OOP Assignment # 01

Group Members: **Arslan Gohar, Zeeshan Allah Rakha**

Roll Number: **SU92-BSITM-F22-016, SU92-BSITM-F22-019**

Question 06:

Make a Circle class.

a. It has three attributes radius, the x and the y coordinates of its center.
b. Make a no argument constructor to initialize it's attributes to 0, and a three argument constructor to initialize with the fixed values given by user.
c. Make void setValues(float, float, float) functions to set x, y and radius.
d. Make float area() function, and a float circumference() function to return area and circumference.
e. Make void print() function to display x , y coordinates and radius of a circle.

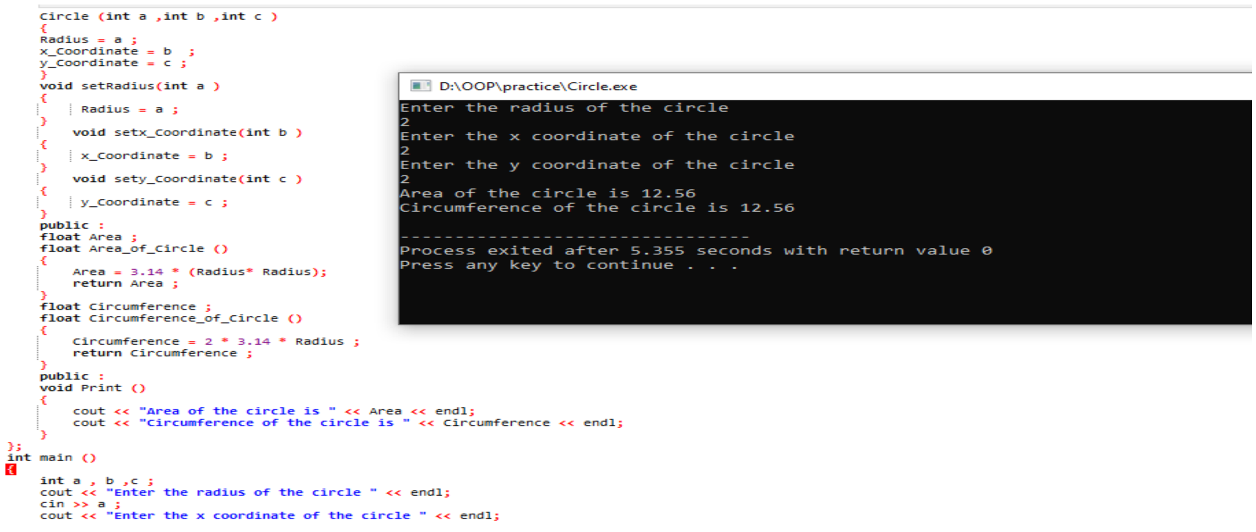
Call these functions in main() to test their working.

Code:

```
#include <iostream>

using namespace std;

class Circle

{

        public :

        int Radius;

        int x_Coordinate ;

        int y_Coordinate;

Circle ()

        {

        Radius = 0 ;

        x_Coordinate = 0 ;

        y_Coordinate = 0 ;

        }

        Circle (int a ,int b ,int c )
```

```
{
Radius = a ;
x_Coordinate = b  ;
y_Coordinate = c ;
}
void setRadius(int a )
{
   Radius = a ;
}
        void setx_Coordinate(int b )
{
   x_Coordinate = b ;
}
        void sety_Coordinate(int c )
{
   y_Coordinate = c ;
}
public :
float Area ;
float Area_of_Circle ()
{
        Area = 3.14 * (Radius* Radius);
        return Area ;
}
float Circumference ;
float Circumference_of_Circle ()
{
        Circumference = 2 * 3.14 * Radius ;
```

```
                return Circumference ;
        }
        public :
        void Print ()
        {
                cout << "Area of the circle is " << Area << endl;
                cout << "Circumference of the circle is " << Circumference << endl;
        }
};
int main ()
{
        int a , b ,c ;
        cout << "Enter the radius of the circle " << endl;
        cin >> a ;
        cout << "Enter the x coordinate of the circle " << endl;
        cin >> b ;
        cout << "Enter the y coordinate of the circle " << endl;
        cin >> c ;
        Circle C (a, b , c ) ;
        C.Area_of_Circle();
        C.Circumference_of_Circle();
        C.Print();
}
```



```
    Circle (int a ,int b ,int c )
    {
    Radius = a ;
    x_Coordinate = b   ;
    y_Coordinate = c ;
    }
    void setRadius(int a )
    {
        Radius = a ;
    }
        void setx_Coordinate(int b )
    {
        x_Coordinate = b ;
    }
        void sety_Coordinate(int c )
    {
        y_Coordinate = c ;
    }
    public :
    float Area ;
    float Area_of_Circle ()
    {
        Area = 3.14 * (Radius* Radius);
        return Area ;
    }
    float Circumference ;
    float Circumference_of_Circle ()
    {
        Circumference = 2 * 3.14 * Radius ;
        return Circumference ;
    }
    public :
    void Print ()
    {
        cout << "Area of the circle is " << Area << endl;
        cout << "Circumference of the circle is " << Circumference << endl;
    }
};
int main ()
{
    int a , b ,c ;
    cout << "Enter the radius of the circle " << endl;
    cin >> a ;
    cout << "Enter the x coordinate of the circle " << endl;
```

D:\OOP\practice\Circle.exe

```
Enter the radius of the circle
2
Enter the x coordinate of the circle
2
Enter the y coordinate of the circle
2
Area of the circle is 12.56
Circumference of the circle is 12.56

--------------------------------
Process exited after 5.355 seconds with return value 0
Press any key to continue . . .
```

Question 05:

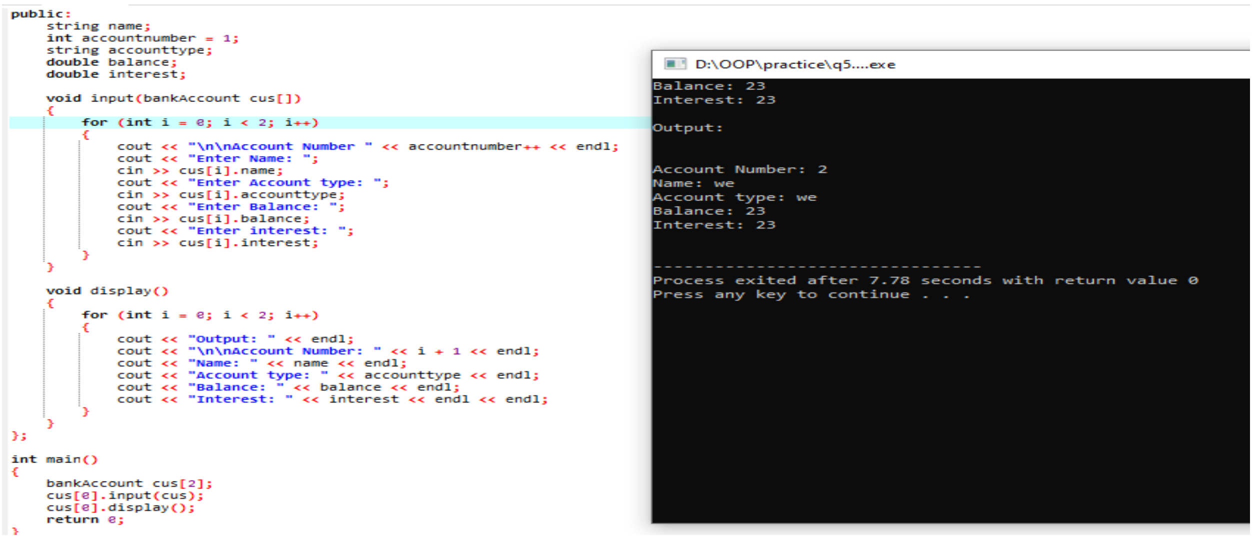
Define the class bankAccount to implement the basic properties of a bank account. An object of this class should store the following data: Account holder's name (string), account number (int), account type (string, checking/saving), balance (double), and interest rate (double). (Store interest rate as a decimal number.) Add appropriate member functions to manipulate an object. Use a static member in the class to automatically assign account numbers. Also declare an array of 10 components of type bankAccount to process up to 10 customers and write a program to illustrate how to use your class.

Code:

```
#include <iostream>

using namespace std;

class bankAccount

{

public:

   string name;

   int accountnumber = 1;

   string accounttype;

   double balance;

   double interest;


   void input(bankAccount cus[])
```

```
    {
        for (int i = 0; i < 10; i++)
        {
            cout << "\n\nAccount Number " << accountnumber++ << endl;
            cout << "Enter Name: ";
            cin >> cus[i].name;
            cout << "Enter Account type: ";
            cin >> cus[i].accounttype;
            cout << "Enter Balance: ";
            cin >> cus[i].balance;
            cout << "Enter interest: ";
            cin >> cus[i].interest;
        }
    }

    void display()
    {
        for (int i = 0; i < 10; i++)
        {
            cout << "\n\nAccount Number: " << i + 1 << endl;
            cout << "Name: " << name << endl;
            cout << "Account type: " << accounttype << endl;
            cout << "Balance: " << balance << endl;
            cout << "Interest: " << interest << endl << endl;
        }
    }
};
```

```
int main()

{

   bankAccount cus[10];

   cus[0].input(cus);

   cus[0].display();

   return 0;

}
```



```
public:
    string name;
    int accountnumber = 1;
    string accounttype;
    double balance;
    double interest;

    void input(bankAccount cus[])
    {
        for (int i = 0; i < 2; i++)
        {
            cout << "\n\nAccount Number " << accountnumber++ << endl;
            cout << "Enter Name: ";
            cin >> cus[i].name;
            cout << "Enter Account type: ";
            cin >> cus[i].accounttype;
            cout << "Enter Balance: ";
            cin >> cus[i].balance;
            cout << "Enter interest: ";
            cin >> cus[i].interest;
        }
    }

    void display()
    {
        for (int i = 0; i < 2; i++)
        {
            cout << "Output: " << endl;
            cout << "\n\nAccount Number: " << i + 1 << endl;
            cout << "Name: " << name << endl;
            cout << "Account type: " << accounttype << endl;
            cout << "Balance: " << balance << endl;
            cout << "Interest: " << interest << endl << endl;
        }
    }
};

int main()
{
    bankAccount cus[2];
    cus[0].input(cus);
    cus[0].display();
    return 0;
}
```

D:\OOP\practice\q5....exe

```
Balance: 23
Interest: 23

Output:


Account Number: 2
Name: we
Account type: we
Balance: 23
Interest: 23


--------------------------------
Process exited after 7.78 seconds with return value 0
Press any key to continue . . .
```

Question 02:

Write a program that converts a number entered in Roman numerals to a decimal. Your program should consist of a class, say, romanType. An object of type romanType should do the following:

a. Store the number as a Roman numeral.
b. Convert and store the number into decimal form.
c. Print the number as a Roman numeral or decimal number as requested by the user.

