# CS217 Object Oriented Programming

## Instructions

1. Make your own files (2 files for each class). Name of files must be same as class name such that for Point class there are header file (Point.h), Implementation file(Point.cpp).Write only one main function. Main function is only for testing purpose don't submit main.cpp.
2. Please use the macros and pre processor instructions (#ifndef, #de_ne, #endif) in each class header file.
3. Data member names of each class should be the same as per mentioned in each question.
4. Please read the questions carefully, read them twice even thrice to understand them completely.
5. In case of any query, please raise your hands and we will be there to solve your query.
6. Please concentrate, understand and code. Good Luck :)

## Task 1

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.

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
3. 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
4. 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

Write a class named as Holiday that represents a holiday during the year. This class has three private data members:

- name: A string that represents the name of holiday.
- day: An integer that holds the day of the month of holiday.
- month: A string that holds the month the holiday is in.

1. Write a default constructor that initializes each data member of class such that name with NULL, day with 0 and month with NULL
   Holiday()
2. Write a constructor that accepts the arguments for each data member such that string n assigned to name, int d to day and string m to month.
   Holiday(const string &n, int d, const string &m)
   Note*:Use member function initialization for all data members.
3. Generate getter setter of each member variable : such that name should never be greater than 50 characters, day should never be negative and month should not be greater than 10 characters.
   - bool setName(const string &s)
   - string getName() const
   - bool setDay(int u)
   - int getDay() const
   - bool setMonth(const string &p)
   - string getMonth() const
4. Write a global function **inSameMonth** in Holiday.cpp file which takes two Holiday objects as arguments, compares two objects of the class Holiday, and returns true if they have the same month otherwise false.
   bool inSameMonth (const Holiday &a, const Holiday &b)
5. Write a global function avgDate in Holiday.cpp file which takes an array of type Holiday and its size as its argument and returns a double that is the average of all the day data member in the Holiday array arr. You may assume that the array is full (i.e. does not have any NULL entries).
   double avgDate(Holiday arr[], int size)

## Task 3

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

- Date(int year, int month, int day)
  Constructs a new Date object to represent the given date.

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

- void add(const int &days)
  Moves this Date object forward by the given number of days. hint:* you should decide on the basis of month and year that given month ends 30,31,28,29 days.
- void add(const int &month , const int &days)
  Moves this Date object forward by the given number of months and days. Months should be within 1 to 12 and days in 1 to 31. For Example Date 2003/12/31and add(1,29) =>Date will be 2004/02/29
- void add(Date & other)
  Moves this Date object forward by the given Date.
- void addWeeks(const int &weeks)
  Moves this Date object forward by the given number of seven-day weeks.
- int daysTo(const Date & other)
  Returns the number of days that this Date must be adjusted to make it equal to the given other Date.
- void subtract(const int &days)
  Moves this Date object backward by the given number of days. hint:* you should decide on the basis of month and year that given month ends 30,31,28,29 days.
- void subtract(const int &month , const int &days)
  Moves this Date object backward by the given number of months and days.
- void subtract (Date & other)
  Moves this Date object backward by the given Date.
- int getDay()const
  Returns the day value of this date; for example, for the date 2006/07/22, returns 22.
- int getMonth()const
  Returns the month value of this date; for example, for the date
  2006/07/22, returns 7.
- int getYear()const
  Returns the year value of this date; for example, for the date 2006/07/22, returns 2006.
- bool isLeapYear()const
  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".

## Task 4

Write a class Point with data members

- x : a integer x coordinate
- y : a integer x coordinate

The class has the following member functions.

1. Default constructor initializing the coordinates to zero
   Point()
2. A constructor that takes the values and initializes coordinates with x1 and y1
   Point (int x1, int y1)

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

3. A constructor that takes Point's reference and initializes coordinates
   Point (Point &copy) // it's a copy constructor
4. a destructor that prints the following statement on screen :"Destructor Called" Point()
5. Getter and Setter functions.

## Task 5

Write a class called Line that represents a line segment between two **Points**. Your Line objects should have the following methods:

- Line(const Point &p1, const Point &p2)
  Constructs a new Line that contains the given two Points.
- Line(int x1, int y1, int x2, int y2)
  Constructs a new Line that contains the given two Points.
- Line(Line &copy) // it's a copy constructor
  Constructs a new Line form given line.
- Point getP1() const
  Returns this Line's first endpoint.
- Point getP2() const
  Returns this Line's second endpoint.
- double getSlope() const
  Returns the slope of this Line. Remember The slope of a line between points (x1, y1) and (x2, y2) is equal to (y2-y1)/(x2-x1).
- String toString()
  Returns a String representation of this Line, such as "[(22, 3), (4,7)]" .

