The University of the South Pacific

School of Information Technology, Engineering, Mathematics and Physics

CS111: Introduction to Computing Science

Assignment 2: Sem 1 2024

Learning Outcome:

1. Apply programming concepts to computing problems
2. Examine code for its syntax and semantic validity

Weight: 7.5%
Due Date: 2 June 2024: 11.55pm (Fiji Time).

Part 1: Assignment Rules and Submission Instruction

1.1 Assignment Teams.

This is a group assignment where you will work in pairs. At most **two** students can work together. You have to find a partner for the assignment yourself. You can seek help from your tutors in the lab if you need assistance in this. Or use Moodle forums to find your assignment partner.
This requirement is there for you to demonstrate your ability to work in a team. Penalties will apply if an individual assignment is submitted and/or groups sizes are larger than 2. [For group sizes n larger than 2 a deduction of (n − 2) /n will apply. This means, if you end up for some reason in a group of four you will be penalized with a 50% deduction]

Please fill in your team members' names in the spreadsheet provided in Moodle.

1.2 Submit before the Deadline

You have to submit your work before the deadline 2 June 2024: 11.55pm (Fiji Time).
Please start EARLY. Also when you finish don't hold on to your assignment till the last moment. Submit as soon as you are satisfied with it.

1.2.1 Submitting late:

If you think that you will not be able to submit the assignment on time due to circumstances which are beyond your control, then you will need to seek approval from the course coordinator prior to the due date.
This means you have to ask for an extension, explaining the reasons, before the deadline, and not after. A late submission will be penalized by 20% for each 24 hour period it is late. Where possible, it is better

to hand in your work early and get credit for partial work than handing in late. A partial work may earn more points than a working assignment which is submitted late.

1.3 Submission

- You will have to submit a program for this Assignment.
- **Only one member of the team has to submit the assignment**.

Note:
Be sure you submit .cpp files only. Name your program file as Assign1_YourIdNumber_YourPartnersIDNumber.cpp, where YourIdNumber is your student id number and YourPartnersIDNumber is your partners ID number (e.g. Assign1_S01234561_S12345687.cpp).

Do not submit the .exe executable versions of your program. Also be sure that you submit the assignment only once you are totally satisfied. We will not accept “corrected versions” submitted through other channels after the due date. However, you can submit the assignment as many times as you like on Moodle before the due date!

1.4 Plagiarism

For this and other work in CS111, it is essential that you avoid plagiarism. Not only do you expose yourself to possibly serious disciplinary consequences, but you will also cheat yourself of a proper understanding of the concepts emphasized in the project. You will almost certainly fail the short tests and/or the final which will test your understanding of the project. It is not plagiarism to discuss the assignment with your friends and consider solutions to the problems together. However, it is plagiarism for you to copy all or part of each other’s programs. If you find somebody has stolen your assignment and produced it as their own, it will be considered plagiarism. We are using automated tools to assist with the detection of plagiarism.

They will highlight any unusual code that is similar between students. All cases that are flagged as potential plagiarism will be checked by hand. So do not leave your flash drives around. Be careful with them. And make sure you log out of the lab machines when you are finished working with them. Don’t copy part of someone solutions, and do not give somebody else an electronic copy of your solution. If you give someone your program it is almost guaranteed that part of it will end up in their program, no matter whether you ask them not to copy.

Any student posting code for this project on Moodle will be considered to be committing plagiarism. Do not submit your code to any discussion group or mail it to anyone except the lecturers or tutors.

Do not be concerned if Moodle tells you that a file cannot be checked by TurnItIn for plagiarism. This will be your program file. We will check programs separately once submitted.

1.5 Support

If you have a problem with the assignment, with C++, Dev C++, the exercises, and questions in this assignment, lecture notes, the book, please ask. First use the forum for your questions. This will also help other students with the same questions. Then ask your tutor for help, or otherwise the course coordinator.

If you have any problems with the assignment or the course, feel free to email, visit or call the course coordinator. All contact information is on Moodle.

