Ex no:1(a)	VOTER'S ELIGIBLITY

Aim:

To create a console application in C# program for implementing voter's eligibility.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project. and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a parent class using class keyboard.

Step 7: Get the input from user.

Step 8: If age is greater than 18 the print able to vote.

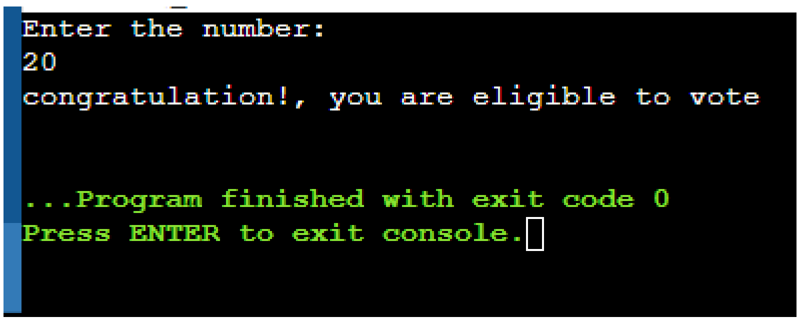
Step 9: Else print unable to vote.

Step 10: The output prints in the Console window.

Source Code:

```
using  System;
class HelloWorld {
  static void Main() {
    Console.WriteLine("Enter the number:");
    int age= int.Parse(Console.ReadLine());
    if(age<18){
    Console.WriteLine("unable to vote");
    }
    else{
        Console.WriteLine("congratulation!, you are eligible to vote");


    }

  }
```

Output:



```
Enter the number:
20
congratulation!, you are eligible to vote


...Program finished with exit code 0
Press ENTER to exit console.
```

Result:

Thus, the console application in C# for voter’s eligibility has been executed and output was verified successfully.

Ex no:1(b)	GREATEST OF THREE NUMBERS

Aim:

To create a console application in C# program to implement greatest of three numbers.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project. and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Read the input from the user.

Step 7: If n1 is greater than n2 and n3 then print n1 is greatest of all

Step 8: Elseif n2 is greater than n3 and n1 print n2 is greatest

Step 9: Else print n3 is greatest of all.

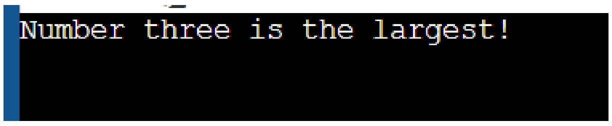
Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start Windows debugging from top level menu bar.

Step 11: The output prints in the Console window.

Source code:

```
using System;
namespace Demo {
  class MyApplication {
    static void Main(string[] args) {
      int num1, num2, num3;
      // set the value of the three numbers
      num1 = 10;
      num2 = 20;
      num3 = 50;
      if (num1 > num2)
      {  if (num1 > num3)


      {
          Console.Write("Number one is the largest!\n");

        } else {
          Console.Write("Number three is the largest!\n");
      }
    }
    else if (num2 > num3)
      Console.Write("Number two is the largest!\n");
    else
      Console.Write("Number three is the largest!\n");
    }
  }
}
```

Output:



```
Number three is the largest!
```

Result:

Thus, the console application in C# program to find the greatest of three numbers has been executed successfully and output is verified

Ex no:1(c)	COLLEGE ADMISSION ELIGIBILITY

Aim:

To create a console application in C# program to implement the college admission eligibility.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Get the input from user.

Step 7: If the total num is greater than 180 or n2+n3 is greater than 140 then print eligible for admission

Step 8: Else print not eligible .

Step 9: Run the code & see the results by passing ctrl+F5. Choose debug->start Windows debugging from top level menu bar.

