Lab9: Static Members and Operator Overloading

Instructions

- *Make your own files (3 files for each class). Names of the files must be the same as the class name; i.e. for Point class there should be header file (Point.h), implementation file (Point.cpp), and main file (Point_test.cpp). Write the main function of each class in its test file.*
- Use the macros and pre-processor directives(#ifndef, #define, #endif) for each class.
- *Data members names of each class should be the same as mentioned in each question.*
- Declare the member functions constant where appropriate. Also implement the required checks in member functions.
- Please read the questions carefully, read them twice even thrice to understand them completely.
- In case of any query, please raise your hands and we will be there to solve your query.
- Please concentrate, understand, and code. Good Luck :)

Task 1

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: An integer that holds the count of objects.
- serialNo: An integer that holds the serial number of objects of Counter class.

1.1 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 1 should 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 that increments the count of the objects and serialNo with the correct serial no (object 1 should have serial no 1, object2 should have serial no 2 and so on).
`Counter(int c)`

1.3 Write destructor of the class which adjust the count of Objects.
`Counter()`

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()`

1.5 Define operator = that assigns the value of count to the left hand operand.

1.6 Define "-" unary operator that inverts the value of count data member for counter class and should allow the statements like $c1 = -c1$; or $c2 = -c1$;

1.7 Write `main` function to test all the implemented functionality.

Task 2

Write a class `Distance` that holds distances or measurements expressed in feet and inches. This class has two private data members:

- feet: An integer that holds the feet.
- inches: An integer that holds the inches.

2.1 Write a constructor with default parameters that initializes each data member of the class such that `int f` is assigned to `feet` and `int i` is assigned to `inches` (Default values 0)
If inches are greater than equal to 12 then they must be appropriatly converted to corresponding feet

2.2 Generate appropriate getter-setter functions for the data members.

- `bool setFeet(int f)`
- `int getFeet()const`
- `bool setInches(int i)` It should ensure proper conversion to feets.
- `int getInches()const`

2.3 Define an operator+ that overloads the standard + math operator and allows one Distance object to be added to another. `Distance operator+(const Distance &obj)`

2.4 Define an operator- function that overloads the standard - math operator and allows subtracting one Distance object from another. `Distance operator-(const Distance &obj)`

2.5 Define an operator> function that overloads the > operator and returns true if the calling object contains a value greater than that of the parameter and returns false otherwise. `bool operator>(const Distance &obj)`

2.6 Define an operator< function that overloads the < operator and returns true if the calling object contains a value smaller than that of the parameter and returns false otherwise. `bool operator<(const Distance &obj)`

2.7 Define an operator= function that overloads the = operator and assign one Distance object to another. `const Distance  operator=(const Distance &obj)`

2.8 Write `main` function to test all the implemented functionality.

Task 3

Design a class `Complex` for handling Complex numbers and include the following:

- real: a double
- imaginary: a double

The class has the following member functions.

3.1 A constructor initializing the number with default parameters.

3.2 Getters and Setters of the class data members as given below

- void setReal(double r)
- double getReal()const
- void setImaginary(double i)
- double getImaginary() const

Overload the following operators in the class by identifying their return types.

3.3 Overload the binary + operator which adds two complex numbers.

3.4 Overload the binary - operator which subtracts the complex number passed as argument from the caller object.

3.5 Overload the binary * operator to multiply two complex numbers.(Think about its return type).

3.6 Overload binary assignemnt = operator, think about its return type.

3.7 Overload binary == operator which returns true if operands are equal and returns false otherwise.

3.8 Overload unary ! operator which returns true if the real and the imaginary parts are zero, otherwise return false. `bool opeator!()const`

3.9 Write `main` function to test all the implemented functionality.

Task 4

Write a class called `Date` that represents a date consisting of a year, month, and day. A Date object should have the following methods and operators:

You are given the operation and you have to overload the required operator for the objects of class.

Date(int year, int month, int day)	Constructs a new Date object to represent the given date.
operator =	Overload = operator to assign values.
d2=d1+1	Overload + operator which takes integer as argument . It moves this Date object forward in time by the given number of days.
d2=d1-4	Overload - operator which takes integer as argument . It moves this Date object backward in time by the given number of days.
d3=d1+d2	Overload + operator which takes Date object as argument.
d3=d1-d2	Overload - operator which takes Date object as argument.
bool a=d1>d2;	Overload > operator which returns true or false.
bool a=d1>=d2;	Overload >= operator which returns true or false.
bool a=d1<d2;	Overload < operator which returns true or false.
bool a=d1<=d2;	Overload <= operator which returns true or false.
bool d1!=d2;	Overload != operator which returns true or false.
bool d1==d2;	Overload == operator which returns true or false.
int getDay()	Returns the day value of this date; for example, for the date 2006/07/22, returns 22.
int getMonth()	Returns the month value of this date; for example, for the date 2006/07/22, returns 7.
int getYear()	Returns the year value of this date; for example, for the date 2006/07/22, returns 2006.
bool isLeapYear()	Returns true if the year of this date is a leap year. A leap year occurs every four years, except for multiples of 100 that are not multiples of 400. For example, 1956, 1844, 1600, and 2000 are leap years, but 1983, 2002, 1700, and 1900 are not.
String toString()	Returns a String representation of this date in year/month/day order, such as “2006/07/22”.

Write `main` function to test all the implemented functionality.