Project 12 (Extra Credit 30 points)
Possible points are: 30, 25, 20, 15, 10 or 0

<u>Submit Your Solution Using ANGEL by 10 PM on Friday April 26, 2013</u>

(Note: there is no late drop box for this project)

To view the order and style of the input and output information for the project, run the <u>provided solution</u> by **<u>typing the following at a terminal window prompt.</u>** (**Note**: Input files must be present in the directory that the terminal window is currently in.)

Sample Solution – for determining vertical spacing differences

/home/work/cpe112/Executables/Project_12/Project_12_solution

Sample input files are provided in P12_in.zip

Comparison Script – for determining line differences

/home/work/cpe112data/Project_12/CompareSolution.bash Project_12.cpp

<u><Project 12 Restrictions></u>

On Project 12, you may only use any concepts presented in Chapters 1 - 11 of your textbook. ***<u>You are not allowed to use global variables.</u>*** If necessary, global constants may be used. Global constants and variables are declared above the int main() line in a program.

<u><Starting Project 12></u>

- Open a terminal window and move (cd) into the **Project_12** directory created in the **CPE112_SPR13** directory (This is the directory structure created in project 1.) The command for this is: **cd CPE112_SPR13/Project_12 (typing cd ~/CPE112_SPR13/Project_12 works as well)**. You will need to modify the names as necessary to match your capitalization style for the two directories.

- Download all needed files from ANGEL into this directory. Once you have finished with the program and it compiles without syntax errors, run the executable and verify that the output for your program matches the output from the provided solution executable

- **Once you are satisfied with your solution, submit Program_12.cpp via Angel**.

NOTE: make sure that you do not change the order in which the information is entered. An automatic script is used to process all lab submissions, and if the order of the input information is modified, the script will not work properly with your program.

<u>NOTE: Output of your program shall match the output of the sample solution. This match includes all information written to the terminal by the program</u>

<Project 12 Description>

You will write a program that performs the following tasks:

(1) Open a user-specified file for input. Prompt for the name of the file, read it into a string variable, echo print it to the terminal and then open it. If the file fails to open, enter into a loop that prints out an error message, resets the input file stream variable **(see the hints section),** obtains a new file name and tries to open the new file. The loop continues until the user successfully enters a valid file name or presses ctrl-c to exit

(2) If the input file is empty, your program should indicate that the input file was empty as shown by the sample solution – note for an empty input file, the empty input file message only is printed out. **(see the hints section on how to perform this test)**

(3) The input file contains the test scores for several students (one student per line). For each student in the file, the program reads a student's last name, first name, and several test scores. The end of the test scores is reached when a test score is negative (sentinel controlled loop). **See the hints section and look at the input files.**

(4) For each student, print a summary of the student's information:
   a. Up to 10 characters (use substring function) of the last name,
   b. Up to 5 characters (use substring function) of the first name,
   c. The average of the student's test scores, and
   d. The corresponding letter grade for the test score average

   as shown in the sample solution. **Print out all test scores and averages with a precision of 2 decimal places.** As the student information is being read, keep track of the highest and lowest test score average for all students in the class.

(5) After the information for all students has been read and processed, compute and output the class average, the highest test score average and the lowest test score average. These are output left justified in a field size of 21 with a period for a fill character.

(6) Lastly, output a histogram summarizing overall class performance as number of A, B, C, D, and F grades (use a standard 10% scale, i.e >= 90 ➔ A, >=80 && < 90 ➔ B, etc). If three students have received a grade of A, then print three stars. Remember, the histogram is not displayed if the input file is empty.

Possible test scores range from 0 to 100, inclusive. If an input file is not empty, you may assume that any data that appears within that file is valid data. Valid files will have at least one student, and that student will have at least one exam score. All students in a valid file have at least one test score.

<Project 12 C++ Hints>

1. Resetting the input stream variable: In Dev C++ and g++, the clear function must be used on a file stream that is repeatedly reopened in a loop. Therefore, use the following statement in your while loop that executes until a valid filename is entered:

   Input_file_stream_var.clear();

   Where **Input_file_stream_var** is the name of your input file stream variable

2. Testing for an empty input file is performed by using a priming read, and then testing the status of the file stream after the priming read – before the while loop is entered. In this case, if the file stream is in the fail state after the priming read, print out an appropriate message and terminate the program. Otherwise proceed to the loop to process and read the rest of the input file

3. Use a while loop to read and process all information in the input file – this will be an outer loop

4. Use a while loop to read and process the test scores for a student – this will be a loop inside the outer loop. It will be sentinel controlled terminating on a test score being negative. Don't forget the priming read for this inner loop

5. Assume that all non-empty input files have valid data only. Each student in a valid file has the same number of test scores and that all students in a valid file have at least one test score. The number of test scores each student has will vary from input file to input file.

6. Summing the test average for each student and then dividing that sum by the number of students in the class determines the class average.

7. The high and low test average scores are found inside the outer loop

8. The outer loop reads in a student's name, the inner loop reads the student test scores and determines the test average for that student. The test average is used for determining the student's grade and if it is the highest or lowest test average.

9. An If-then-else-if statement is best used to process a student's test average to update the counters keeping track of the number of A's, B's, etc.

10. Procedure for processing the information in a file is to read in the information for a single student, process it and then read in information for the next student

11. Use priming reads for each loop.

12. The following lines of code set up the header for the output:

```
cout << fixed << showpoint << setprecision(2) << left;

cout << endl << setw(3) << '#' << setw(12) << "Last" << setw(7)
     << "First" << setw(9) << "Average" << setw(6) << "Grade" << endl;

cout << setw(3) << '-' << setw(12) << "----" << setw(7)
     << "-----" << setw(9) << "-------" << setw(6) << "-----" << endl;
```

13. The following line of code outputs the information necessary. Remember, you will have to change the variables shown to match those in your program

```
cout << setw(3) << NumStu << setw(12) << last.substr(#,#)
     << setw(7) << first.substr(#,#) << setw(9) << avg << setw(6) << letter;
```

Where NumStu is the number of students processed so far. It provides the count under the # sign of the heading. The substring functions also contain # signs. These # signs have to be replaced by appropriate numbers to output the information required. In the output statement letter is the letter grade for the test average. **Note: the last item output is in a field width of 6. Because of left justification being used, the field width specification is not necessary; however, the solution uses it and so should your solution.**

14. Use loops (for or while) to print out the histogram results for each letter grade.

15. The following code sets up the histogram title:

```
cout << right << setw(18) << "Grade Histogram"  << endl;
cout << "           1         2" << endl;
cout << "  12345678901234567890" << endl;
```

16. Output information for each student as they are processed – output statements occur within the outer loop. The histogram appears after the outer loop has finished.