CS111

Assignment 2, 2023, Semester 1

Due Date: Friday, 26th May, 2023 [11:55pm Fiji Time]

This project is worth 10% of your coursework.

Learning Outcome:

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

Section 1: Rules

1.0 Assignment Teams.

This is a group assignment where you will work in pairs. Only **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.

1.1 Submit before the Deadline

You have to submit your work before the deadline. Lab computers are scarce, so the trick is to 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.

How to Submit:

- 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 Assign2_YourIdNumber_YourPartnersIDNumber.cpp, where YourIdNumber is your student id number and YourPartnersIDNumber is your partners ID number (e.g. Assign2_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!

Submit your completed assignment in your Moodle Drop Box by 11:55 PM (Fiji Time), Friday, 26th May, 2023. We will not accept “corrected versions” submitted through other channels, nor will we accept late assignments.

1.2 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 co de 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.

1.3 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.
If you have any other problem with the assignment or the course, feel free to email, visit or call the course coordinator.
All contact information is on Moodle.

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 30% 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.

Section 2 (50 marks)

2.1 Problem Statement

Recently at a University, there has been a huge increase in the number of enrollments. All records of students are entered into a flat file by the registrar when the students register and updated accordingly if there is a need. As a result of increase in enrollments, this file now has a lot of records. Hence the statistical officers are now finding a hard time analyzing this file to produce various reports for the management. You are required write a program which will read in this file and produce various reports as outlined.

Input File:
The input file called StudentRec.txt contains a list of student records. Each record is on a single line and the fields are separated by spaces. The names of the fields are:

- Lastname - string
- Firstname (initial) - char
- ID# - integer
- Year of birth - integer
- GPA - double

An example record may have the following form:

Doe J 20211234 1990 3.2

And the format of the actual file is shown below. (Note: file doesn't show all records and for clarity, fields here are shown separated by tabs, but in the actual file they are separated by spaces). Assume that the file can have a maximum of 300 records.

```
File containing student details
Lastname Firstname      ID#         YearOfBirth       GPA
-------------------------------------------------------------
Fast        A          20212221     1999              3.2
Bond        D          20201236     2000              1.0
Hero        F          20193476     2002              4.0
Kuar        G          20211122     1998              2.2
Rich        X          20225412     2002              3.9
```

- The first three lines are just header lines and can be discarded
- Student id contains a 8 digit number.
- Year of birth contains a 4 digit number
- GPA is a one decimal place number between 0 - 4

Program Requirements:

1. You must read this file from the current directory and store the records of students into appropriate arrays.
2. Your program should then be able to process the following commands. i.e. program should provide user with the following menu.
    - 1. Print the entire list as per read file.
    - 2. Print the entire list with inclusion of last name initial and age.
    - 3. Print list sorted by GPA.
    - 4. Print list of students which match a given lastname initial
    - 5. Calculate and display the corresponding degree classification (according to GPA) as provided in Table A.
    - 6. Produce a **file** called Enrollments.txt (in the current directory), with all the details [Last name, Lastname Initial, Firstname Initial, ID#, Age, GPA and degree classification].
    - 7. Exit program.

Explanation of each menu item.

1. *Print the entire class list.*

    This menu item is selected by typing 1 and it should simply print the contents of the file in suitable format on the screen. (Fields (columns) to display in output are: Last name, Firstname Initial, ID#, Age, GPA and degree classification

2. *Print the entire class list with inclusion of last name initial and age*

    This menu item is selected by typing 2 and it should simply print the contents of the file in suitable format on the screen with the inclusion of last name initial and age. (Fields (columns) to display in output are: Last name, Lastname Initial, Firstname Initial, ID#, Age and GPA)

3. *Print class list sorted by GPA*

    This menu item is selected by typing 3. Program then prints the list in ***ascending*** order of GPA. (Fields (columns) to display in output are: Lastname Initial , id# and GPA.) [An algorithm to sort an array is given at the end of this project description]

4. *Print class list of students which match a given lastname initial*

    The user selects this option by typing 4. Then the user is asked to enter the search initial. If there are students who have that initial, program prints their record, otherwise a if no students have last names matching the search initial a "no matching record found" message is displayed. (Fields (columns) to display in output are: lastname and id)

