



# CSCI 2270 – Data Structures Summer 2020

*Instructors: Ashraf, Godley*

## Assignment 0

Due Sunday, June 7, 11:59PM

## C++ FUNDAMENTALS

### OBJECTIVES

1. Read-in command line arguments
2. Read a file
3. Loop through an array
4. Create an array of struct types
5. Read and Write to a file

### Instructions

Write code to complete the problem below. To receive credit for your code, you will need to paste your solution into **Moodle's autograder (CodeRunner)**. Your code needs to compile and pass the given test cases in order to receive points. Note that the autograder submissions are divided into separate parts, so that you will be required to test the individual functions prior to testing the entire program. You must also zip your code and attach it to this week's Dropbox via Moodle.

### Problem

**Overview:** In this question, you will write a program that:

1. Reads a ".csv" file with up to 10 lines and 5 columns containing information on student grades.
2. Stores the information in an array of structs.
3. Writes the lines into the **output.csv** file where the overall student grade is an A.
4. Prints the content of the entire array.

### **Specifics:**

Create an array that holds the **studentData struct objects**. Use the following struct declaration:

```
struct studentData {  
    string studentName;  
    int homework;  
    int recitation;  
    int quiz;  
    int exam;  
    double average;  
};
```



# CSCI 2270 – Data Structures Summer 2020

*Instructors: Ashraf, Godley*

## 2A. Write a function named **addStudent**:

- i. The **addStudentData** function has the following declaration:

```
// length: Number of items currently stored in the array
void addStudentData(studentData students[], string studentName, int homework, int
recitation, int quiz, int exam, int length);
```

- ii. Instantiate a struct and store the **studentName**, **homework**, **recitation**, **quiz**, and **exam** values in it.
- iii. Assume equal weights amongst everything that needs to be graded. Take the average of the homework, recitation, quiz, and exam for each respective student and store it in the struct.
  - a.  $\text{student.average} = (\text{homework} + \text{recitation} + \text{quiz} + \text{exam}) / 4;$
- iv. Add the struct to the **students** array.

## 2B. Write a function named **calcLetter**:

- i. The **calcLetter** function has the following declaration:

```
char calcLetter(double avg);
```

- ii. Write IF-ELSE conditions to assign letter grades based on the following
  - a.  $>90 = A$
  - b.  $80-89.9 = B$
  - c.  $70-79.9 = C$
  - d.  $60-69.9 = D$
  - e.  $<60 = F$
- iii. Return the letter grade.

## 2C. Write a function named **printList**:

- i. The **printList** function has the following signature:

```
// length: Number of items in the array
void printList(const studentData students[], int length);
```

- ii. Loop through the **students** array.
- iii. Print out each element of the **student** array in the following format.  
 $\text{<student\_name> earned <student\_average> which is a(n) <letter\_grade>}$   
`cout << students.studentName << " earned " << students.average << " which is a(n): " << calcLetter(students[index].average) << endl;`

### Example:

"John Doe earned 95 which is an A"



# CSCI 2270 – Data Structures Summer 2020

*Instructors: Ashraf, Godley*

2D. Write a **complete program** which includes the following:

- I. The **struct** and **functions** listed above.
- II. A **main()** function defined as below:
  1. Your **main()** should handle **four command line arguments**
    1. the name of the **input “.csv”** file
    2. the name of the **output “.csv”** file
    3. a low bound letter grade
    4. an upper bound letter grade
  2. Input and output files need to be stored in the same directory as your program.
  3. Read from the input file, **“students.csv”**:
    - a. Each line of the file can be read using **getline** function.
    - b. Parse each line using **stringstream** and convert each entry into its appropriate data type. **Keep in mind that you must use the exact data types that are specified in the struct.** (*Hint: If needed, use **stoi**, **stof** functions to convert from strings to numbers*)
    - c. Call **addStudentData** function to update the **students** array.
  4. Call the **printList** function after the array has been filled with data.
  5. Writing out to external files in C++ can be done very similarly as reading in. Instead of using an object of the *ifstream* class, as is done with input streams, use an object of the *ofstream* class. A “csv” stands for comma separated values. Write into **output “.csv”** file:
    - a. Write the <student\_name>, <student\_average>, <letter\_grade> of the students whose <letter\_grade> is >= **lower\_bound** and <letter\_grade> <= **upper\_bound** (read from command line) into the **output “.csv”** file.
    - b. **You should not sort them while writing to output file.** The relative order will be same as input file. Only those students with overall letter grade within the bounds should be written to the output file.
  6. Make sure your program closes the output file when it is finished writing.

**Check next page for sample input and output.**



# CSCI 2270 – Data Structures Summer 2020

*Instructors: Ashraf, Godley*

## Sample Input and Output:

### Test case 1:

#### File Contents: students.csv

Tim Thomas, 90, 92, 95, 100

Bob Brown, 80, 90, 88, 81

Jenny Jackson, 69, 79, 90, 73

Samantha Smith, 65, 57, 72, 59

Ralph Rogers, 55, 59, 62, 48

### Your print Output:

Tim Thomas earned 94.25 which is an A

Bob Brown earned 84.75 which is an B

Jenny Jackson earned 77.75 which is an C

Samantha Smith earned 63.25 which is an D

Ralph Rogers earned 56 which is an F

### Your output.csv file with area between C and A should contain the following:

Tim Thomas, 94.25, A

Bob Brown, 84.75, B

Jenny Jackson, 77.75, C