Object-Oriented Programming

Lab # 7

**Important Instructions:**

* You are required to create a **multi-file project** to accomplish each task.
* Indent your code properly.
* Use meaningful variable and function names. Follow the naming conventions.
* Make sure that there are no **dangling pointers** and **memory leaks** in your program.

**Task #1**

Create a class **Student** that has private member variables to store each of the following six attributes:

* **Roll Number**: An integer variable that holds the student’s roll number
* **Name**: A c-string that holds the student’s name (max length 40) (*Note: Declare the size of the c-string as a named constant before the start of class declaration*)
* **Marks**: An int\* through which you will allocate an array (of appropriate size) to store the marks obtained by the student in different quizzes
* **Number of Quizzes**: An integer to store the number of quizzes taken by the student (*Note: Number of quizzes can be different for each student*)
* **Total Marks of each Quiz:** A constant integer to store the total marks of each quiz.
* **Number of Students**: A static integer variable to store the number of Student objects currently existing in the program.

Now, carry out the following tasks in the given order:

1. Implement a **Default Constructor** for **Student** class in which roll number should be given the value 0, name should be initialized to empty c-string, marks should be initialized as a null pointer, number of quizzes should be initialized to 0, and total marks of each Quiz should be initializes to 10.
2. Implement an **Overloaded Constructor** for **Student** class that accepts 4 arguments: student’s roll number, student’s name, number of quizzes taken by the student, and total marks of each quiz. The values supplied in the arguments should be used to initialize the corresponding member variables, and dynamically allocate memory through the pointer marks. Each element of the marks array should be initialized to 0.
3. Implement the **Destructor** for **Student** class which should deallocate all dynamically allocated memory.
4. Implement a member function **inputMarks** of the **Student** class, which should ask the user to enter the marks of quizzes for the Student object on which this function has been called.

**VERY IMPORTANT:** If the Student object was created through the default constructor, then the Marks array would not have been allocated. In this case, this function should firstly ask the user how many quizzes were taken by the student. Then, it should allocate memory through the pointer marks, and after that ask the user to enter the marks of quizzes.

1. Implement the **Copy Constructor** for **Student** class which should make a deep copy of the Student object whose reference it will receive as a parameter.
2. Implement a static member function **getNumberOfStudents()** of the **Student** class, which should return the number of Student objects that currently exist in the program.
3. Implement a global function **Student generateOneStudent()** which should ask the user about all attributes of a Student (roll number, name, number of quizzes, total marks of each quiz), then it should create a Student object by using the overloaded constructor that you implemented for the Student class. After that, this function will call the **inputMarks** function (which you implemented in part (iv) above) on the Student object to input and store the marks of all quizzes. Finally, this function will return the Student object that it just created.
4. Write a driver program (main function) which should demonstrate all of the above functions.
5. Modify the **inputMarks** function (which you implemented in part (iv) above) to perform **input validation** on the quiz marks entered by the user. Marks of each quiz should be greater than or equal to 0 (no negative marking ☺) and less than or equal to the total marks of the quiz.
6. Implement a public member function **displayDetails()** of the **Student** class which should display the roll number, name, marks obtained by the student in different quizzes, and the average marks obtained by the student.
7. Implement a public member function **storeInFile(ofstream&)** of the **Student** class that stores all information of the Student in the text file which has been opened through the file handle passed into this function.
8. Implement a public member function **readFromFile(ifstream&)** of the **Student** class that reads all information of a Student from the text file (which has been opened through the file handle passed into this function) and stores this information in the member variables of the Student object on which this function has been called.
9. Write a driver program which should create several Student objects and store them in a text file by using the member function **storeInFile** that you implemented in part (xi) above. Decide about the format of the file yourself, but make sure that all necessary information is stored in the file in a manner so that it can be read later on (see the next step).
10. Write a driver program which should open the file created in step (xiii) and reads information of all the Student objects stored in it. Here, you will be using the function **readFromFile** of the Student class that you implemented in part (xii) above. After reading all objects from the file, the driver program should also display information of all Student objects on screen (using the **displayDetails()** function on each Student object).

**Task #2**

Implement the copy constructor for the Number Array class which you implemented to solve the Programming Challenge # 11 given at the end of Chapter 13 in Gaddis’ book.

**Task #3**

Write a C++ program for the Programming Challenge # 2 (Day of the Year) given at the end of Chapter 14 (Page 864) in Gaddis’ book.

**Task #4**

Implement the StraightLine and Parabola classes and defined the point of intersect between these two equations by calling the intersect () function. This function is defined as the friend function of both classes.