You are to write a program that uses a two-dimensional array to store integer test scores. The classroom holds 12 desks that are bolted to the floor in a 4 by 3 matrix, that is, 4 rows with 3 seats in each row. This classroom is obviously not located at CCSF. The students are required to sit in the same assigned desk. Thus, they can be identified by their row and seat assignments.

**Requirements:**
1. Include a header comment that gives the following information:

   // Project name
   // your last name, first name
   // 110B - your section number

   // Brief statement of the program's task

2. Your program should consist of main() and three functions.

   main() declares all arrays and calls the three functions. Use two external named constants to specify the dimensions of the arrays. Initialize the scores for seat 1 in rows 1, 2, and 3 to 20, 18, and 16, respectively.

   The input function is `void` and takes the 2-d array as its argument. It enters the row and seat numbers, followed by the test score for the student sitting in that desk. If either the row or seat number is zero, the data entry stops. The row and seat numbers must be tested for values that are too large or too small before the entry of the test score.

   The processing function is `void` and takes 5 arrays as its arguments. The array arguments are the 2-d array of test scores, two 1-d arrays of counters, and two 1-d arrays of accumulators for counting and accumulating by rows and seats. The function computes the individual sums of the scores both by rows and by seats. It also counts the number of actual scores in each row and in each seat column. These sums and counts will be used by the output function to calculate the averages.

   The output function is `void` and takes the same arguments as the processing function. It produces the grade report. The averages by rows and by seats should be computed and outputted, not stored in variables. In other words, simply have the expression representing the average in a `cout`. The format of your output must be the same as that in the sample runs shown on page 2.

3. Run the program twice, using the exact same data that was used in the sample runs shown on page 2. Note that the data is shown in **bold** type.

4. Use good programming style, that is, vertically align matching braces, code nothing but a brace on a source line, use proper indentation, and use whitespace judiciously.
Run 1
Enter row seat (zeros to Quit): 1 3
Enter score: 17
Enter row seat (zeros to Quit): 4 2
Enter score: 19
Enter row seat (zeros to Quit): 3 3
Enter score: 15
Enter row seat (zeros to Quit): 3 2
Enter score: 20
Enter row seat (zeros to Quit): 0 0
Seat 1 2 3 Averages By Rows
Row 1 20 17 18.5
2 18
3 16 20 15 17.0
4 19
Averages By Seats 18.0 19.5 16.0

Run 2
Enter row seat (zeros to Quit): 5 3
Invalid row or seat.
Enter row seat (zeros to Quit): 3 3
Enter score: 25
Enter row seat (zeros to Quit): 4 4
Invalid row or seat.
Enter row seat (zeros to Quit): 1 3
Enter score: 24
Enter row seat (zeros to Quit): 0 0
Seat 1 2 3 Averages By Rows
Row 1 20 24 22.0
2 18
3 16 25 20.5
4
Averages By Seats 18.0 24.5