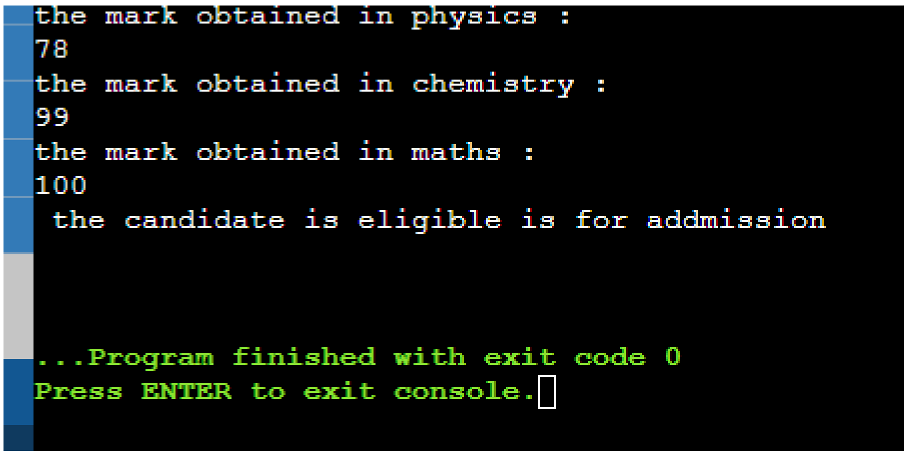
Step 10: The output prints in the Console window.

Source code:

```
using System;
class program {
static void Main() {
Console.WriteLine("the mark obtained in physics :");
int n1= int.Parse(Console.ReadLine());
Console.WriteLine("the mark obtained in chemistry :");
int n2= int.Parse(Console.ReadLine());
Console.WriteLine("the mark obtained in maths :");
int n3= int.Parse(Console.ReadLine());
int tot=n1+n2+n3;
if((n1>=55 && n2>=50 && n3>=65 && tot>=180)|| ((n2+n3)>=140)){
```

```
Console.WriteLine(" the candidate is eligible is for addmission \n");
    }
else{
Console.WriteLine(" not eligible\n");
    }
  }
}
```

Output:



```
the mark obtained in physics :
78
the mark obtained in chemistry :
99
the mark obtained in maths :
100
 the candidate is eligible is for addmission



...Program finished with exit code 0
Press ENTER to exit console.
```

Result:

Thus, the above C# program to implement the college admission eligibility has been executed successfully and output is verified.

Ex no:1(d)	VOWELS OR CONSONANTS

Aim:

To create a console application in C# to determine the given character is vowel or not.

Algorithm:

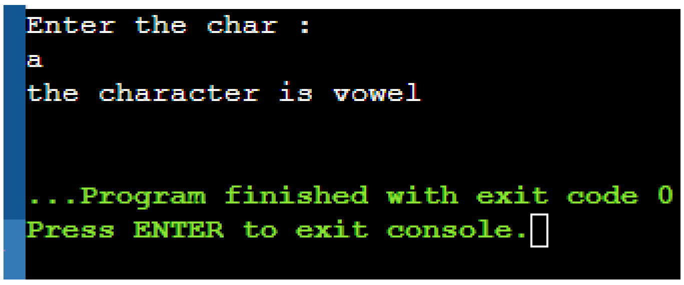
```
Step 1: Start the Visual Studio 2022 framework.
Step 2: On the menu bar, choose File->New->Project.
Step 3: Choose Visual C# from templates and then choose console.
Step 4: Specify a name for your project and click OK button.
Step 5: This creates a new project in Solution Explorer.
Step 6: Get the input from user.
Step 7: Use switch statement for determine vowel or not.
Step 8: Run the code & see the results by passing ctrl+F5. Choose debug->start
        Windowsdebugging from top level menu bar.
Step 9: The output prints in the Console window.
Step 10: Stop the program
```

Source code:

```
using System;
class program {
static void Main() {
Console.WriteLine("Enter the char :");
char c= char.Parse(Console.ReadLine().ToUpper());
switch(c){
case 'A':
Console.WriteLine("the character is vowel");
break;
case 'E':
Console.WriteLine("the character is vowel");
break;
case 'I':
```

```
Console.WriteLine("the character is vowel");
break;
case 'O':
Console.WriteLine("the character is vowel");
break;
case 'U':
Console.WriteLine("the character is vowel");
break;
default :
Console.WriteLine("the character is consonant");
break;
    }
  }
}
```

Output:



```
Enter the char :
a
the character is vowel


...Program finished with exit code 0
Press ENTER to exit console.
```

Result:

Thus, the above C# program to implement the given character is vowel or not has been executed successfully and output is verified.

Ex no:2(a)	WORD COUNT OF A STRING

Aim:

To create a console application in C# to implement the word count of a given string.

