```
      IF DISCRIM<0 THEN DO;
                          PUT EDIT('COMPLEX ROOTS')(SKIP,A);
                          PUT DATA(A,B,C);
                          GOTO MISS;
                        END;
      ROOT=SQRT(DISCRIM);
      ROOT1=(-B+ROOT)/(2*A);
      ROOT2=(-B-ROOT)/(2*A);
      PUT SKIP DATA(A,B,C,ROOT1,ROOT2);
MISS: ...
```

The actions after the DO-END are done if and only if the DO-END block is *not* done; they should be part of an ELSE:

```
      IF DISCRIM < 0 THEN DO;
         PUT EDIT ('COMPLEX ROOTS') (SKIP, A);
         PUT DATA (A, B, C);
      END;
      ELSE DO;
         ROOT = SQRT(DISCRIM);
         ROOT1 = (-B+ROOT) / (2*A);
         ROOT2 = (-B-ROOT) / (2*A);
         PUT SKIP DATA (A, B, C, ROOT1, ROOT2);
      END;
```

In Fortran, it is hard to make the structure of an IF-ELSE explicit, since there is no ELSE, and only a single (restricted) statement can follow the IF. For even more complicated combinations, things get tough indeed. Consider this fragment for keeping track of the largest and smallest A(I):

```
      IF(A(I).LE.BIG) GO TO 100
      BIG=A(I)
      GO TO 49
  100 IF(A(I).GE.SMAL) GO TO 49
      SMAL=A(I)
   49 CONTINUE
```

This is an essentially mechanical translation of the algorithm into Fortran, and as such is hard to fault. It is possible, however, to write the code rather more clearly in this special case:

```
      IF (A(I) .GT. BIG) BIG = A(I)
      IF (A(I) .LT. SMAL) SMAL = A(I)
```

If the first test succeeds, the second presumably cannot, but an occasional redundant test is a small price to pay for improved readability.

By the way, it is necessary to be quite careful when tests might overlap. Avoid situations like this one:

```
      IF HRS_WORKED<=40
          THEN CALL REGPAY;
      IF HRS_WORKED>=40
          THEN CALL OTPAY;
```

People who work exactly forty hours are rewarded with a double paycheck! An IF-ELSE divides things into two separate pieces, only *one* of which is done. It also ensures that someone reading the code can see that only one thing is done. Thus: