CSC 1300-001 Programming Assignment 4: Processing Student Information for a Class using Structures

Objective

The major objective of this programming assignment is to demonstrate mastery of the following topics:

- Commenting in C++
- Formatting code in a readable way
- Variables and data types
- Branching
- Loops
- Functions
- Arrays
- Dynamic Memory Allocation
- Pointers
- File I/O
- Structure

Description

The program will take the name of a text file as the command line argument from the user. A sample text file "input.txt" is given for experimentations. The first line of the text file states how many students' information are there in the text file. In other words, it states the size of the dynamic array to structure (named "Student") that you are expected to create. The size could be any value between 1 to 1,000. Then, the following number of lines will contain four pieces of information per student, which include "full name", "quiz marks", "midterm marks", and "final term marks". Each student's full name is a string that contains two parts: first name and last name – a space separating both parts. The marks are in double formatted.

The structure, defined as "Student", has several members: "full name", "tech email", "quiz marks", "midterm marks", "final term marks", "total marks", and "letter grade". You are expected to generate the tech email based on the respective student's name. For example, if the name of the student is "John Doe", then his email is going to be "JDoe@tntech.edu" (where we'll take the first character of first name and the entire part of the last name, followed by @tntech.edu). In addition, you are required to compute the "total marks",

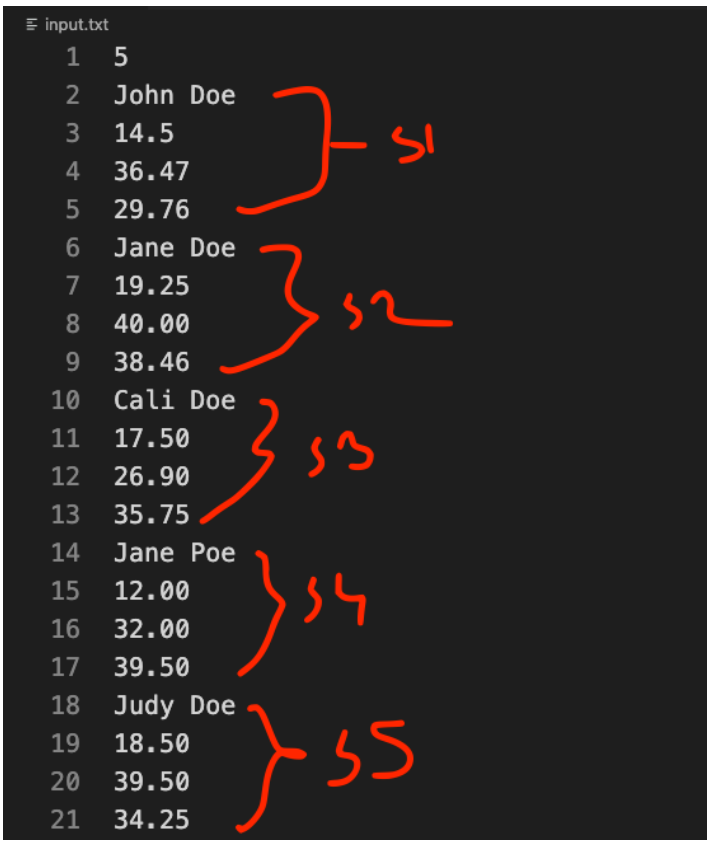
which is the summation of “quiz marks”, “midterm marks”, and “final term marks”. Then, based on the computed total marks, you are expected to derive the letter grade for each student. The letter grade distribution is as follows –

- 90 – 100: A
- 80 – 89: B
- 70 – 79: C
- 60 – 69: D
- 50 – 59: E
- 0 – 49: F

All the aforementioned tasks should be done as functions that are defined for you in “utils.h” file. Please implement the functions in “utils.cpp” file. Having done that, you are going to perform the following tasks: (1) display the students’ information, (2) display the student’s information who has done the best in quiz, (3) display the student’s information who has done the best in midterm, (4) display the student’s information who has done the best in final term, (5) display the student’s information who has done the best in total marks, and (6) display the class average for quiz, midterm, final term, and total marks obtained. All these tasks are also to be done in functions, which are defined for you in “reports.h” file. Please implement the function in “reports.cpp” file.

The following is a walkthrough of what I expect your program to do after execution –

1. Let's assume the input file contains the following values –



The first line states the number of students' information present in the text file (ranging from 1 to 1,000), which is 5 as per the above example. Then, we can expect that the following 4 line contains the 1st student's information, the next 4 is for the 2nd student, …, the last 4 is for the 5th student's information. That being said, the first line out of 4 lines of information per student is designated to show the full name, the second line is for quiz marks (ranging from 0 to 20), the third line is for midterm marks (ranging from 0 to 40), and the last line is for final term marks (ranging from 0 to 40).

