criticism and revision.

Let us conclude the chapter with another example that illustrates several failings. This program is a sorting routine.

```
      DIMENSION N(500)
      WRITE (6,6)
    6 FORMAT (1H1,26HNUMBERS IN ALGEBRAIC ORDER)
      DO 8 I=1,500
    8 READ (5,7) N(I)
    7 FORMAT (I4)
      DO 10 K=1,1999
      J=K-1000
      DO 10 I-1,500
      IF(N(I)-J)10,9,10
   10 CONTINUE
      STOP
    9 WRITE (6,95) N(I)
   95 FORMAT (1H ,I4)
      GO TO 10
      END
```

The code suffers not only from lack of generality, but from an ill-advised algorithm, some dubious coding practices, and even a typographical error. The line

```
      DO 10 I-1,500
```

is wrong: the "–" should be "=". The program was contrived in part to illustrate that the range of a DO loop can be extended by a transfer outside and back, even though in this case the inner DO loop *and* the code of the extended range can all be better written in line as

```
      DO 10 I = 1, 500
         IF (N(I) .EQ. J) WRITE (6,95) N(I)
   95       FORMAT(1X, I4)
   10 CONTINUE
```

More to the point is the question of whether programmers should be encouraged to use extended ranges in the first place. Jumping around unnecessarily in a computer program has proved to be a fruitful source of errors, and usually indicates that the programmer is not entirely in control of the code. The apparently random statement numbers in this example are often a symptom of the same disorder.

The program has other flaws. It reads in 500 numbers, one per card, and sorts them about as inefficiently as possible — by comparing each number with all integers between −999 and +999. It does this once, for only one set of numbers, then stops.

But wait. With an I4 input format, it is possible to read positive numbers as large as 9999, since we can leave out the plus sign; the program as it stands will fail to list four-digit numbers. To correct the oversight will slow the algorithm by a factor of more than five, without extending its generality in the least. Extending this method to handle larger integers would slow it by orders of magnitude, and to ask it to handle floating point numbers would be unthinkable.

We will not attempt to rewrite this code, since we disagree with its basic approach. (Chapter 7 contains several better sorting programs.) We just want to