The decimal values of the Roman numerals are:

M	1000
D	500
C	100
L	50
X	10
V	5
I	1

Code:

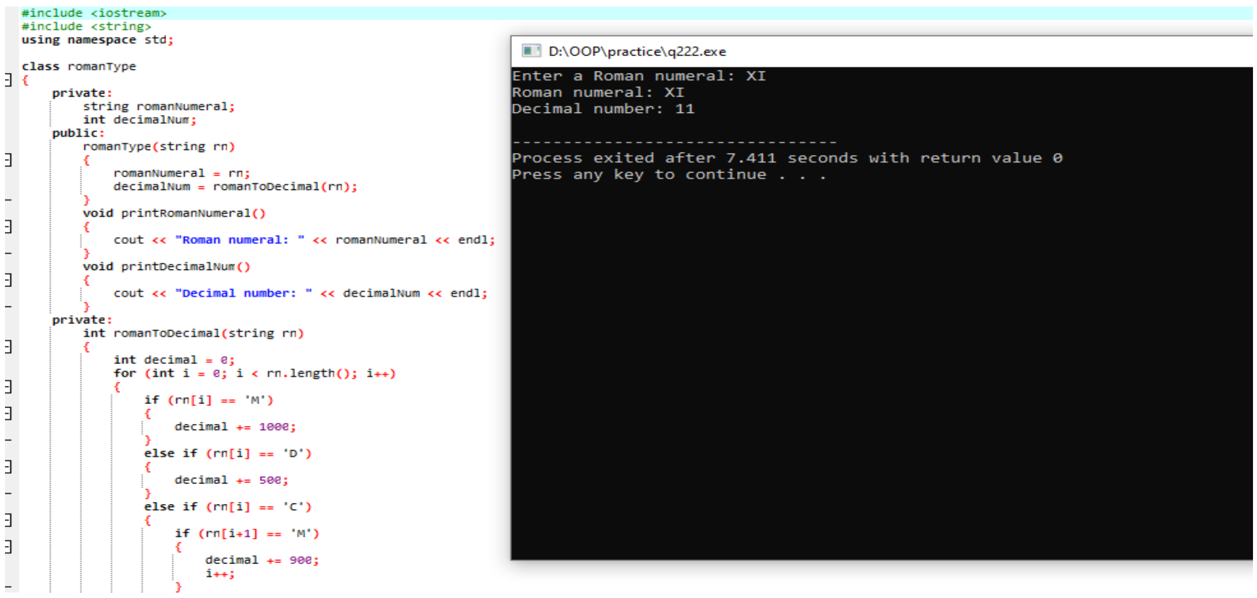
```
#include <iostream>
```

```
#include <string>
using namespace std;

class romanType
{
  private:
    string romanNumeral;    // the Roman numeral
    int decimalNum;         // the decimal equivalent
  public:
    romanType(string rn)
                {// constructor to set Roman numeral
      romanNumeral = rn;
      decimalNum = romanToDecimal(rn);
    }
    void printRomanNumeral()
                { // print Roman numeral
      cout << "Roman numeral: " << romanNumeral << endl;
    }
    void printDecimalNum()
                { // print decimal number
      cout << "Decimal number: " << decimalNum << endl;
    }
  private:
    int romanToDecimal(string rn)
                { // convert Roman numeral to decimal number
      int decimal = 0;
      for (int i = 0; i < rn.length(); i++)
                        {
```

```
if (rn[i] == 'M')
                              {
   decimal += 1000;
}
else if (rn[i] == 'D')
                              {
   decimal += 500;
}
else if (rn[i] == 'C')
                              {
   if (rn[i+1] == 'M')
                                        {
      decimal += 900;
      i++;
   }
        else if (rn[i+1] == 'D')
                                        {
           decimal += 400;
           i++;
        }
        else
        {
           decimal += 100;
        }
}
else if (rn[i] == 'L')
                              {
   decimal += 50;
```

```
}
else if (rn[i] == 'X')
                              {
   if (rn[i+1] == 'C')
                                        {
      decimal += 90;
      i++;
   }
   else if (rn[i+1] == 'L')
                                        {
      decimal += 40;
      i++;
   }
   else {
      decimal += 10;
   }
}
else if (rn[i] == 'V')
                              {
   decimal += 5;
}
else if (rn[i] == 'I')
                              {
   if (rn[i+1] == 'X')
                                        {
      decimal += 9;
      i++;
   }
```

```
                    else if (rn[i+1] == 'V')
                                                    {
                      decimal += 4;
                      i++;
                    }
                    else
                                                    {
                      decimal += 1;
                    }
                  }
                }
                return decimal;
              }
        };

        int main()
        {
          string rn;
          cout << "Enter a Roman numeral: ";
          cin >> rn;
          romanType roman(rn);
          roman.printRomanNumeral();
          roman.printDecimalNum();
          return 0;
        }
```



```
#include <iostream>
#include <string>
using namespace std;

class romanType
{
    private:
        string romanNumeral;
        int decimalNum;
    public:
        romanType(string rn)
        {
            romanNumeral = rn;
            decimalNum = romanToDecimal(rn);
        }
        void printRomanNumeral()
        {
            cout << "Roman numeral: " << romanNumeral << endl;
        }
        void printDecimalNum()
        {
            cout << "Decimal number: " << decimalNum << endl;
        }
    private:
        int romanToDecimal(string rn)
        {
            int decimal = 0;
            for (int i = 0; i < rn.length(); i++)
            {
                if (rn[i] == 'M')
                {
                    decimal += 1000;
                }
                else if (rn[i] == 'D')
                {
                    decimal += 500;
                }
                else if (rn[i] == 'C')
                {
                    if (rn[i+1] == 'M')
                    {
                        decimal += 900;
                        i++;
                    }
```