# Task 6

Write a class called Rectangle that represents a rectangular two-dimensional region.

class Rectangle with data members

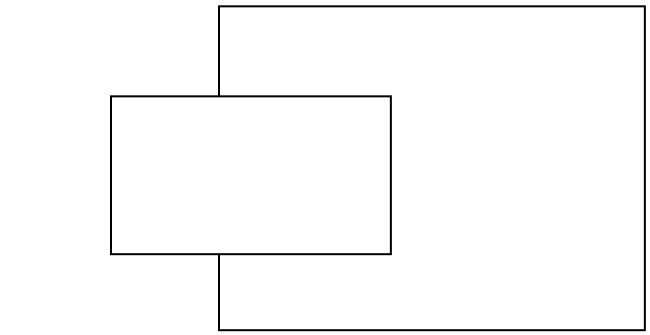
- p : a Point having x coordinate and y coordinate.
- width: a integer width of a Rectangle.
- height: a integer height of a Rectangle.

Your Rectangle objects should have the following methods:

- Rectangle(int x, int y, int width, int height)
  Constructs a new Rectangle whose top-left corner is specified by the given coordinates and with the given width and height.

  Note*:Use member function initialization for all data members.
- Rectangle(Rectangle &copy) // it's a copy constructor
  A constructor that takes Rectangle's reference and initializes attributes

  Note*:Use member function initialization for all data members.
- int getHeight()const
  Returns this Rectangle's height.
- int getWidth()const
  Returns this Rectangle's width.
- int getX()const
  Returns this Rectangle's x-coordinate.
- int getY()const
  Returns this Rectangle's y-coordinate.
- String toString()
  Returns a String representation of this Rectangle, such as "Rectangle[x=1, y=2, width=3, height=4]".
- Rectangle(const Point & p, int width, int height)
  Construct a new Rectangle whose top-left corner is specified by the given Point and with the given width and height.
- bool contains(const Point &p)
  Returns whether the given Point or coordinates lie inside the bounds of this Rectangle. *Boundary of rectangle is included.
- Rectangle union(const Rectangle &rect)
  Returns a new Rectangle that represents the area occupied by the tightest bounding box that contains both this Rectangle and the given other Rectangle.
  Hint:*
  1. If the Rectangles do not intersect at all, returns a Rectangle with width and height both equal to 0.
  2. Both the rectangles must have same width OR height .



Result in this case

Rectangle (0,0,0,0)

Result in this case will be the valid union Rectangle

## Task 7

1. Write a class Course that includes the following members:

| Data Members | Properties Description |
|---|---|
| courseCode | Stores course code, like “CS 103” |
| courseTitle | Stores course title, “Computer Programming" |
| creditHours | Stores credit hours of the course e.g. 1,2,3 & 4. |
| section | Stores section in which student is registered. E.g. A, B, C etc |
| repeatCount | Stores repeat count of the course for the student. Like 1,2,3 |

Member Functions:

a. getters and setters for each data members.
b. Default constructor
c. Copy constructor
   Course(Course & c)
d. Parametrized constructor
   Course(string cc, string ct, int ch,char s, int rc)

2. Write a class **Semester** that includes the following members:

| Data Members | Properties Description |
|---|---|
| semesterCode | Stores the code of semester, like “Spring 2018" |
| courseCount | Stores the number of courses registered between one and five courses. |
| courses pointer of Course Class | Course array (dynamically created using courseCount) |

Member Functions:

a. getters and setters for each data members.
b. Default constructor.

c. Copy constructor
   Semester(Semester & s)
d. Parametrized constructor
   Semester(string sc,int c, Course *courseArr)

After defining the classes, write the following global functions in Semester.cpp file:

1. **GetCreditHoursCount**: receives a semester as parameter and returns the total number of credit hours registered in it.
3. **FindCourseInSemesterRegistration**: receives a semester, and a course code as parameters and returns true if the course is registered in the semester.

# Task 8

Your goal in this question is to write a program to manage a ShoppingList. For this you will need to create a class ShoppingList that should be able to store all the details of shopping items with the following functionalities:
Your ShoppingList have a fixed capacity (10) of ShoppingItems. For each **ShoppingItem** you will store its name, its quantity and its price. A new ShoppingItem cannot be added to ShoppingList if the list is already full.
Whenever you buy a new item you will add it to the shoppinglist if the capacity is not full.
Write classes for **ShoppingList** and **ShoppingItems**. Identify all the data members and member functions of these structures and write them. For instance, your class must provide, apart from other functions, following interface (public) functions.

| **ShoppingList** | |
|---|---|
| Member Function | Description |
| Default constructor | Sets all the member to reasonable (default) values. |
| AddItem | Add a new shopping item to the shopping list. |
| Print | Should print the current list of added items to list |
| TotalCost | Should print the total cost of shopping list |
| **ShoppingItem** | |
| Member Function | Description |
| **ShoppingItem (**constructor**)** | Add details for the current item. |
| Display | Should display the information of item |