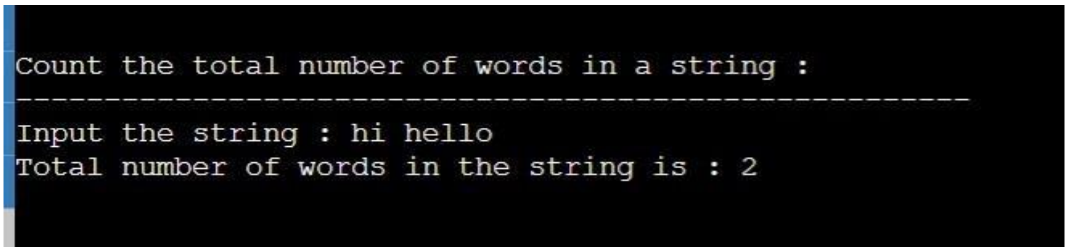
Algorithm:

```
Step 1: Start the Visual Studio 2022 framework.
Step 2: On the menu bar, choose File->New->Project.
Step 3: Choose Visual C# from templates and then choose console.
Step 4: Specify a name for your project and click OK button.
Step 5: This creates a new project in Solution Explorer.
Step 6: Get the input from user.
Step 7: Use variable.length() method to find length of word
Step 8: If there are slashes or spaces increment the word count .
Step 9: Run the code & see the results by passing ctrl+F5. Choose debug->start
        Windowsdebugging from top level menu bar.
Step 10: The output prints in the Console window.
Step 11: Stop the program
```

Source code:

```
using System;

namespace reverseString{
    class Program
    {
        static void Main(string[] args)
        {
            string str = "", reverse = "";
            int Length = 0;
            Console.WriteLine("Enter a Word");
            //Getting String(word) from Console
            str = Console.ReadLine();
            //Calculate length of string str
            Length = str.Length - 1;
            while(Length>=0)
{
    reverse = reverse + str[Length];
    Length--;
}
//Displaying the reverse word
Console.WriteLine("Reverse word is {0}",reverse);
Console.ReadLine()
}
}
```

OUTPUT:



```
Count the total number of words in a string :
------------------------------------------------------
Input the string : hi hello
Total number of words in the string is : 2
```

Result:

Thus, the above console application in C# program to implement word count of a given string has been executed successfully.

Ex no:2(b)	REVERSING A STRING

Aim:

To create a console application in C# program to implement the reversing a given string.

Algorithm:

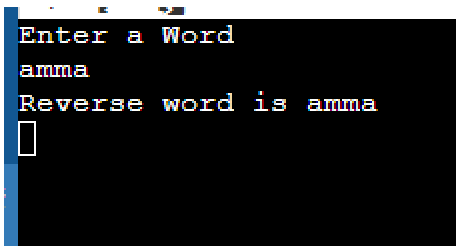
```
Step 1: Start the Visual Studio 2022 framework.
Step 2: On the menu bar, choose File->New->Project.
Step 3: Choose Visual C# from templates and then choose console.
Step 4: Specify a name for your project and click OK button.
Step 5: This creates a new project in Solution Explorer.
Step 6: Get the input from user.
Step 7: Use variable.length() method to find length of word
Step 8: Print the string in reverse order using for loop.
Step 9: Run the code & see the results by passing ctrl+F5. Choose debug->start
        Windowsdebugging from top level menu bar.
Step 10: The output prints in the Console window.
Step 11: Stop the program.
```

Source code:

```
 using System;
namespace reverseStrinG{
   class Program{
      static void Main(string[] args)
      {
         string str = "", reverse = "";
         int Length = 0;
         Console.WriteLine("Enter a Word");
         //Getting String(word) from Console
         str = Console.ReadLine();
         //Calculate length of string str
         Length = str.Length - 1;
         while(Length>=0)
```

```
                {
                   reverse = reverse + str[Length];
                   Length--;
                }
                //Displaying the reverse word
                Console.WriteLine("Reverse word is {0}",reverse);
                Console.ReadLine();
            }
}
```

Output:



```
Enter a Word
amma
Reverse word is amma
```

Result:

Thus, the above C# program to implement has been executed successfully and output is verified.