D:\OOP\practice\q222.exe

```
Enter a Roman numeral: XI
Roman numeral: XI
Decimal number: 11

--------------------------------
Process exited after 7.411 seconds with return value 0
Press any key to continue . . .
```

Question 01:

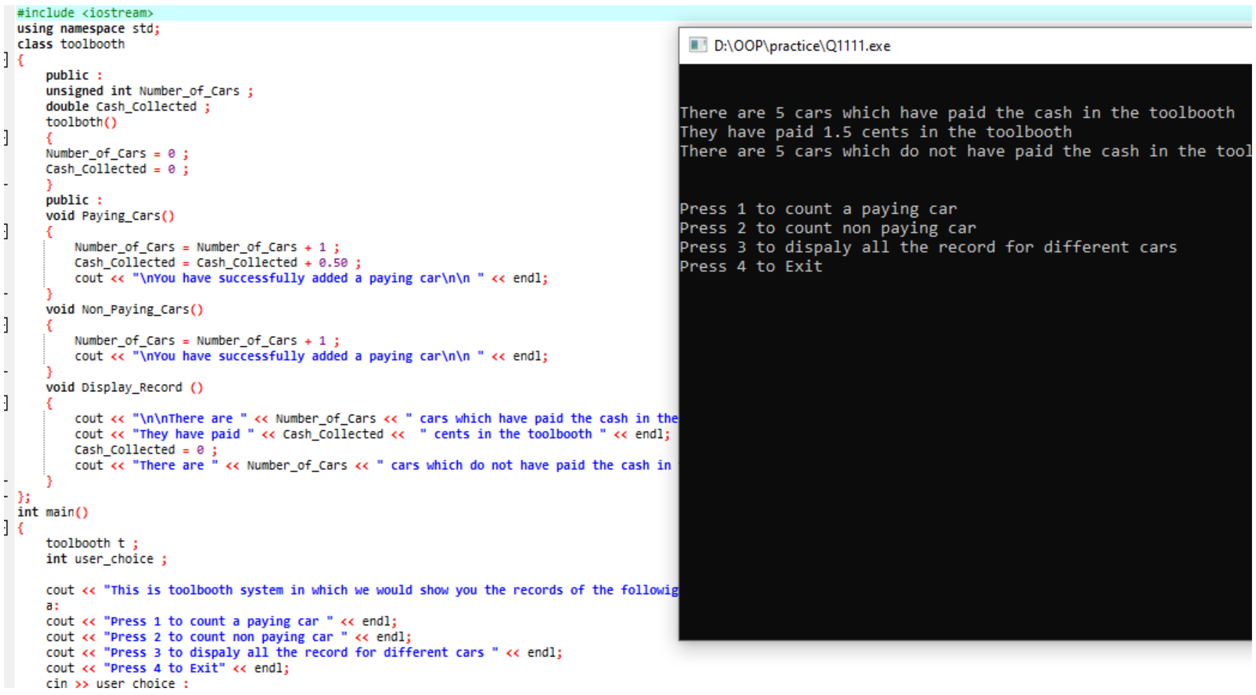
Imagine a tollbooth at a bridge. Cars passing by the booth are expected to pay a 50 cent toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth keeps track of the number of cars that have gone by, and of the total amount of money collected. Model this tollbooth with a class called 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.

