**CS217 Object Oriented Programming**

**Task 1:**

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 2:**

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 3:**

* + - 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:

* + - * 1. getters and setters for each data members.
        2. Default constructor
        3. Copy constructor

Course(Course & c)

* + - * 1. Parametrized constructor

Course(string cc, string ct, int ch,char s, int rc)

* + - 1. 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:

* + - * 1. getters and setters for each data members.
        2. Default constructor.
        3. Copy constructor

Semester(Semester & s)

* + - * 1. Parametrized constructor

Semester(string sc,int c, Course \*courseArr)

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

**GetCreditHoursCount**: receives a semester as parameter and returns the total number of credit hours registered in it.

* + - 1. **FindCourseInSemesterRegistration**: receives a semester, and a course code as parameters and returns true if the course is registered in the semester.

**Task4:**

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)**
* **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".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".