Project #05 (20 points): File Input/Output

➔10% grading bonus if your program output matches the sample solution exactly 🡐

<u>Submit Your Solution Using Canvas by Noon on Friday, 09/21/18</u>
(late submissions will be accepted from Noon to 2pm on 09/21/18)

<u>Obtaining Project 5 Input Order and Output Requirements</u>: (Sample Solution and Comparison Script)

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 path

/home/work/cpe211/Executables/Project_05/Project_05_solution

Provided sample input files are zipped in the file P5_in.zip

When running this solution you must provide command line arguments for the input and output files(see page 3)

Comparison Script Path

/home/work/cpe211data/Project_05/CompareSolution.bash Project_05.cpp

<u>Project 5 Restrictions</u>

Only material from Chapters 1 through 5 and any extra code in this handout is allowed.
You cannot use any C++ techniques that are covered in Chapters 6 and higher.
No Loops Allowed

<u>You are not allowed to use any global variables, Loops are not allowed.</u>
Global variables are declared above the int main() line in a program.

<u>Starting Project 5:</u>

- Open a terminal window and move (cd) into the Project_05 directory created in the CPE211_FALL18 directory (This is the directory structure created in project 1.) The command for this is: **cd CPE211_FALL18/Project_05 (typing cd ~/CPE211_FALL18/Project_05 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 Canvas into this directory. Using your favorite text editor (i.e. gedit), open the existing **Project_05.cpp** file (this is the header file created in Project 2) and add code to it to complete this project. 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

- Remember to use the firefox browser when viewing Canvas. Also, to view a pdf file saved in a directory, type the command **evince filename.pdf** at the prompt in a terminal window that has the same working directory as the directory in which the file is present.

- **Once you are satisfied with your solution, submit Project_05.cpp via Canvas**.

<u>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>

<Project 5 Description>

For this project, you will write a complete C++ program that performs ***all of the following tasks***. **Use variables of Data Type *float* to contain all numerical values**.

(1) Using command line arguments for an input file and output file name (**see command line argument slides and page 3**), open the input file provided as the first command line argument. Verify that the file opened successfully. If it did not open successfully, output an error message and terminate the program. – **See page 4 for information on this test**.

(2) If the input file is successfully opened, open the output file provided as the second command line argument. Verify that the output file opened successfully. If it did not open successfully, output an error message and terminate. Use the filename "**`Bad/file`**" to cause the open function to fail for the output file.

(3) For steps 1 and 2 output a statement stating the name of the file being opened-see the solution

(4) Read in the title line from the input file and write it to the output file.

(5) Write the column headings to the output file (see the sample solution output)

(6) For the first person, read in their first name, last name and four test scores. (check the project 5 input file information on page 3)

(7) Find the average of the four test scores.

(8) Write the information for the first person to the output file. Output the first 9 characters of the last name, the first 10 characters of the first name, the test average of the person and the letter grade for the test average (90,80,70,60 split for A, B, C, D and F) – use an if-then-else-if statement to process the average

(9) Repeat steps 6, 7 and 8 for the second and third person information in the input file – no loops

(10) Output of the program shall match that of the sample solution

<Project 5 Hints>

- Make sure output is to the output file – including the output statements that set up the formatting.
- Use the ignore function (See page 3) to remove the new line character from the input stream (which is the input file specified) when a getline is used after an extraction operation.
- **The name columns are in a field width of 12 and left justified. The average column is output in a field width of 9, and the letter grade is just output (no setw used for it)**
- Print the comment line and the headers to the output file before processing the first person
- One set of variables only are needed to hold the first name, last name, four test scores and average. Do all calculations and outputs for one person and then copy the code for use on the second and third persons. DO NOT USE LOOPS
- Output the first 9 characters of the last name and the first 10 characters of the first name – use the substring function
- Run the sample solution (not the comparison script) to see what the program writes to the terminal and to the output file, and the order in which the information is written.
- Look at the contents of an input file (see page 3) to see what has to be read. Note the use of commas to separate name parts.
- Loops are not allowed on this project. Write the code necessary to process one student and then copy and paste it twice to process the next two students. Just remember to use ignore statements following extraction operations to remove the new line character.

<Project 5 Input File Format>

The Input file format is similar to the information shown below. Look at all of the provided input files to determine the exact format.

- **On all lines, a comma separates the last name from the first name and the scores from the last name. All scores are separated by spaces.**
- **The first line of the input file is a header line which is to be copied to the output file**

Note, use the getline function terminated on a comma to read in the first and last name of the person. Use the extraction operator to read the scores. Remember that first and last names can consist of multiple words.

```
// Class Roll For CPE211
Ron,Bowman, 100.75 100.3 100 99.43
Johnny B.,Quick, 54 53 52 51
Mary,Francis Beacon, 75 96 32 68
```

<Project 5 ignore function>

Use the following format for the ignore function to ignore past the new line character in the input stream**. INT_MAX is defined in the header file climits.**

```
inputFileStream.ignore(INT_MAX, ‘\n’);
```

<Project 5 Error Messages>

The program is to have error messages for the following conditions. Each of these tests results in termination of the program:

- Two file open error message: one for the input file and one for the output file
    - o 15 *’s on each side of title, 47 across the bottom

Run the sample solution to see the various error messages. See page 4 for more details.

<Project 5 Command Line Arguments>

Be sure to look at the command line argument slides
For this program, instead of int main() use: `int main(int argc, char *argv[ ])`

To run your program, type the following at the command line:

```
./Project_05  input_file_name  output_file_name
```

In your program, assign argv[1] to a string variable **(i.e. inFileName=argv[1];)** that is to hold the input file name and argv[2] to a string variable**(i.e. outFileName = argv[2])** that is to hold the output file name. Use these string variables in the open function when opening the files.

When running the sample solution (not the comparison script), you must provide the input and output file names as command line arguments.

< Testing the State of the File Stream>

The following code is used to test the status of a file stream. **Replace the file stream variable indicated (your_file_stream)** with the actual name of the file stream used in your program. Also**, replace the file name string variable (filename)** with the actual name of the variable that holds the name of the file entered by the user.

```
if(your_file_stream.fail())
{
  cout << string(15,’*’) << " File Open Error "
       << string(15,’*’) << endl;
  cout << "==> Input file failed to open properly!!\n";
  cout << "==> Attempted to open file: " << filename << endl;
  cout << "==> Terminating program!!!\n";
  cout << string(47,’*’) << endl;
  return 1;
}
```

Output file open error message is the same as above – just change the wording to reflect output versus input.