Code:

```
#include <iostream>

using namespace std;

class toolbooth

{

        public :

        unsigned int Number_of_Cars ;

        double Cash_Collected ;

        toolboth()
```

```
	{
	Number_of_Cars = 0 ;
	Cash_Collected = 0 ;
	}
	public :
	void Paying_Cars()
	{
		Number_of_Cars = Number_of_Cars + 1 ;
		Cash_Collected = Cash_Collected + 0.50 ;
		cout << "You have successfully added a paying car " << endl;
	}
	void Non_Paying_Cars()
	{
		Number_of_Cars = Number_of_Cars + 1 ;
		cout << "You have successfully added a paying car " << endl;
	}
	void Display_Record ()
	{
		cout << "There are " << Number_of_Cars << " cars which have paid the cash in the
toolbooth" << endl;
		cout << "They have paid " << Cash_Collected <<  " cents in the toolbooth " << endl;
		Cash_Collected = 0 ;
		cout << "There are " << Number_of_Cars << " cars which do not have paid the cash
in the toolbooth" << endl;
	}
};
int main()
{
	toolbooth t ;
```

```
    int user_choice ;



    cout << "This is toolbooth system in which we would show you the records of the followign
cars :\n1.Paying Cars\n2.Non Paying Cars" << endl;
            cout << "Press 1 to count a paying car " << endl;
            cout << "Press 2 to count non paying car " << endl;
            cout << "Press 3 to dispaly all the record for different cars " << endl;
            cin >> user_choice ;
            switch(user_choice)
            {
                        case 1:
                                    system("cls");
                        t.Paying_Cars();
                        break;
                        case 2:
                                    system("cls");
                        t.Non_Paying_Cars();
                        break;
                        case 3:
                                    system("cls");
                        t.Display_Record();
                        default
                        cout << "invalid" << endl;
            }
}
}
```



```
#include <iostream>
using namespace std;
class toolbooth
{
    public :
    unsigned int Number_of_Cars ;
    double Cash_Collected ;
    toolboth()
    {
    Number_of_Cars = 0 ;
    Cash_Collected = 0 ;
    }
    public :
    void Paying_Cars()
    {
        Number_of_Cars = Number_of_Cars + 1 ;
        Cash_Collected = Cash_Collected + 0.50 ;
        cout << "\nYou have successfully added a paying car\n\n " << endl;
    }
    void Non_Paying_Cars()
    {
        Number_of_Cars = Number_of_Cars + 1 ;
        cout << "\nYou have successfully added a paying car\n\n " << endl;
    }
    void Display_Record ()
    {
        cout << "\n\nThere are " << Number_of_Cars << " cars which have paid the cash in the
        cout << "They have paid " << Cash_Collected <<  " cents in the toolbooth " << endl;
        Cash_Collected = 0 ;
        cout << "There are " << Number_of_Cars << " cars which do not have paid the cash in
    }
};
int main()
{
    toolbooth t ;
    int user_choice ;

    cout << "This is toolbooth system in which we would show you the records of the followig
    a:
    cout << "Press 1 to count a paying car " << endl;
    cout << "Press 2 to count non paying car " << endl;
    cout << "Press 3 to dispaly all the record for different cars " << endl;
    cout << "Press 4 to Exit" << endl;
    cin >> user_choice ;
```

D:\OOP\practice\Q1111.exe

```
There are 5 cars which have paid the cash in the toolbooth
They have paid 1.5 cents in the toolbooth
There are 5 cars which do not have paid the cash in the tool


Press 1 to count a paying car
Press 2 to count non paying car
Press 3 to dispaly all the record for different cars
Press 4 to Exit
```

Question 07:

Create a class **Rectangle** with attributes **length** and **width**

a. Make a no argument constructor to initialize attributes to 1
b. Also make a two argument constructor.
c. Make member functions that calculate and return the perimeter and the area of the rectangle.
d. Also, provide void **set(int l, int w)** and **void get()** functions for the length and width attributes.
e. Make a Draw function that draws a rectangle using a character * on console.

```
*********
*       *
*       *
*********
```

Code:

#include <iostream>

using namespace std;

class Rectangle

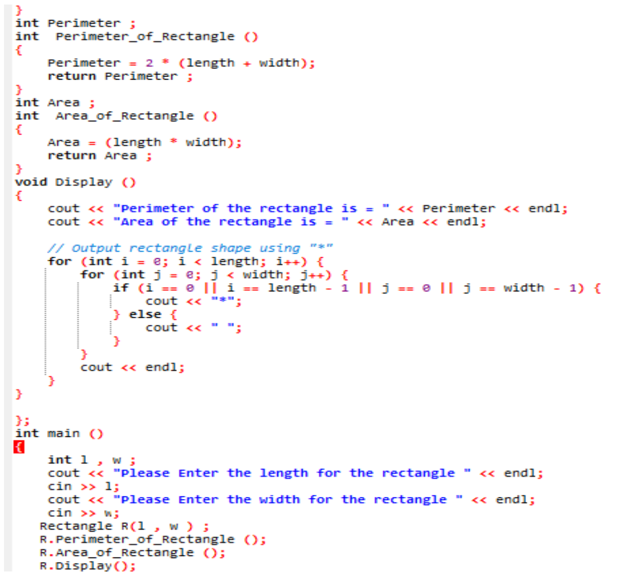
{

public:

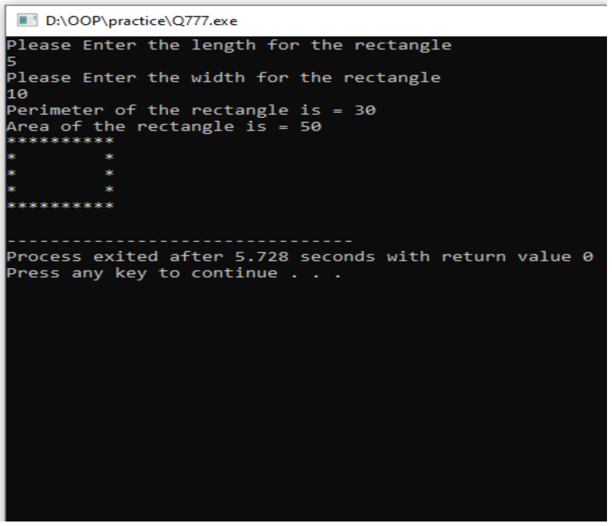
```
            int length ;
            int width ;
   Rectangle (int l , int w )
   {
            length = l ;
            width = w ;
            }
void set_Length ( int a )
{
            length = a ;
}

void set_Width ( int b )
{
            width = b ;
}
int Perimeter ;
int  Perimeter_of_Rectangle ()
{
            Perimeter = 2 * (length + width);
            return Perimeter ;
}
int Area ;
int  Area_of_Rectangle ()
{
            Area = (length * width);
            return Area ;
}
```

```
void Display ()
{
    cout << "Perimeter of the rectangle is = " << Perimeter << endl;
    cout << "Area of the rectangle is = " << Area << endl;

    // Output rectangle shape using "*"
    for (int i = 0; i < length; i++) {
        for (int j = 0; j < width; j++) {
            if (i == 0 || i == length - 1 || j == 0 || j == width - 1) {
                cout << "*";
            } else {
                cout << " ";
            }
        }
        cout << endl;
    }
}


};
int main ()
{
             int l , w ;
             cout << "Please Enter the length for the rectangle " << endl;
             cin >> l;
             cout << "Please Enter the width for the rectangle " << endl;
             cin >> w;
   Rectangle R(l , w ) ;
   R.Perimeter_of_Rectangle ();
```

```
  R.Area_of_Rectangle ();

  R.Display();

}
```



```
}
int Perimeter ;
int  Perimeter_of_Rectangle ()
{
    Perimeter = 2 * (length + width);
    return Perimeter ;
}
int Area ;
int  Area_of_Rectangle ()
{
    Area = (length * width);
    return Area ;
}
void Display ()
{
    cout << "Perimeter of the rectangle is = " << Perimeter << endl;
    cout << "Area of the rectangle is = " << Area << endl;

    // Output rectangle shape using "*"
    for (int i = 0; i < length; i++) {
        for (int j = 0; j < width; j++) {
            if (i == 0 || i == length - 1 || j == 0 || j == width - 1) {
                cout << "*";
            } else {
                cout << " ";
            }
        }
        cout << endl;
    }
}

};
int main ()
{
    int l , w ;
    cout << "Please Enter the length for the rectangle " << endl;
    cin >> l;
    cout << "Please Enter the width for the rectangle " << endl;
    cin >> w;
   Rectangle R(l , w ) ;
   R.Perimeter_of_Rectangle ();
   R.Area_of_Rectangle ();
   R.Display();
```



D:\OOP\practice\Q777.exe

```
Please Enter the length for the rectangle
5
Please Enter the width for the rectangle
10
Perimeter of the rectangle is = 30
Area of the rectangle is = 50
**********
*        *
*        *
*        *
**********


--------------------------------
Process exited after 5.728 seconds with return value 0
Press any key to continue . . .
```

Question 03:

Design and implement a class dayType that implements the day of the week in a program. The class dayType should store the day, such as Sun for Sunday. The program should be able to perform the following operations on an object of type dayType:

a. Set the day.
b. Print the day.
c. Return the day.
d. Return the next day.
e. Return the previous day.
f. Calculate and return the day by adding certain days to the current day. For example, if the current day is Monday and we add 4 days, the day to be returned is Friday. Similarly, if today is Tuesday and we add 13 days, the day to be returned is Monday.
g. Add the appropriate constructors.

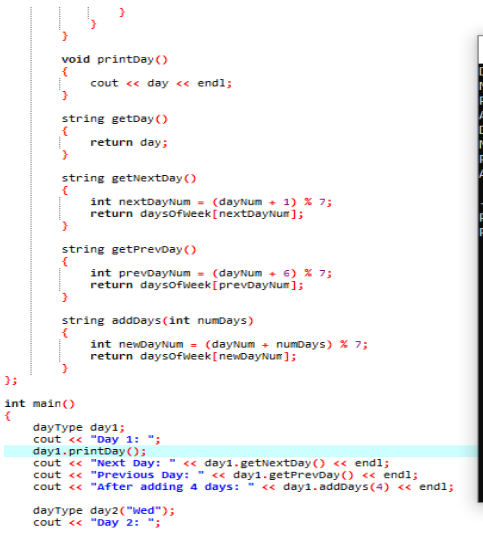
Code:

```
#include<iostream>

#include<string>

using namespace std;

class dayType {

   private:

      string day;
```

