SWE20004 -Technical Software Development

Assignment 3

(This assignment is worth 10% of the subject's total assessment marks)

Due Date: Saturday 26th October 2019 at 11:59 pm

Submission instructions

Submit a soft copy of required documents through Canvas.

Introduction:

This assignment requires the knowledge of variables, constants, stream input / output, output formatting, assignment statements, expressions, sequence, selection (if/switch statements), loops, arrays, structure, functions, files, pointers etc.

The assignment:

This assignment has two parts – problem 1 and problem 2.
Develop and submit an original implementation of a menu-driven program performing a number of tasks relating to students' details: displaying all students' details, adding new student to the record, displaying the average mark of all students, finding the maximum marks etc.

You should complete **two problems (Problem 1 and Problem 2).**
Need to submit **Task 8.5** and **Task 8.6** from week 8, **Task 9.2** and **Task 9.3** from week 9 and **Task 10.2** and **Task 10.3** from week 10.

Problem 1

Use the following structure for problem 1.

```
struct student{
      string name;
      int id;
      float mark;
};
```

Declare an array named `studentArray` of the structure type shown above, the size of the array is limited to maximum 100.

The `main()` function handles all interactions with the **user** and other functions:

- It displays an appropriate welcoming message introducing the program.
- Calls a function named `readFile()` which opens the text file `grades.txt` for reading and stores all of the students details from the file to an array named `studentArray`. The `grades.txt` has three columns, first column contains `name`, second column contains `id` and third column contains `mark`. The `readFile()` function has two parameters: one for receiving the file variable and one for the array, both receiving arguments passed by reference.
- It then repeatedly calls the `menu()` function to display user options, get the user selection returned by the menu() function, use a `switch` statement to

process user request by calling appropriate function(s)
- It displays the result with an appropriate message after processing user request.
- It displays a goodbye message when the user selects the **Quit** option from the menu and terminates the program.

The `menu()` function has no parameters. When called, it displays a menu of **8** options allowing the user to select one and returns this option to the calling `main()` function. The options displayed should be:

(1) Display students' details
(2) Calculate average of all students' marks
(3) Sort the students' details
(4) Search for a particular student's mark
(5) Find maximum
(6) Add new student to the record
(7) Quit program

- **Option (1)** will use a function called `displayStudents()` called from the `main()` to display the contents of the `studentArray` array on the screen in an appropriate format. The `displayStudents()` function has two parameters: the array and the size of the array.
- **Option (2)** will use a function called `calculateAverage()` which is designed to calculate the average value of all marks in `studentArray` and return the result to the `main()` function which will then display it with an appropriate message. This function also has two parameters: the array and the size of the array.
- **Option (3):** Ask if the user wants to sort the students' details with respect to either name or marks. Call appropriate functions to do the same.
- **Option (4)** is to search for a particular student's mark. The function should have three parameters, array, size and the name (a string variable) that the user searching for. This function returns the index of the array if it is found otherwise it returns -1. Display all the details of that student in main function. **User should have the options to use either linear search or binary search.**
- **Option (5)** will use a function called `findMaximum()` which is designed to find the largest value of all marks in `studentArray` and return the result to the `main()` function which will then display it with an appropriate message. This function has two parameters: the array and the size of the array.
- **Option (6)** will first use a function called `updateFile()` which will open the file in **append** mode, prompt the user for new student's name, id and mark, and then write the new data at the end of the file using the same format as the original file. It will then the call the `readFile()` function used in the beginning of the program again to read the contents of the updated file and repopulate the `studentArray`.
- **Option (7)** will terminate the program after displaying an appropriate goodbye message.

Problem 2

Use the following self-referential structure for problem 2.

```
struct student
{
      string name;
      int id;
      float mark;
      student *next;
}
```

Create a linked list using the above self-referential structure for this problem. This problem is very much similar to problem 1, but a linked list is used instead of an array. Contents of the file is read into the data part of each node and these nodes are connected together to create a linked list. Must use the linked list for all options listed below.

This problem does not have sorting option.

The options displayed should be:
(1) Display students' details
(2) Calculate average of all students' marks
(3) Search for a particular student's mark
(4) Find maximum
(5) Add new student to the record
(6) Quit program

This problem uses the following functions in addition to main:
`readFile, menu, displayStudents, calculateAverage, linearSearch, findMaximum and updateFile.`
You can decide the parameters and return values of these functions.

Other requirements:

- This assignment must be written in C++ in a procedural language format.
- Your code must have appropriate header(multiline/block) comments including your name and student number, the name of the .cpp file, the purpose of the program, brief explanations of variables and explanations of any code, which is not obvious to another programmer, summarising the input, output and local variables as well as expressions used in your program and test data.
- Include inline (single-line) comments throughout the program describing important statements.
- Use appropriate and descriptive variable following the naming rules and conventions.
- Provide a report, which contains your problem description, evidence of testing – screen shots or pasted output text of several tests, and the contents of the .cpp file (code copied and pasted from text editor).
- Marks will be allocated depending on the amount of original work submitted. Marks will be deducted for plagiarised and/or un-attributed work.

