Lab 4

Object Oriented Programming

Instructions:

• **Attempt the following tasks exactly in the given order.**

• You are required to create a **multi-file project** for each task.

• Indent your code properly.

• Use meaningful variable and function names. Follow the naming conventions.

• Use meaningful prompt lines and labels for all input/output.

• Make sure that there are **NO dangling pointers** or **memory leaks** in your program.

Task # 1

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

- ID (an integer to uniquely identify each employee)
- Name (a char array to hold the employee's name)
- Dept. Name (a char array to hold the employee's department name)

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

i.) Implement a **default constructor** for **Employee** class that assigns 0 to the ID, and null to the name and department.

ii.) Implement an **overloaded constructor** for **Employee** class that accepts the following 3 values (in the given order) as arguments and assigns them to the appropriate member variables: employee's ID, employee's name, and department name.

iii.) Implement an **overloaded constructor** for **Employee** class that accepts the following 2 values (in the given order) as arguments and assigns them to the appropriate member variables: employee's ID and employee's name. The department name should be assigned null.

iv.) Implement the getter and setter functions for each member variable of the **Employee** class.

v.) Implement a member function **display** of the **Employee** class which should neatly display all attributes of an employee on screen. This function should use the getter functions to get the values of all attributes of an employee.

vi.) In the main function, dynamically allocate an array of **Employee** objects. Your program should ask the user about the size of the array and then allocate the array dynamically. Then, your program should ask the user to enter the values of all attributes for each **Employee**.

After that, your program should display the details of each object on screen (by calling the **display** function for each **Employee**).

Task # 2

Create a class **Fraction** containing two integer data members named **num** and **den**, used to store the numerator and denominator of a fraction having the form **num/den**.

- Implement an **overloaded constructor** for **Fraction** class which will receive two integer values and store them in **num** and **den** respectively. The default values of these two arguments should be 1.
- Implement a member function **setFraction(int,int)** of the **Fraction** class which sets the values of **num** and **den** to the values passed as arguments into this function.

- Implement a member function **display()** to display a **Fraction** on screen in the form num/den. For example, if **num** is **3** and **den** is **4** the fraction should be displayed as **3/4**.

- Implement a member function **add,** adding two Fractions and returning the result. The prototype of this function should be:

  Fraction add(const Fraction&) const;

  The addition of two fractions is defined by the following rule:

$$a/b + c/d = (a*d + b*c)/(b*d)$$

  *Note: Fraction is always stored in its lowest form. For example if **num** is 9 and **den** is 6 than it store as 3/2 instead of 9/6. Result of Addition is also stored in lowest form.*

Task # 2

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 char array that holds the student's name (assume that max length of name is 40)
• **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*)
• **Marks**: An int* through which you will allocate an array (of appropriate size) to store the marks obtained by the student in different quizzes
• **Total Marks of each Quiz**: An integer to store the total marks of each quiz.
Now, carry out the following tasks in the given order:

i.) Implement a **Default Constructor** for **Student** class in which roll number should be given the value 0, name should be initialized to null, marks should be initialized as a null pointer, number of quizzes should be initialized to 0, and total marks of each Quiz should be initialized to 10.

ii.) 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.

iii.) Implement the **Destructor** for **Student** class which should deallocate all dynamically allocated memory (if any).

iv.) Implement a public member function **getInputFromUser** of the **Student** class, which should ask the user to enter the following 4 attributes: Roll No., Name, No. of Quizzes taken by the student, and Total marks of each quiz. After storing this data in appropriate member variables, this function should allocate the Marks array. If the Marks array has been previously allocated make sure to deallocate it first (see the description of overloaded constructor above in step **(ii)**). After that this function should call the following private member function **inputMarks** to input the marks of each quiz (see next step).

v.) Implement a **private** member function **inputMarks** of the **Student** class, which should ask the user to enter the marks of all quizzes (one-by-one) and store these marks in the array inside the calling Student object. Also perform **input validation** on the quiz marks entered by the user. Marks of each quiz should be greater than or equal to 0 and less than or equal to the total marks of the quiz.

vi.) Implement a public member function **display()** 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.

vii.) 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.

viii.) Implement a **global function** **void printStudent (Student)**. This function should take a Student object which is passed by value into it. This function should display the details of that Student object on screen by calling its display function. There will be only a single statement in the body of this function ☺. The purpose of this function is to test the implementation of Copy Constructor which you implemented above in step **(viii)**.

ix.) Write a driver program (**main function**) which should demonstrate all of the above functions.