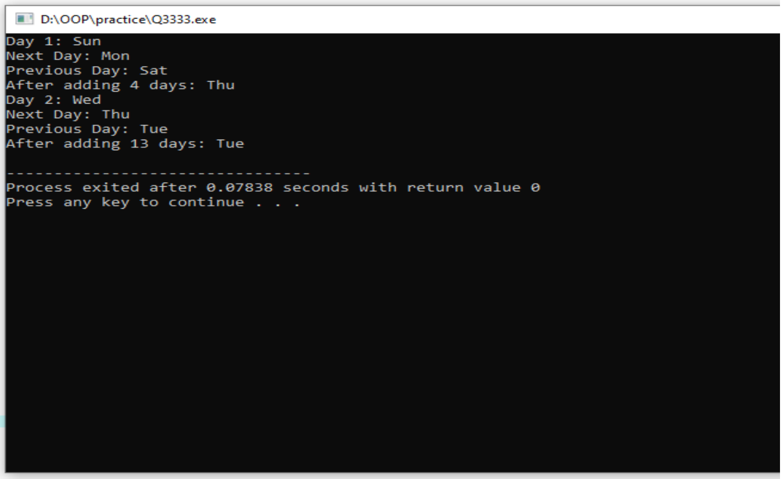
```
    int dayNum;
    string daysOfWeek[7] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};

public:
    dayType()
                    {
        day = "Sun";
        dayNum = 0;
    }

    dayType(string d)
                    {
        setDay(d);
    }

    void setDay(string d)
                    {
        for(int i=0; i<7; i++)
                                {
            if(d == daysOfWeek[i])
                                            {
                day = d;
                dayNum = i;
                break;
            }
        }
    }
```

```
         void printDay()
                    {
        cout << day << endl;
     }

     string getDay()
                    {
        return day;
     }

     string getNextDay()
                    {
        int nextDayNum = (dayNum + 1) % 7;
        return daysOfWeek[nextDayNum];
     }

     string getPrevDay()
                    {
        int prevDayNum = (dayNum + 6) % 7;
        return daysOfWeek[prevDayNum];
     }

     string addDays(int numDays)
                    {
        int newDayNum = (dayNum + numDays) % 7;
        return daysOfWeek[newDayNum];
     }
};
```

```
int main()

{

    dayType day1;

    cout << "Day 1: ";

    day1.printDay();

    cout << "Next Day: " << day1.getNextDay() << endl;

    cout << "Previous Day: " << day1.getPrevDay() << endl;

    cout << "After adding 4 days: " << day1.addDays(4) << endl;


    dayType day2("Wed");

    cout << "Day 2: ";

    day2.printDay();

    cout << "Next Day: " << day2.getNextDay() << endl;

    cout << "Previous Day: " << day2.getPrevDay() << endl;

    cout << "After adding 13 days: " << day2.addDays(13) << endl;


    return 0;

}
```



```
                }
            }
        }

        void printDay()
        {
            cout << day << endl;
        }

        string getDay()
        {
            return day;
        }

        string getNextDay()
        {
            int nextDayNum = (dayNum + 1) % 7;
            return daysOfWeek[nextDayNum];
        }

        string getPrevDay()
        {
            int prevDayNum = (dayNum + 6) % 7;
            return daysOfWeek[prevDayNum];
        }

        string addDays(int numDays)
        {
            int newDayNum = (dayNum + numDays) % 7;
            return daysOfWeek[newDayNum];
        }
};

int main()
{
    dayType day1;
    cout << "Day 1: ";
    day1.printDay();
    cout << "Next Day: " << day1.getNextDay() << endl;
    cout << "Previous Day: " << day1.getPrevDay() << endl;
    cout << "After adding 4 days: " << day1.addDays(4) << endl;

    dayType day2("Wed");
    cout << "Day 2: ";
```



D:\OOP\practice\Q3333.exe

```
Day 1: Sun
Next Day: Mon
Previous Day: Sat
After adding 4 days: Thu
Day 2: Wed
Next Day: Thu
Previous Day: Tue
After adding 13 days: Tue

--------------------------------
Process exited after 0.07838 seconds with return value 0
Press any key to continue . . .
```

Question no 08:

Question 08:
(10 Points)

Create a class called time that has separate int member data for hours, minutes, and seconds.

a. One constructor should initialize this data to 0, and another constructor should initialize it to fixed values.
b. Make **void print()** to display time in 23:59:59 format.
c. Make **void setHour(int)** to set hours.
d. Make **void setminute(int)** to set minutes.
e. Make **void setSecond(int)** to set seconds.
f. Make **void setTime(int,int,int)** to set hour, minute, second
g. Make **int Hour(); int Minute(); int Second();** to return hours, minute and seconds respectively.
h. Include a **tick()** member function that increments the time stored in a Time object by one second.
i. An add member function that should add two objects of type time passed as arguments.
j. Be sure to test the following cases:
    - Incrementing into the next minute.
    - Incrementing into the next hour.
    - Incrementing into the next day (i.e., 23:59:59 to 00:00:00).

Make 1000 times loop in a main function. Call tick and print functions in that loop for an object. Also make two objects and add them to a third object and print their values.