Assignment submission information:

Submissions through **Canvas** must be made on or before the due date/time.

Each submission should have two files.
1. A report (name of the report should be with your student number, eg: 1012546_assignment3.docx) – use template provided with this assignment.

This report will be used for plagiarism check using turnitin software. **20% of marks will be deducted if this report is missing for plagiarism check.** Report must (***.doc/docx, .pdf*** or ***.rtf*** format – use SWE20004_AssignmentReportTemplate) contain:

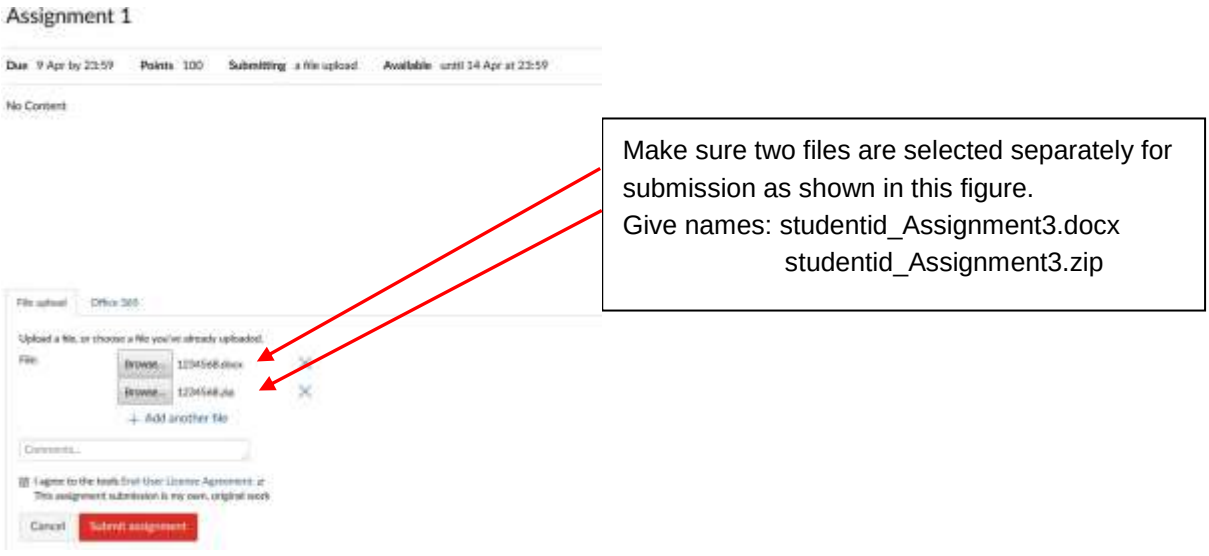
- Description of the problem,
- A copy of the contents of the **.cpp** file for all tasks (copy and paste the code not the screenshot of the code).
- Pasted **text output** or **screen shots** of the working program resulting from the testing of the program

2. A ***.zip*** file (name of the zip file should be your student number, eg: 1012546_assignment3.zip) containing:

a) The actual program (.**cpp** source code) with comments.
   - The name of your **.cpp** files must be **your student number_problem1.cpp (eg**. 386123_problem1.cpp and 386123_problem2.cpp) for the assignment questions.

b) Task 8.5 and Task 8.6
c) Task 9.2 and Task 9.3
d) Task 10.2 and Task 10.3
   - If you are using Visual Studio for this assignment, do not include the solution files, folders or **exe** files.

Submissions larger than 5 MB will not be accepted.



Rubrics

Requirement	Weight (%)	Mark awarded
Program specification and design: (1) Specification of what the purpose and functionality of the program.		
Documentation: (1) Header comments describing (a) the purpose and function of the program (b) Subject and assignment details (c) personal details of the author. (2) Inline comments where appropriate to describe crucial program statements		
Coding: Program includes the following elements in order to meet the requirements - Necessary pre-processing directives - Namespace specification - Correct main() function header - Opening and closing braces for the body of main() - Return statement to terminate main() and program - Variable and constant declarations - Input and Output statements - Proper use of arrays, structure, functions and files - Appropriate use of formatting - Processing including assignment statements, expressions, formulae and calculations as necessary - Appropriate use of sequence, selection - Correct syntax - Correct logic - No runtime errors - Appropriate use of identifier naming rules and conventions - Use of appropriate indentation - testing (screen shots)	50 (problem 1) 35 (problem 2)	
Task 8.5 and 8.6	5+5	
Task 9.2 Task 9.3	5+15	
Task 10.2 Task 10.3	15+10	
Deductions: Marks will be deducted accordingly for invalid submission of required documents such as missing files, corrupt files, incorrect file formats, plagiarism, use of programming language(s) other than C++ and late or non-submission		
Total Assignment mark (out of 140)	140	
Contribution to unit mark (out of 10)	10	