simple enough to write one per line. Don't let anyone tell you this is not efficient — it doesn't take all that much time to make the whole set of tests, and you're more likely to get the code right the first time. If it does take too much time, and you have measurements that prove it, then and only then should you re-write it with GOTO's.

The THEN-IF was the culprit in this example, but there is another symptom of the same problem. Note the null ELSE clause in the original, required to make the unstacking of the nested IF's come out right when one of the conditions has no corresponding action. These seemingly useless statements cauterize the stumps of any ill-thought-out THEN-IF's buried in the code. A program containing null ELSE clauses is suspect, if for no other reason than that it was written by someone burned by THEN-IF's often enough to sprinkle null ELSE's around for insurance.

The THEN-IF does have its uses. It is often the only way to ensure that tests are performed in the proper order, as in

```
IF I > 0 THEN
   IF A(I) = B(I) THEN ...
```

which checks that I is in range before it is used as an index. Some languages provide special AND's and OR's which guarantee left-to-right evaluation and early exit as soon as the truth value of the expression is determined. But if you are not fortunate enough to be able to program with these useful tools, wrap a DO-END around the inner IF so you don't have to worry about trailing ELSE's.

Avoid THEN-IF *and null* ELSE.

Consider this procedure for finding the largest of a set of positive numbers:

```
FINDNUM: PROC OPTIONS (MAIN);
         DCL NEWIN DEC FLOAT (4);
             LARGE DEC FLOAT (4) INIT (.0E1);
         /* .0 x 10**1 = .0 x 10 = 0.0                    */
NEXT_C:  GET LIST (NEWIN);
         IF NEWIN >=0
            THEN IF NEWIN > LARGE
                    THEN LARGE = NEWIN;
                    ELSE GO TO NEXT_C;
            ELSE GO TO FINISH;
         GO TO NEXT_C;
FINISH:  PUT LIST (LARGE);
         END;
```

Change the illegal semicolon into a comma in the second line. Ignore the curious zero in the INIT attribute, and the equally curious explanatory comment. Now, try to trace the flow of control. This is not a trivial exercise. How does one get to that last GO TO NEXT_C, for example? Why, from the innermost THEN clause, of course.

An ELSE GOTO tells you where you went if you didn't do the THEN, leaving you momentarily at a loss in finding the successor to the THEN clause. And when ELSE GOTO's are used one after the other, as here, the mind boggles. Needless to say,