```
200 FORMAT(1X, 'WEIGHT = ', F10.1, '  METAL = ', I2,
   $    '  ERROR FOUND IN CARD NUMBER ', I3)
```

The program has good points, too. It counts its data cards, instead of relying on the user. It copies the input onto the output (after computation) so it can be inspected visually. And, most important, it validates its input, testing for negative or zero weight and invalid metal code. It prints the offending card and its sequence number, then continues to the next data.

But our enthusiasm is tempered by discovering that if a negative or zero weight is encountered, the program prints the value of METAL for the *previous* input card. Label 100 is placed incorrectly, one statement too late.

If the conversion

```
METAL = XMETAL
```

were done once, immediately after the READ, the bug would vanish, and so would the need for statement 300 and the one before statement 100.

The other, more general, failing is the use of numeric codes (floating point at that) to name the metals. (Was floating point used because all the metals had names that were "floating point" variables in Fortran? What if we added LEAD, NICKEL, and IRON?) A typical line of output looks like

```
SHIPMENT NO.( 1)   METAL=  1   WEIGHT =     1000.0   COST =    2750.00
```

It is only after scanning the listing that we can deduce that metal 1 is aluminum.

The use of numeric codes is bad practice in a program that people use directly. The codes in this program are not even in alphabetical order. How can a user remember them? (As we mentioned, the textbook presents another version of this program a few pages later, which uses alphabetic names instead of numeric codes for the metals, but discards all error checking.)

Make input easy to prepare and output self-explanatory.

Here is an example that shows ill-chosen mnemonic values:

Write a program segment that will assign sales area 1 to each even salesman, and sales area 2 to each odd salesman.

```
     DIMENSION SALESM(17)
     DO 20 IJ=1,17
 10  IF(IJ/2*2-IJ) 12,11,12
 11  SALESM(IJ)=1.
     GO TO 20
 12  SALESM(IJ)=2.
 20  CONTINUE
```

The curious expression IJ/2*2-IJ is a Fortran idiom that determines whether IJ is evenly divisible by two. The original problem can be solved more clearly and succinctly with the MOD function: