National University of Computer & Emerging Sciences – FAST (CFD)
Department of Computer Science

National University of Computer and Emerging Sciences

Assignment 02

Object Oriented Programming

Course Coordinator	Rizwan Ul Haq
Course Coordinator(s)	Sajid Anwer
Semester	Spring 2024
Open Date	14-Mar-2024
Submission	22-Mar-2023

FAST School of Computing

National University of Computer & Emerging Sciences – FAST (CFD)
Department of Computer Science

Submission Instruction

1. Make a folder by the name in the following format. Assignment_02_RollNumber (Assignment_02_22F1234). Copy all .cpp files only (not the whole project) with name of the question (Q1.cpp, Q2.cpp) in this folder. Compress it and submit on google classroom.
2. A .doc file containing all the codes and screenshots should also be included.
3. The code must be properly commented.
4. Assignment will be marked as zero in case of any plagiarism. Submit your own work only.
5. Any assignment not following the above given instructions will not be evaluated.

Question# 1

Create a student database of Students using a class. Your program will have two classes One for sections and One for Students. Class student will have the following **private** attributes:

1. Name
2. CNIC
3. Gender
4. CGPA

*CNIC and gender values cannot be changes once initialized.
*Having required accessor functions.

And the class section will the following **private** attributes:

1. An array of Objects of class Students less than 40.
2. Section name
3. Class teacher.

Having functions **editSection(), addStudent()**, **updateStudent()**, **printList().** In main function, you'll have a pointer of class section. You'll ask the user for number of sections he wants to create, and then create as many objects as user wants. Now, provide a menu to perform these functionalities: 1. Edit Section Attributes 2. Add Student in a section 3. Update Student of a section 4. Print List of Students of a section 5. Print List of Sections You can access functions of class student directly from main(). Only call the function of class student. Delete all the memory at the end.

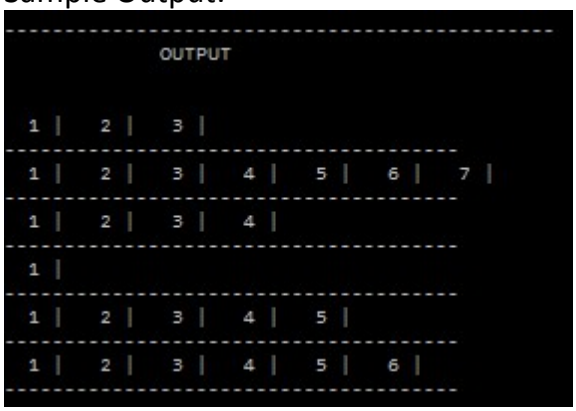
Question# 2

Design a Class called **Jagged** that will explore the jagged array concept (2-D dynamic array).

Jagged class has attributes int row, column and 1-D array as private data members and following public methods;

- Default Constructor: initialize all members variables with Null values
- **Input**(), return type will be **Jagged** and with no input argument. It will ask from user about the number of rows and then ask for the number of column of each row. Number of column will be different for each row and will be stored in 1-D array. Array size will be equal to the number of rows in jagged array.
- **Display**(), return type should be void and accepts **Jagged** object. Print the all data of your 2D jagged array in a proper and clear way.
- Destructor that will release all the memory

Sample Output:



Question# 3

A. Create a class that counts its own instances, automatically. You may name the class “CountClass”. Any and all member variables will be private.
NOTE: There must be NO hardcoding.
B. Find the errors in the following class and explain how to correct those

```
1   class Example
2   {
3       public:
4           Example( int y = 10 )
5           : data( y )
6           {
7               // empty body
8           } // end Example constructor
9           int getIncrementedData() const
10          {
11              return ++data;
12          } // end function getIncrementedData
13          static int getCount()
14          {
15              cout << "Data is " << data << endl;
16              return count;
17          } // end function getCount
18      private:
19          int data;
20          static int count;
21  }; // end class Example
```

Provide Logical reasoning with proper explanation !

Question# 4

Imagine a tollbooth at a bridge. Cars passing by the booth are expected to pay a 50 cent toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth keeps track of the number of cars that have gone by, and of the total amount of money collected. Model this tollbooth with a class called **tollBooth**. The two data items are a type unsigned int to hold the total number of cars, and a type double to hold the total amount of money collected. A constructor initializes both of these to 0. A member function called **payingCar()** increments the car total and adds 0.50 to the cash total. Another function, called **nopayCar(),** increments the car total but adds nothing to the cash total. Finally, a member function called

display() displays the two totals. Make appropriate member functions const.

Include a program to test this class. This program should allow the user to push one key to count a paying car, and another to count a nonpaying car. Pushing the Esc key should cause the program to print out the total cars and total cash and then exit.

Question# 5

Create a class called time that has separate int member data for hours, minutes, and seconds. One constructor should initialize this data to 0, and another should initialize it to fixed values.
Another member function should display it, in 11:59:59 format. The final member function should add two objects of type time passed as arguments.
A **main()** program should create two initialized time objects (should they be const?) and one that isn't initialized. Then it should add the two initialized values together, leaving the result in the third time variable.
Finally it should display the value of this third variable. Make appropriate member functions const.

Question# 6

Write the code, compile, and test the following exercise. The class interface (.h), class implementation (.cpp), and main program (.cpp) should be stored in three separate files.

Design a **TestScores** class that has a pointer member variable to hold test scores. The class should have a constructor that allows the user to specify the number of test scores. The constructor then dynamically allocates the memory.

- Overload the operator=
- Provide a copy constructor.
- Include a destructor.
- Include a member function that returns the average of the scores.

Demonstrate the class by writing a separate program that creates at least 2 instances of the class. The program should ask the user the number of test scores,

and then define a TestScores object. The program should call the mutator function to prompt the user for the values, storing them in the TestScores object. Then, the program should display the TestScores in the object, by calling the accessor member function. Display the average of your test scores.

When you instantiate the second TestScores object, initialize it to the first object (to test the copy constructor). Then, make changes, and test the assignment operator.

Question# 7

Make a class Car having the attributes carName, model, plateNumber, and color.
Design a constructor, as well as a copy constructor.
Make two objects for two different cars.
Now, use the copy constructor to set all the information of Car # 1 similar to Car # 2.
Make a member function that returns the information of both cars. The information of Car 1 (that was not originally set) should now be similar to whatever input you gave for Car # 2.

Sample Output:

```
Set Information for Car 2...
Enter Plate Number:
LOU-9041
Enter Color:
Orange
Enter Model:
2010
Using copy constructor to copy information...
Displaying info using Car 1 object
Plate Number: LOU-9041
Color: Orange
Model: 2010
Displaying info using Car 2 object
Plate Number: LOU-9041
Color: Orange
Model: 2010
```

Question# 8

Create a class “Scientist” containing id, name, DOB, graduatedFrom, pickedDomain etc…
This class has two methods **inputData()** and **displayData().**
Make another class “Chemist” derived from “Scientist” which includes the attribute favouriteChemical, and chemRank.

Now, this class also has two methods of the same name **inputData()** and **displayData()** Make only ONE object of the class Chemist, input all information and then display it.

Your output must include all attributes including id, name, age, graduatedFrom, pickedDomain, favouriteChemical and chemRank