**CS217 Object Oriented Programming**

**Week 6- Classes II**

**Task 1**

Design a class **Complex** for handling Complex numbers and include the following **private** data members:

* real: a double
* imaginary: a double

The class has the following member functions.

1. A constructor initializing the number with **default parameters**.
2. Overloaded Constructors.

* Complex(double r, double i)

Note\*:Use member function initialization for all data members.

* Complex(Complex & copy) // copy constructor

1. Getters and Setters of the class data members as given below

* void setReal(double r)
* double getReal()
* void setImaginary(double i)
* double getImaginary()

1. Overload the following member function in the class

* **Complex addComplex( double r)**

It adds r of type double to real part of complex number while imaginary part remains same. And returns newly generated complex number.

* **Complex addComplex(Complex &c1)**

It adds both complex numbers and returns newly generated complex number.

* **Complex subComplex(double r)**

It subtracts r of type double from real part of complex number while imaginary part remains same. And returns newly generated complex number.

* **Complex subComplex(Complex &c1)**

It subtracts both complex numbers and returns newly generated complex number.

* **Complex mulComplex(double n)**

It’s a scalar multiplication. Real and imaginary parts are multiplied by n. and returns newly generated complex number.

* **Complex mulComplex(Complex &c1)**

It multiplies both complex numbers and returns newly generated complex number. (a+bi)(c+di) = (ac−bd) + (ad+bc)i

**Task 2**

You are developing a C++ program for a car rental service, and you're tasked with creating the `Car` class to manage car rental operations. The `Car` class should have static data members, constructors, and additional member functions as follows:

1. Define the `Car` class with the following data members:

- `make` (string) to store the car's make (e.g., "Toyota," "Ford").

- `model` (string) to store the car's model (e.g., "Camry," "Mustang").

- `rentalPrice` (double) to store the daily rental price of the car.

2. Implement a static data member `totalCarsRented` to keep track of the total number of cars rented by the service.

3. Create a constructor to initialize the `make`, `model`, and `rentalPrice` data members. The constructor should also increment the `totalCarsRented` count each time a new car object is created.

4. Add a member function `displayCarInfo()` that displays the details of a specific car, including its make, model, and rental price.

5. Create a static member function `getTotalCarsRented()` that returns the total number of cars rented by the service.

6. Demonstrate the use of the `Car` class by creating several car objects, renting and returning them, displaying car information, and showing the total number of cars rented using the provided member functions.

**Task 3:**

We want to create a class of Counter, the object of which holds the count of anything. Also we want to keep track of total objects of class and serial No of each object. Write a class Counter. This class has **three private data members:**

* **count:** An integer that holds a count value.
* **objCount:** A static integer that holds the count of objects.
* **serialNo:** An integer that holds the serial number of objects of Counter class.

**2.1:** Write a default constructor that initializes each data member of the class such that count with 0, objCount that increments the count of the objects and serialNo with the correct serial no (object 1should have serial no 1, object2 should have serial no 2 and so on). Counter( )

1.2 Write a constructor that accepts an argument int c that is assigned to the data member count. Also initialize objCount with the count of the objects and serialNo with the correct serial no. Counter(int c)

1.4 Create the getter-setter functions for the data members.

void setCount(int c)

int getCount()const

int getSerialNo()const

static int getObjCount()

static int IncrementObjCount()

1.7 Write main function to test all the implemented functionality.