**Lab: Member Functions**

**Objectives:** In this lab, students will:

* learn how to implement membership functions
* distinguish between the stand-alone and member functions
* learn how to implement/use constructors
* prepare for Assignment #4

**Background:** Member functions are those functions declared inside the struct – the prototype of these functions are provided within the struct like any other variable, as follow:

struct Time {

int hours, minutes, seconds;

void print ();

};

The function print() is a member function of the struct Time. We usually use this technique with functions that access or modify a value of this struct type. This approach distinguishes this function from others elsewhere that might have the same name and parameter types. Member functions are implemented in the source (.cpp) file as follow:

void Time::print(){

⋮

}

Please notice the scope resolution operator:: and the name of the struct which precede the function name. A member function is called with an instruction like: myTime.print();

**Start:** To start this lab, open Code::Blocks and create a new project for this lab. Next, add the provided files to your project and then read the problem description.

***Important Notes:***

* *The lab includes three sections (Part A, B, and C). Please read the all information for each part before working on it.*
* *Few files* (main.cpp, studuentData.h, studentData.cpp, Advisor.cpp *and* Advisor.h)*, which include skeleton code, are provided to help you complete the lab.*
* *You don’t need the advisor module* (Advisor.cpp *and* Advisor.h) *to complete Part A of this lab. This module is required to complete Part B of the lab.*
* *Part C is optional.*

**Part A:**

**Problem Description:** Assume, you want to help the academic advisor(s) in the CS Department at ODU. The academic advisor(s) will continuously/regularly check students’ performance each semester (*fall*, *spring* and *summer*). Your task is to write a C++ program to automate this process and to ease the job of the academic advisor. It is very important for the academic advisor to know the average scores of the students so he/she can decide which student(s) need more of his/her attention and advising time.

You can read the student information for this program from the “StudentData.txt” file, which is provided. This file includes the students’ information as entered by the registrar office. The file is lacking some important information about the students’ average scores – which the academic advisor needs. You should write a program C++ to generate the missing average score information. Assume every student in the CS Department are taking 3 classes. The student data consists of the following information.

Name of the student, UIN, Course 1 grade, Course 2 grade, Course 3 grade

Your task is to implement the following three functions:

1. sortStudentData(student \* studentList, int numberOfStudents);
2. int studentAverage();
3. bool checkIfGreater(student studentRecord );

The first function is standalone and performs sorting of student data by their UIN. The second function is to find the average grade of a student. The third function is to compare two students’ averages. The second and third functions have to be implemented as member functions of the struct while the first function is a standalone function.

**Input:**

The input to this program is read from the text file ‘studentData.txt’.

The student data consists of student Name, UIN, marks for Course1, Course2 and Course3.

We store these values by creating a struct named ‘student’ with corresponding fields and then reading from the text file.

**Output:**

The output of this program should write the sorted list of the average scores of the students in a text file named ‘StudentResult.txt’. Make sure you put the student name and UIN before the average so you know which students the adviser needs to contact for a meeting.

**Part B:**

**Problem Description:** Assume after completing Part-A of this lab, you presented your complete/working program to one of the advisors in the CS Department. The advisor gave you his/her feedback and a couple tips on how to improve the performance of your program. The advisor wants your program to categorize students into four different categories based on the students’ average grade in their completed classes. Students with average grade above 93 are categorized as “doing well”. Students with average grade between 93 and 85 (*inclusive*) are categorized as “Advisor needs to monitor the student performance – no advising is required at the moment”. Students with average grade between 85 and 70 are categorized as “Student needs some advising”. Anyone with grades lower than 70 is categorized as “Needs urgent advising.”

Assume all advisors in the CS Department want to use your program. As a result, you will need to create a new module to implement an ‘advisor’ struct. You will also need create a list of this struct to represent the 3 advisors. The advisor struct should have the advisor’s name, and an array of his/her student’s ID numbers. The first 3 students in ‘studentData.txt’ are advised by advisor Ben, second 3 students are advised by advisor Jerry, and the last 3 students are advised by advisor Tom. Please do not forget to assign these names to the advisors. You will need to implement a standalone function that matches the students to their advisors.

**Output:** You should also have a member function that prints the names and IDs of the students that they are going to be meeting just below the student information in the studentResults.txt file, separated by a line (Example, “Ben should meet Alex 01000036”). You can search for “appending text to a txt file” to find how to output to the end of a txt file.

**Part C:**

**Background:** A constructor is a member function – mainly to initialize a struct or class with a user defined values. If a constructor is not provided, the compiler generates one and uses it to initialize the member elements. A good programming habit is to provide at least one constructor for each structs/class you will create. An example can be seen below:

struct Phone{

string manufacturer;

int memory\_size;

double camera\_MP;

string operatingSystem;

bool touchscreen;

Phone()

{

manufacturer = “”;

memory\_size = 0;

camera\_MP = 0;

operatingSystem = “”;

touchscreen = false;

}

};

**Problem Description:** Like any programmer, because you built the program, now you are in charge of maintaining it. The department is expanding and they decided to hire more advisors to be able to keep up with the increasing number of students. To ease the process of getting the new advisors registered in your program, and to avoid any future problems, you decide to add constructors to your advisor struct. You will need to add a default constructor (no parameter) and a constructor with parameters (string for advisor name) and implement that either in your struct like shown above or like a member function utilizing advisor.cpp.