trivial — IEMPID is not set immediately after input. A similar instance of this error appears in Chapter 5.

Disorganized code often leads to errors. One wonders if the excessively complicated structure of this program comes from an attempt to be "efficient." There are two distinct cases (overtime or not), with a different wage calculation for each. In order to test HOURS only once per employee, these cases are separated and then brought back together again incorrectly.

But if we test HOURS twice:

```
      IF (HOURS .LE. 40.0) WAGE = SRATE * HOURS
      IF (HOURS .GT. 40.0) WAGE = SRATE * 40.0 + ORATE*(HOURS-40.0)
```

indeed we do an extra test. But the program is simpler: two lines replace six, the unnecessary variables A and B disappear, and the logic flows directly from beginning to end. Not only is it now correct, but it can be seen to be correct.

Splitting the computation into two parts, only one of which is done, is preferred if each half requires a complicated calculation. But the code should still flow from top to bottom:

```
      IF (HOURS .GT. 40.0) GOTO 25
         standard calculation
         ...
         GOTO 30
   25    overtime calculation
         ...
   30 WRITE ...
```

in the standard Fortran implementation of an IF-ELSE.

Make it right before you make it faster.

The following program, which computes the bills for an electric company, is rather similar. The problem is specified as follows:

```
      usage>500         bill = 14.50 + 0.025*(usage-500)
      500>=usage>100    bill = 3.50 + 0.0275*(usage-100)
      100>=usage>50     bill = 2.00 + 0.035*(usage-50)
      else              bill = 2.00
```

In addition there is a discount for users of electric heat:

```
      if heated by electricity, and
                        usage<1000      no discount
                        1000-10000      5%
                        >10000          10%
```

The first version of the program is