Homework – Chapter 5 – Bowling Scores

33. Write a complete C++ program to do the following: The main program reads in and prints three bowling scores, score1, score2, and score3. It then calls a series of functions to process these scores.

The main program calls a function validgroup() to determine if this set of three values forms a valid group. The function validgroup() receives **four parameters (the 3 scores and a reference to an ofstream)**. For the group to be valid, each number must be in the range from 0 to 300 (the possible scores in a bowling game). If the group is valid, **the function prints a message saying so**. If one or more of the numbers is negative or greater than 300, **the function prints an overall message that the group is invalid**. **In addition, for each invalid value, the function prints the score and a message**. The **function returns a signal** (say 1 or 0) indicating the validity of the group. (Hint: use six "if" statements.)

If the group is not valid, the main program skips processing and simply goes on to the next group of three values.

If the group is valid, the main program calls a function onegamescore(), sending it **two parameters, the value score1 and a reference to an ofstream**. This score is an integer from 0 to 300 (how can we be sure of this?). The function converts the score into a rating, using the following system: 250 to 300 is a professional game; 200 to 249 is an excellent game; 140 to 199 is a very good game; 100 to 139 is a good game; 50 to 99 is a poor game; below 50 is a horrible game. **The function prints a message with the original score and the bowler's rating**. Then the main program repeats this process for score2 and score3.

Next the main program calls a function avg3scores(), sending it three parameters: the three scores. The function avg3scores() finds the average (as an integer) of the three scores and sends it back. **The main program prints the average**. Finally, the main program calls onegamescore() again, sending it the resulting average from the function avg3scores().

The main program then prints three blank lines.

Then the main program goes on to the next group of three values. When the main program runs out of groups (**hint: use a sentinel**), **it prints the final values of three counters it has been keeping track of: the total number of groups processed, the number of valid groups, and the number of invalid groups**.

Note:

Output must be file directed.
Do not send prompts to the output file.
Do not use global ofstream and ifstream objects.
ofstream and ifstream objects should be declared in main() and passed to functions as needed.

Suggested Implementation:

1. Write code to read and print a group of three scores
2. Write code to continue reading and printing groups of three scores until a sentinel is reached.
3. Create an input file from which to read the groups of scores
4. Have the output go to an output file
5. Implement a counter to count the total number of groups processed
6. Implement validgroup(). The prototype for validgroup() should be: **int validgroup(int,int,int,ofstream &);**
7. Implement the counters for the number of valid groups and the number of invalid groups
8. Implement onegamescore(). The prototype for onegamescore() should be: **void onegamescore(int,ofstream &);**
9. Implement avg3scores(). The prototype for avg3scores() should be: **int avg3scores(int,int,int);**
10. Complete the program and documentation as necessary.