2. After the compilation of your program, the executable file needs to be executed along with the name of the file, which is "input.txt" as per for the above example.

```
$ g++ -o program4 main.cpp utils.cpp fileio.cpp reports.cpp
$ ./program4 input.txt
```

Another way to compile the program is as follows (which makes sure to combine all the CPP files inside the working directory or folder) –

```
$ g++ -o program4 *.cpp
```

3. Then, the program will generate the output somewhat like follows by performing the aforementioned tasks in any order –

```
(base) ahsans-mbp:P4 ahsanayub$ g++ -o program4 *.cpp
(base) ahsans-mbp:P4 ahsanayub$ ./program4 input.txt
Student Information
Name: John Doe
Email: JDoe@tntech.edu
Quiz: 14.5
Midterm: 36.47
Final Term: 29.76
Total: 80.73
Grade: B
Name: Jane Doe
Email: JDoe@tntech.edu
Quiz: 19.25
Midterm: 40
Final Term: 38.46
Total: 97.71
Grade: A
Name: Cali Doe
Email: CDoe@tntech.edu
Quiz: 17.5
Midterm: 26.9
Final Term: 35.75
Total: 80.15
Grade: B
Name: Jane Poe
Email: JPoe@tntech.edu
Quiz: 12
Midterm: 32
Final Term: 39.5
Total: 83.5
Grade: B
Name: Judy Doe
Email: JDoe@tntech.edu
Quiz: 18.5
Midterm: 39.5
Final Term: 34.25
Total: 92.25
Grade: A

Best performance in Quiz
Name: John Doe
Email: JDoe@tntech.edu
Quiz Marks: 14.5

Best performance in Midterm
Name: John Doe
Email: JDoe@tntech.edu
Midterm Marks: 36.47

Best performance in Final Term
Name: John Doe
Email: JDoe@tntech.edu
Final Term Marks: 29.76

Best performance in Total Marks
Name: John Doe
Email: JDoe@tntech.edu
Total Marks: 80.73

Class Average
Quiz Avg: 16
Midterm Avg: 34
Final Term Avg: 35
Total Avg: 86
```

Development Instructions

You are given three files in a compressed folder that I'd like you to utilize to perform all the tasks. The files are "main.cpp", "utils.cpp", "utils.h", "fileio.h", "fileio.cpp", "reports.h", and "reports.cpp". All the documentation of how each function is supposed to work is given in the header files. Please don't modify any content on the header files; however, you will see the function prototypes / declarations on there that I want you to implement in all the CPP files. Only the "main.cpp" file should include the main function (from where the program will start its execution), whereas the other CPP files shouldn't include the main function – only the definition of each function that is declared in its header file.

Please use the "main.cpp" file to invoke all the function that you've implemented to perform the tasks. Please see the recording of the class when the program is going to be released to get more insight on the development instructions. The link will be posted on the MS Teams > Discussion – Programming Assignment channel for your quick access.

Constraints:

- The size of the list of items will any positive number between 1 and 1,000.
- 0 <= Quiz marks <= 20
- 0 <= Midterm marks <= 40
- 0 <= Final term marks <= 40
- "Full name" of a student will always has a space character to distinguish between the first name and the last name.
- Please don't worry about the duplicate tech emails. For example, the tech emails for both "John Doe" and "Jane Doe" will become "JDoe@tntech.edu".
- Please modify the input text file with different input cases and validate the program on your own. As the evaluator will be running your program with a varieties of test cases to inspect the robustness of your implementation.

Things to Note

- I ask you to put comments throughout your code. Try not to exceed 80 characters per line.
- Declare the ownership of your program at the top of your code.
- **Submission guideline**: Please place all the files for your program along with "readme.txt" in a folder named "Program4_<your_Tech_user_name>" (e.g., "Program4_mayub42"). Then, please zip the folder and submit it on iLearn before the deadline.
- The submission deadline is on **Thursday, December 2, 2021 at 11:59 PM (Central Time)**. *Please note that no late submission will be accepted*. I plan on finalizing all the grades but the final term exam by the end of December 6th week.

How Your Submission Might Be Graded

Points Possible	Trait	Exceptional	Acceptable	Amateur	Unsatisfactory
25	**Compiling / Syntax Errors**	Compiles with no syntax errors.			Does not compile.
60	**Specifications**	The program runs without errors and meets all of the specifications.	The program produces the correct results and displays them as specified. It also meets most of the other specifications.	The program produces correct results but does not display them correctly. OR The program displays results perfectly but results are not all correct.	The program is producing incorrect results and has display issues.
15	**Readability**	The code is exceptionally well organized and very easy to follow.	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.
-	**Delivery**	The program was delivered on time.	Subtracting 5 points off per day late. Will only be allowed to be turned in up to two days late. May not be submitted via email.		