Ex no:2(c)	SUM AND AVERAGE OF AN ARRAY

Aim:

To create a console application in C# program to find the sum and average of an array.

Algorithm:

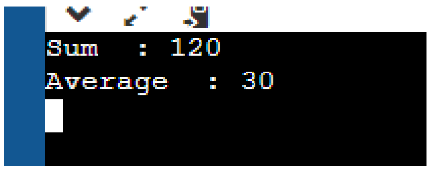
```
Step 1: Start the Visual Studio 2022 framework.
Step 2: On the menu bar, choose File->New->Project.
Step 3: Choose Visual C# from templates and then choose console.
Step 4: Specify a name for your project and click OK button.
Step 5: This creates a new project in Solution Explorer.
Step 6: Initialize the array
Step 7: Declare a variable to calculate array sum.
Step 8: Initialize for loop and calculate the sum.
Step 9: Print the sum & calculate the average of array.
Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start
         Windowsdebugging from top level menu bar.
Step 11: The output prints in the Console window.
Step 12: Stop the program
```

Source code:

```
using  System;
class HelloWorld {
static void Main()
      {
         double sum=0, avg=0;
         double[] numbers = { 10, 20, 50, 40};
         for(int i=0;i<numbers.Length;i++)
         {
            sum += numbers[i];
         }
          avg = sum / numbers.Length; Console.WriteLine("Sum : "+sum);
          Console.WriteLine("Average : "+avg);
```

```
            Console.ReadKey();
                }
                }
```

OUTPUT:



```
Sum   : 120
Average   : 30
```

Result:

Thus, the above C# program for finding the sum and average of an array has been executed successfully and output is verified.

Ex no:2(d)	MAXIMUM AND MINIMUM OF AN ARRAY

Aim:

To create a console application in C# program to find maximum and minimum of an array.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a variable min and max

Step 7: Initialize max and min =0

Step 8: If the element is smaller than min then update min , if the element the is greater than max then update max.

Step 9: Repeat the step3 using for loop

Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start Windowsdebugging from top level menu bar.

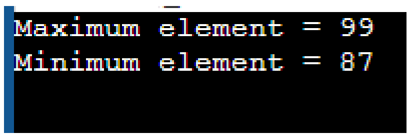
Step 11: The output prints in the Console window.

Step 12: Stop the program.

Source code:

```
using System;
 public class Demo {
 public static void Main() {
int[] arr = new int[5] {99, 95, 93, 89, 87};
int i, max, min, n;

n = 5;

max = arr[0]; min = arr[0];

for(i=1; i<n; i++) { if(arr[i]>max) { max = arr[i];
}
if(arr[i]<min) { min = arr[i];
}
}
Console.Write("Maximum element ={0}\n", max);
 Console.Write("Minimum element ={0}\n", min);
}
}
```

Output:



```
Maximum element = 99
Minimum element = 87
```

Result:

Thus, the above C# program to implement the maximum and minimum of an array has been executed successfully and output is verified.

Ex no:2(e)	STUDENT DETAILS USING CLASSES AND OBJECTS

Aim:

To create a console application in C# program to print student details using classes and objects.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Get the input from user.

Step 7: Create a parent class using class keyword

Step 8: Inherit the child class using derived_class : base_class syntax.

Step 9: Call the function using object.

Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start Windowsdebugging from top level menu bar.

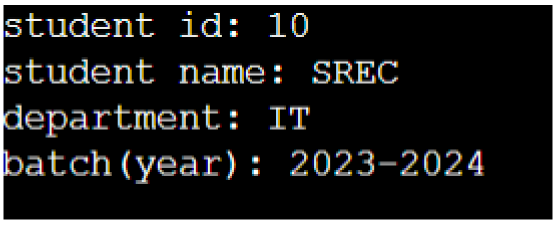
Step 11: The output prints in the Console window.

Step 12: Stop the program.

Source code:

```
using System;
public class Student
{
    public int id;
    public string name,batch,department;
    public void insert(int i, string b, string d, string n){
        id=i;
        name=n;
        batch=b;
        department=d;
    }
    public void display(){
        Console.WriteLine("student id: "+id);
        Console.WriteLine("student name: "+name);
        Console.WriteLine("department: "+department);
        Console.WriteLine("batch(year): "+batch);
    }
}
class HelloWorld {
  static void Main() {
      Student o = new Student();
      o.insert(10,"2023-2024","IT","SREC");
      o.display();
  }
}
```

Output:



```
student id: 10
student name: SREC
department: IT
batch(year): 2023-2024
```

Result:

Thus, the above C# program to print the student details using classes and objects has been executed successfully and output is verified.