5. Calculate and display the corresponding degree classification as per GPA as provided in Table A. The user selects this option by typing 5. The program then calculates and displays the degree classification according to GPA. The format of the resultant display is shown below:

Lastname	Lastname Initial	Firstname Initial	ID	Age	GPA	Degree
Fast	F	A	20212221	24	3.2	Lower second class honours
Bond	B	D	20201236	23	1	Ordinary degree
Hero	H	F	20193476	21	4	First class honours
Kaur	K	G	20211122	25	2.2	Third class honours
Rich	R	X	20225412	21	3.9	Upper second class honours

[The table (Table A) for degree classification as per GPA is given at the end of this project description]

6. Produce a **file** called **Enrollments.txt** (in the current directory), with all the details [Last name, Lastname Initial, Firstname Initial, ID#, Age, GPA and degree classification].

   The user selects this option by typing 6. The program then generates a file "Enrollments.txt" in the current directory with the following details [Last name, Lastname Initial, Firstname Initial, ID#, Age, GPA and degree classification].

7. *Exit program*

   User selects this option by typing 7. This is when program should end.

[**Note: After each command is processed (except menu 7) program must display the menu options again.** Also, it not necessary that you implement this assignment using functions, however 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 print GPAs in ascending order.
3. A function to calculate age.
4. A function that will search and print all students given a lastname initial as search key (argument).
5. A function to calculate the corresponding degree classification as per GPA

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.]

Function to sort an array:

There are many algorithms to sort array according to ascending or descending criterion. However we will use the following as it is easiest to understand and CODE.

Minimum Element Sort

Go through the array and find the smallest element. Swap it with the first element of the array. Now find the smallest element of the rest of the array and swap it with the second element of the array. And so on.

Rather more formally: to sort (x0,..., xn) into ascending order

```
for i = 0 to n-1 begin
  find smallest of (xi,..., xn)
  swap with xi
```

end

The C++ function code below sorts an array in an ascending order. Please use this function in your program for sorting purposes.

```
//--------------------------------------------------------------------------
void SortAscending(int nElements,string initial[] , int id[], int birth_year[],double gpa[], int age[],string lastname[], string degree[], string firstname[])
{
    double min;
    int minPos;
    for (int i=0; i<nElements-1; i++){
       min = gpa[i];
       minPos = i;
       for (int j=i+1;j<nElements-1;j++)
          if(gpa[j] < min){
                min = gpa[j];
                minPos = j;
          }
       Swap(i, minPos, initial, id, birth_year,gpa,age,lastname,degree, firstname);
    }
}
//--------------------------------------------------------------------------
void Swap(int index, int minIndex, string initial[], int id[], int birth_year[], double gpa[],int age [], string lastname[], string degree[],string firstname[])
{
        string tempstr;
        double temp;

//swap initial array elements
        tempstr = initial[index];
        initial[index] = initial[minIndex];
        initial[minIndex] = tempstr;
//swap lastname array elements
        tempstr = lastname[index];
        lastname[index] = lastname[minIndex];
        lastname[minIndex] = tempstr;
//swap id array elements
        temp = id[index];
        id[index] = id[minIndex];
        id[minIndex] = temp;
//swap age array elements
        temp = age[index];
        age[index] = age[minIndex];
        age[minIndex] = temp;
//swap gpa array elements
        temp = gpa[index];
        gpa[index] = gpa[minIndex];
        gpa[minIndex] = temp;
//swap first array elements
        temp = gpa[index];
```

```
        gpa[index] = gpa[minIndex];
        gpa[minIndex] = temp;
        //swap degree array elements
        tempstr = degree[index];
        degree[index] = degree[minIndex];
        degree[minIndex] = temp;
        //swap gpa array elements
        temp = gpa[index];
        gpa[index] = gpa[minIndex];
        gpa[minIndex] = temp;

}
```

GPA	Degree Classification
4	First-class honours
3.3 - 3.9	Upper second-class honours
2.7 - 3.2	Lower second-class honours
2 - 2.6	Third class honours
0 - 1.9	Ordinary degree (no honours)

Table A: Degree classification corresponding to GPA

Getting started:

- Download the input files StudentRec.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 StudentRec.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 StudentRec.txt in the same directory as your program,.

3.2 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**