Part 2: Programming – Student Marks Processing (50 marks)

2.1 Problem Statement

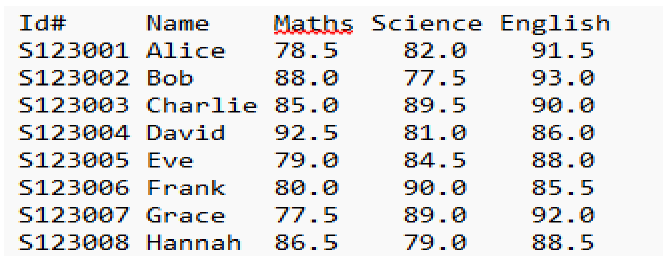
You are tasked with writing a program to process student marks for multiple subjects. The input file *studentmarks.txt* contains student names followed by their marks for Math, Science, and English.

Input File:

The input file called *studentmarks.txt* contains a list of records of students and their marks for three subjects. Each record is on a single line and the fields are separated by spaces. The names of the fields are:

- Student ID
- Student Name
- Math Marks
- Science Marks
- English Marks

An example record may have the following form but the file can have a maximum of **300 records**:



```
Id#     Name    Maths Science English
S123001 Alice   78.5    82.0    91.5
S123002 Bob     88.0    77.5    93.0
S123003 Charlie 85.0    89.5    90.0
S123004 David   92.5    81.0    86.0
S123005 Eve     79.0    84.5    88.0
S123006 Frank   80.0    90.0    85.5
S123007 Grace   77.5    89.0    92.0
S123008 Hannah  86.5    79.0    88.5
```

2.2 Program Specifications:

A. Read the input file and store the data in appropriate arrays.

B. Program should provide user with the following menu [sample start menu is provided in Section 2.4]

   1) **Print the entire list**: Print the provided names and marks of all students as per input file.

   2) **Print details of student matching a provided ID number**: Search for student by the user-input ID # and print their details as provided in the input file.

   3) **Calculate total and display entire List with total:** Calculate the total marks (of the three subjects) of each student and store it in an array and display with other details.

   4) **Print list sorted by total**: Print the list of students sorted by their total mark (lowest to highest). [Please use the algorithm to sort an array given in Section 2.5]

5) **Write report to file:** Prepare a detailed Report as per 2.2.1 below and write to a file named "*summary.txt*"

6) **Exit program**.

C. Repeat step B until the user enters "Q" or "q". After each command is processed in step B, (except menu 6) program must display the menu options again.

D. Note that if option 4 or 5 is chosen before option 3, a message should be displayed instructing the user to invoke option 3 first to calculate the Total Marks before storing or displaying them.

*Note: a sample input file "*studentmarks.txt*" is provided. The number of records for the input files to test your code could differ.

2.2.1 Details of Output File ("summary.txt")

Output in File should have:

- the names and marks of students including the total marks (of the three subjects).
- The output is ranked according to total marks.
- the class average mark for each subject.
- the average of total mark.
- the highest and lowest marks for the overall total marks, including corresponding student details.

<u>Note: The format, organization, readability and neatness of display (on file or screen) will carry marks.</u>

2.2.2 Use of Functions in Assignment

It is necessary that you implement this assignment using functions. To obtain the maximum score you must demonstrate some ability to create and use functions of your own. For example, you may consider creating and using the following functions in your program:

1 A function to read the contents of the file and populate the respective arrays;
2 A function that will search and print student details given ID # as search key (argument);
3 A function to calculate the total marks.
4 A function to evaluate sort the list according to total marks.

2.3 Important Note:

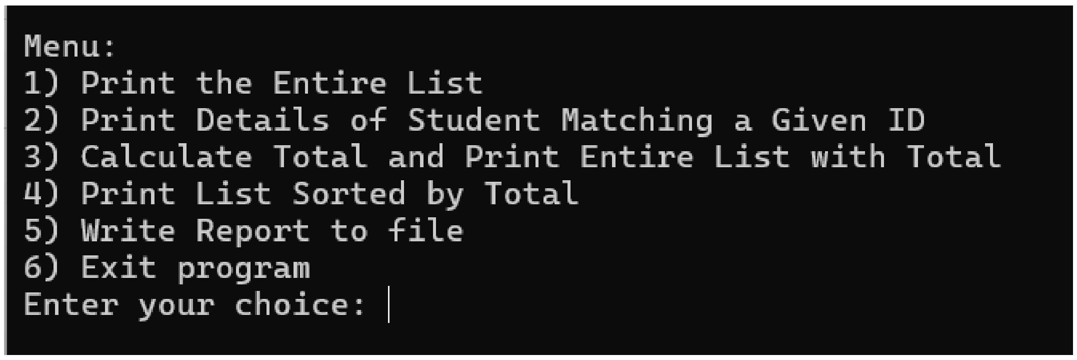
- In all your program constructs you must write comments where necessary. Don't write comments for obvious code, but segments of code which seem complex. Also include your name, student Id# and tutorial group as comments at the top of your program code.

