# Programming Assignment: Final Project

# **Determine Letter Grades**

5208 C Programming for Beginners

Instructor: Bineet Sharma

**Summary:** Write a program that determines letter grades for students in a class, using your understanding of control flow, arrays (and/or pointers, structures), disk files, and error conditions in C.

**Description:** Your program accepts two command line arguments. The first argument is the name of a disk file that contains the names of students, and their test scores, separated by commas followed by one or more spaces. Each line in the file will contain scores for one student. The second argument to your program is the name of an output disk file. Your program creates a new output disk file using that name. You will write grade information of all students whose information was read from input file, in a sorted order into this output file. You will be writing one line in the output file for one student. You will write name of one student and the letter grade he/she got in the class in each line (you should format the texts in the output file). The format of the data in the input file is fixed, however the number of students in the input file is unknown during compile time. The name of input and output files could be anything and only known during run time. Besides writing to the output file, you will also need to display the averages of scores along with minimum and maximum scores for each test in the screen/console.

*Calculation:* The test scores are weighed. There are four quizzes, 40% total, midterm I is 20%, midterm II is 15% and the final is 25%. All scores in the input file are recorded out of 100. You need to apply the weight for each score in the program to calculate the final score. The final score is tabulated as follows:

Final Score = quiz1 * .10 + quiz2 * .10 + quiz3 * .10 + quiz4 * .10 + midi * .20 + midii * .15 + final * .25
Determination of letter grade is according to the following logic:

Final Score >= 90% then letter grade is A, 80%-89% B, 70%-79% C, 60-69% D, <= 59% F

**Sample input data file: input_data.txt** (the input data format is: name, quiz1, quiz2, quiz3, quiz4, midi, midii, final – Assume that data will be correctly formatted as described.), e.g.:

```
Thui Bhu,           100, 90, 80, 100, 89, 99, 88
Ariana B. Smith,    90, 90, 100, 100, 99, 100, 95
Emily Gonzales,     100, 90, 100, 70, 78, 78, 80
Jennifer L,         80, 90, 90, 100, 89, 99, 85
Maria Jones,        65, 72, 77, 68, 62, 70, 65
Bill Gates,         60, 54, 38, 62, 65, 60, 50
Escobar Morris,     83, 77, 88, 76, 79, 72, 76
Anne Latner,        80, 80, 85, 95, 90, 95, 90
..
<<more if there are more students>>
```

**Sample output file: output_data.txt** (the output format is: Name: letter grade, sorted by name), e.g.

```
Letter grade for 8 students given in input_data.txt file is:

Anne Latner:                B
Ariana B. Smith:            A
Bill Gates:                 F
Emily Gonzales:             B
Escobar Morris:             C
Jennifer L:                 B
Maria Jones:                D
Thui Bhu:                   A
…
<<more if there are more students>>
```

**Sample Run of the program (**name of your program is **Lettergrader):** Remember that there are two sets of outputs. Letter grade is written in the output disk file (which is not shown in the screen), only score averages are displayed on the console (and are not written in the disk file).

**Example Run 1:**

C:>lettergrader input_data.txt output_data.txt

*Letter grade has been calculated for students listed in input file input_data.txt and written to output file output_data.txt*

*Here is the class averages:*

| | Q1 | Q2 | Q3 | Q4 | MidI | MidII | Final |
|---|---|---|---|---|---|---|---|
| *Average:* | *82.25* | *80.38* | *82.25* | *83.88* | *81.38* | *84.13* | *78.63* |
| *Minimum:* | *60* | *54* | *38* | *62* | *62* | *60* | *50* |
| *Maximum:* | *100* | *90* | *100* | *100* | *99* | *100* | *95* |

*Press any key to continue . . .*

C:>_

**Sample Run 2:**

C:>lettergrader data1.txt data2.txt

*Letter grade has been calculated for students listed in the input file data1.txt and outputted in the output file data2.txt.*

*Here is the class averages:*

| | *Q1* | *Q2* | *Q3* | *Q4* | *Mid1* | *Mid2* | *Final* |
|---|---|---|---|---|---|---|---|
| *Average:* | *82.25* | *80.38* | *82.25* | *83.88* | *81.38* | *84.13* | *78.63* |
| *Minimum:* | *60* | *54* | *38* | *62* | *62* | *60* | *50* |
| *Maximum:* | *100* | *90* | *100* | *100* | *99* | *100* | *95* |

*Press any key to continue . . .*

C:>_

**Note:** You will be able to run this application with your choice of IDE as well, including Visual Studio by providing command line arguments in the settings. All IDE allow you to enter the command line arguments.

**Score:** Maximum score you can receive is 100 and based on the following criteria:

1) The program compiles, runs, and provides the correct answers (70%)
2) The program design is modular which includes good choices of functions and data structures. (20%)
3) The program has properly formatted output, and error checking (10%)

**Design Hint:** You are free to use your own design however; here is a guide to make it modular design:

- Storing data into memory using suitable data structures will help you write code which will be modular. You then need to calculate the averages by going through all the data for each test score for all the students and divide by the number of students (you will only know the number of students during run time).
- You are free to use any data structures you want. Parallel arrays for all the scores is one easy way (for simplicity you can assume maximum number of students to be 100). Alternatively, you can create an array of structures to store data, one structure will hold one student's all the scores and name.
- If you want to explore pointers, then a linked list would be the best choice. This is especially helpful, as you do not know how many student records are in the input file until you have read all the data in it.
- I will not penalize you for choosing one data structure vs. another, or if you assume 100 to be the maximum number of students, as long as you make sure to calculate the average using the

actual number of students during run time (you need to keep a running total in that case). Make a decision on your own regarding data structures; that in itself is good experience.

- You need to choose (or write your own) sorting algorithm to sort the data so that you can display in sorted order by name. Also, you need to write proper functions to give you minimum, maximum, and the average scores in each class.

**Submission requirement:** Submit your source code (a *.c file) through UCSC web portal (as an attachment) before the deadline. You can write your application spread out in multiple .c and .h files (remember that there is only one main in any application, even if you have multiple .c and .h files for that application). Follow this naming convention for your files: If functions are written in a separate file then give it a name: **utility*_johndoe*.c** file and then write a driver program as **testlettergrader*_johndoe*.c** file which uses functions from utility file, by replacing **johndoe** with your name.

I do NOT need your test data files I will write my own data and will compile and run your code using my data file with different names. Your code needs to handle the sample data shown above. You can be certain that the data file will not have junk and it will have correct data.