```
...
I=1;
/* INITIALIZE SUM */
IN: SUM=0;
/* READ SCORES AND COMPUTE SUM */
DO J=1 BY 1 TO 5;
GET LIST (SCORE(I,J));
SUM=SUM+SCORE(I,J);
END;
/* STOP READING IF FIRST SCORE IS NEGATIVE */
IF SCORE(I,1)<0 THEN GO TO OUT;
/* COMPUTE AVERAGE */
AVG(I)=SUM/5;
/* READ NAME */
GET LIST (NAME(I));
I=I+1;
/* GET NEXT STUDENT'S SCORES */
GO TO IN;
/* COMPUTE TOTAL NUMBER OF STUDENTS */
OUT: I=I-1;
/* PRINT ... */
...
```

The comments imply that scores and names are read until the first score is negative; this indicates the end of data. Unfortunately, what comments imply is not always precisely what happens. Because GET LIST reads free-form input, this program requires *five dummy scores* to terminate the DO loop that reads scores, although only the first need be negative. If less than five are given, the program encounters the end of the input unexpectedly, before the GET LIST is satisfied. Since no ENDFILE action has been specified, the program simply terminates without printing the desired output. This is not implied by the comments.

PL/I's explicit test for end-of-file is much superior to the Fortran-like mechanism used here. Let us use it to correct the error. One possibility is

```
ON ENDFILE
   GOTO OUT;

DO I = 1 TO IMAX;  /* LOOP UNTIL EOF OR ARRAYS FULL */
   GET LIST (SCORE(I,*), NAME(I));
      should check data for validity here
   AVG(I) = SUM(SCORE(I,*))/5;
END;
   get here if arrays are full

OUT:
   output processing
```

(A * subscript repeats an operation over all legal values of that subscript. Thus GET LIST (SCORE(I,*)) reads SCORE(I,1), SCORE(I,2), etc.; SUM(SCORE(I,*)) adds them all up.) We have dispensed with the negative dummy score for terminating the input. As an added bonus, we can easily avoid reading more input than the arrays can hold, by limiting the range of the DO loop that replaces