- All input should be validated wherever necessary
- **Please do not use techniques, commands and processes outside the scope of CS111. You will be penalised**

Partial Credit:

Even if your program does not work perfectly, you will receive partial credit for each part you get to work. For example, if your program compiles but gives incorrect results, you can still receive a mark provided that your code is well-written.

2.4 Sample Menu



```
Menu:
1) Print the Entire List
2) Print Details of Student Matching a Given ID
3) Calculate Total and Print Entire List with Total
4) Print List Sorted by Total
5) Write Report to file
6) Exit program
Enter your choice:
```

2.5 Sorting Algorithm

Algorithm to sort an array:
Please **modify** and use the bubble sort code given below to sort the arrays.

```
void bubbleSort(int array[], int size) {
    for (int i = 0; i < size - 1; ++i) {
        // Last i elements are already sorted
        for (int j = 0; j < size - i - 1; ++j) {
            if (array[j] > array[j + 1]) {
                // Swap array[j] and array[j + 1]
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
} similar swap should be applied to other arrays as well.
```

Getting started:

- Download the input files **studentmarks.txt** from Moodle. Depending on the browser you are using, since this is a text file, Moodle will open the file instead of giving you an option of saving it.
  For example, to download file **studentmarks.txt,** you'll need to copy the entire contents of the file after it has been displayed into Moodle, open NOTEPAD program [How? Go to START -> Program Files -> Accessories -> Notepad], paste the copied contents in Notepad and save it as **studentmarks.txt**   in the same directory as your program,.

2.6 Marking

This exercise will be marked using the rubric attached. The following may lead to deductions for the different categories:

Documentation

- Forgetting to add your name as comment.
- Poor comments or complete lack thereof.
- Pointless comments.

Programming Style

- Poor indentation.
- Lack of brackets.
- Poor variable names.
- Poor layout.
- Magic constants.

Semantics

- Uninitialized variables.
- Unused variables.
- Wrong type of loop.
- Wrong data type.
- Wrong Boolean expressions or conditions.
- Wrong use of ifs.

Correctness

- Wrong result for normal test cases.
- Wrong results for boundary cases.
- Does not handle unusual input.
- Wrong or no input validation.

Note that this list is not exhaustive. There are many more mistakes that will result in deductions.

Programing Rubric

Documentation **Deductions for Forgetting to add your name as comment.** **Poor comments, or complete lack thereof.** **Pointless comments.**	Excellent Names and student numbers given, plus good comments. **5points**	Acceptable Some poor comments, or lack of comments except name and student number. **4points**	Amateur If the code contains student names and numbers it at least at this level. **3points**	Unsatisfactory Forgotten to include names. Poor comment otherwise. **1points**	Fail Nothing done on documentation. **0points**
Programming Style **Deductions for** **Poor indentation.** **Lack of brackets.** **Poor variable names.** **Poor layout.** **Magic constants.** **Lowercase constants**	Perfect layout and names. **10points**	Minor glitches in layout. **8points**	Follow general coding guidelines. **6points**	A few attempts at layout. **2points**	No attempt at style**0points**
Semantics **Deductions for** **Uninitialized variables.** **Unused variables.** **Wrong type of loop.** **Wrong data type.** **Mixing float and double** **Wrong boolean expressions or conditions.** **Wrong use of ifs.**	Uses programming construct competently and as intended. **20points**	Some minor mistakes or confusions, that do not affect correctness. **18points**	Some mistakes and confusions, that may or may not affect correctness, but point to lack of knowledge of core concept. **12points**	Several serious mistakes that point to a lack of knowledge of a core concept, and that makes that the program is incorrect. 6points	The program doesn't compile, or if it compiles, then by accident. Several core concepts are not understood. **0points**
Functional correctness **Deductions for** **Not implementing correct algorithm.** **Wrong result for normal test cases.** **Wrong results for boundary cases.** **Does not handle unusual input.** **Wrong or no input validation.**	Computes correct results, and deal with all input, even unusual input. **15points**	Computes correct results, and deals with all normal input. **14points**	Computes correct results for most normal input. **10points**	It works correctly for some inputs, but it is easy to find a mistake. **5points**	Does not compute anything correct. **0points**