Code:

```
#include<iostream>

using namespace std;


class Time

{

	int hours, mins, secs;


public:

	Time();
```

```
        Time(int h, int m, int s);

        void display();

        void sthour(int ho);

        void stminu(int mi);

        void stsec(int sec);

        void stTime(int hou, int min, int se);

        int hour();

        int min();

        int sec();

        void Tick();

        int Add(Time t1, Time t2);

};


Time::Time()

{

        hours = 0;

        mins = 0;

        secs = 0;

}


Time::Time(int h, int m, int s)

{

        hours = h;

        mins = m;

        secs = s;

}


void Time::display()
```

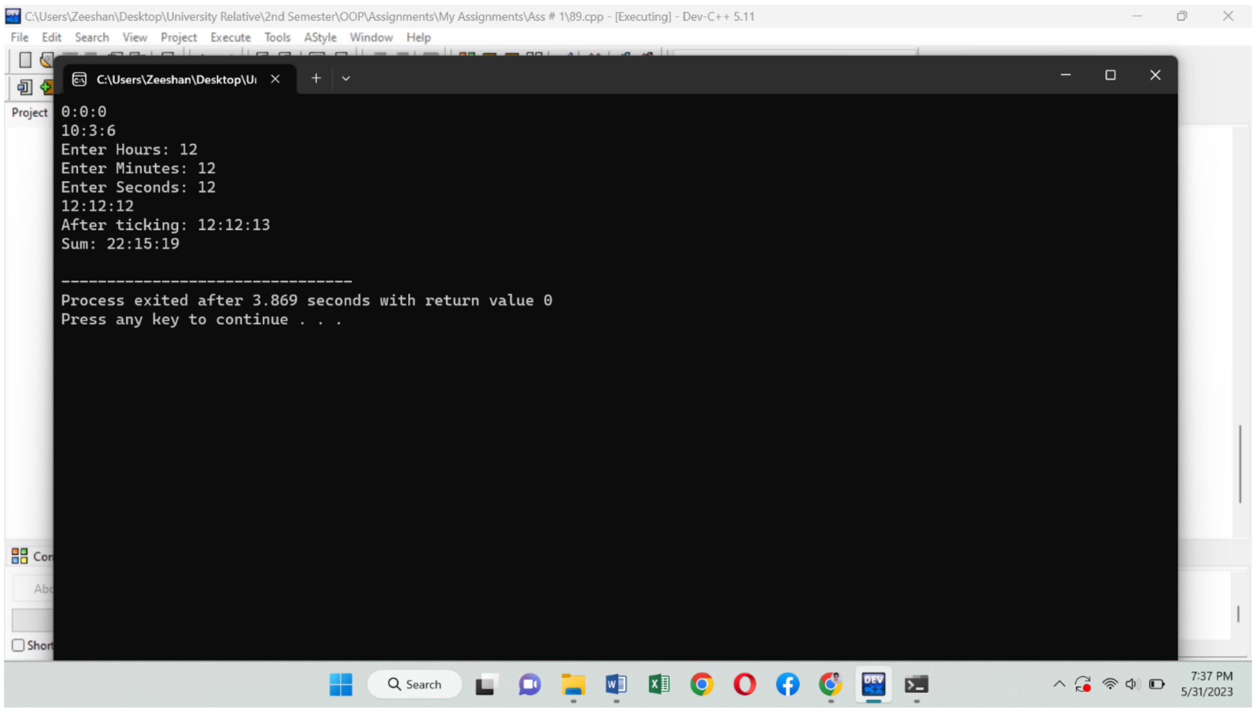
```
{
        cout << hours << ":" << mins << ":" << secs << endl;
}

void Time::sthour(int ho)
{
        hours = ho;
}

void Time::stminu(int mi)
{
        mins = mi;
}

void Time::stsec(int sec)
{
        secs = sec;
}

void Time::stTime(int hou, int min, int se)
{
        hours = hou;
        mins = min;
        secs = se;
}

int Time::hour()
{
    return hours;
}

int Time::min()
{
    return mins;
}

int Time::sec()
{
    return secs;
}

void Time::Tick()
{
    secs++;

    if (secs >= 60)
    {
        secs = 0;
        mins++;

        if (mins >= 60)
        {
            mins = 0;
            hours++;

            if (hours >= 24)
```

```
                    hours = 0;
            }
        }
}
int Time::Add(Time t1,Time t2)
{
        int h = t1.hour() + t2.hour();
        int m = t1.min() + t2.min();
        int s = t1.sec() + t2.sec();

        if (s >= 60)
        {
                m += s / 60;
                s %= 60;
        }

        if (m >= 60)
        {
                h += m / 60;
                m %= 60;
        }

        h %= 24; // To ensure the hours do not exceed 24

        cout << "Sum: " << h << ":" << m << ":" << s << endl;
}
```

```
int main()
{
	int ho, mi, sec;
	Time T1;
	T1.display();

	Time T2(10, 3, 6);
	T2.display();

	cout << "Enter Hours: ";
	cin >> ho;

	cout << "Enter Minutes: ";
	cin >> mi;

	cout << "Enter Seconds: ";
	cin >> sec;

	T1.sthour(ho);
	T1.stminu(mi);
	T1.stsec(sec);

	T1.display();

	T1.Tick();
	cout << "After ticking: ";
	T1.display();
```

```
        Time T3;

        T3.Add(T1,T2);


        return 0;

}
```

Output:


C:\Users\Zeeshan\Desktop\University Relative\2nd Semester\OOP\Assignments\My Assignments\Ass # 1\89.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\Zeeshan\Desktop\U
7:37 PM
5/31/2023

```
0:0:0
10:3:6
Enter Hours: 12
Enter Minutes: 12
Enter Seconds: 12
12:12:12
After ticking: 12:12:13
Sum: 22:15:19

--------------------------------
Process exited after 3.869 seconds with return value 0
Press any key to continue . . .
```

We have query in Question no 4.