```
DECLARE (A, R, M, C, P) FIXED DECIMAL(13,4);
DECLARE TRUE BIT(1) INITIAL ('1'B);

DO WHILE (TRUE);
   GET LIST (A, R, M);
   PUT SKIP(3) EDIT ('THE AMOUNT IS', A) (A, F(10,2))
      ('   THE INTEREST RATE IS', R) (A, F(6,2))
      ('   THE MONTHLY PAYMENT IS', M) (A, F(8,2));
   C = A*R/1200;
   IF C >= M THEN
      PUT SKIP(2) EDIT ('UNACCEPTABLE MONTHLY PAYMENT') (A);
   ELSE DO;
      PUT SKIP(3) EDIT
         ('MONTH', 'BALANCE', 'CHARGE', 'PAID ON PRINCIPAL')
         (X(8), A, X(6), A, X(7), A, X(3), A);
      PUT SKIP;
      DO I = 1 TO 60 WHILE (A+C >= M);
         P = M - C;
         A = A - P;
         PUT SKIP EDIT (I, A, C, P) (F(13), 3 F(13,2));
         C = A*R/1200;
      END;
      IF A+C >= 0.005 THEN
         PUT SKIP(2) EDIT ('THERE WILL BE A LAST PAYMENT OF:', A+C)
            (A, X(5), F(10,2));
   END;
END;
```

We have now combined the separate loop exits into one, which ensures that the exit conditions will all be consistent. The interest charge C is now up-to-date whenever it might be needed. And bringing the tests together at the top keeps the program from doing anything inside the loop if there is nothing to do, so silly messages are avoided.

Avoid multiple exits from loops.

Here is another example of a common error. The program is another binary search procedure to find out where in a sorted table X an element A lies. If the table contains an entry that matches A, both of the indices LOW and IHIGH should point to that value; otherwise LOW and IHIGH should be the indices of the two table elements immediately below and above the input value A. The elements of the array X are already sorted into increasing order.