empty remarks like

```
      K13 = K13 + 1;    /* INCREMENT COUNTER */
```

and

```
C     PRINT VALUE OF VOLTAGE
   50 WRITE(6,60) V
```

and

```
      ON ENDFILE(SYSIN) GO TO DATA_ERROR; /* TEST END-OF-FILE */
```

and