Ex no:2(f)	EMPLOYEE DETAILS USING CLASSES AND OBJECTS

Aim:

To create a console application in C# program to implement employee details using class and objects.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Get the input from user.

Step 7: create a parent class using class keyword.

Step 8: Inherit the child class using derived_class: base_class syntax.

Step 9: Create a class for object.

Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start Windowsdebugging from top level menu bar.

Step 11: The output prints in the Console window.

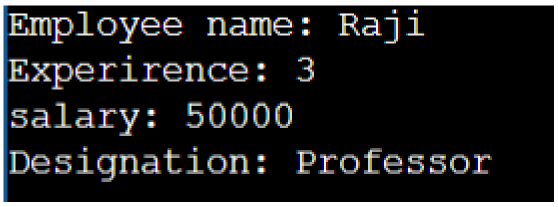
Step 12: Stop the program.

Source code:

```
using System;
public class Employee
{
   public int id,salary,experience;
   public string name,designation;
   public void insert(int i, int s, int e,string d, string n){
      id=i;
      name=n;
      experience=e;
      salary=s;
      designation=d;
   }
```

```
    public void display(){
        Console.WriteLine("Employee id: "+id);
        Console.WriteLine("Employee name: "+name);
        Console.WriteLine("Experirence: "+experience);
        Console.WriteLine("salary: "+salary);
        Console.WriteLine("Designation: "+designation);
    }
}
class HelloWorld {
  static void Main() {
      Employee o = new Employee();
      o.insert(40,50000,3,"Professor","Raji");
      o.display();
  }
}
```

Output:



```
Employee name: Raji
Experirence: 3
salary: 50000
Designation: Professor
```

Result:

Thus, the above C# program to print the employee details using class and objects has been executed successfully and output is verified.

Ex no:3(a)	SINGLE INHERITANCE

Aim:

To create a console application in C# program to implement the concept of single inheritance.

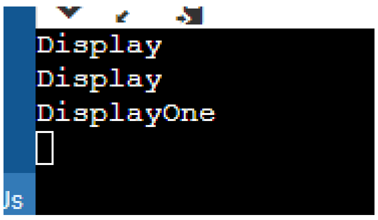
Algorithm:

```
Step 1: Start the Visual Studio 2022 framework.
Step 2: On the menu bar, choose File->New->Project.
Step 3: Choose Visual C# from templates and then choose console.
Step 4: Specify a name for your project and click OK button.
Step 5: This creates a new project in Solution Explorer.
Step 6: Create a parent class using class keyword.
Step 7: Inherit the child class using derived_class: base_class syntax.
Step 8: Create a class for object.
Step 9: Run the code & see the results by passing ctrl+F5. Choose debug->start
        Windowsdebugging from top level menu bar.
Step 10: The output prints in the Console window.
Step 11: Stop the program.
```

Source code:

```
using System;
namespace MyAplication {
  class Demo {
    static void Main(string[] args) {
      // Father class
      Father f = new Father();
      f.Display();
      // Son class
      Son s = new Son();
      s.Display();
      s.DisplayOne();
      Console.ReadKey();
    }
    class Father {
      public void Display() {
        Console.WriteLine("Display");
      }
      }
    class Son : Father {
      public void DisplayOne() {
        Console.WriteLine("DisplayOne");
      }
    }
  }
}
```

Output:



```
Display
Display
DisplayOne
```

Result:

Thus, the above C# program to implement the concept of single inheritance has been executed successfully and output is verified.

Ex no:3(b)	HIERARCHICAL INHERITANCE

Aim:

To create a console application in C# program to implement the concept of hierarchical inheritance.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a parent class using class keyword.

Step 7: Inherit the child class using derived _class: base_class syntax.

Step 8:Call the function using object .

Step 9: Run the code & see the results by passing ctrl+F5. Choose debug->start Windowsdebugging from top level menu bar.

Step 10: The output prints in the Console window.

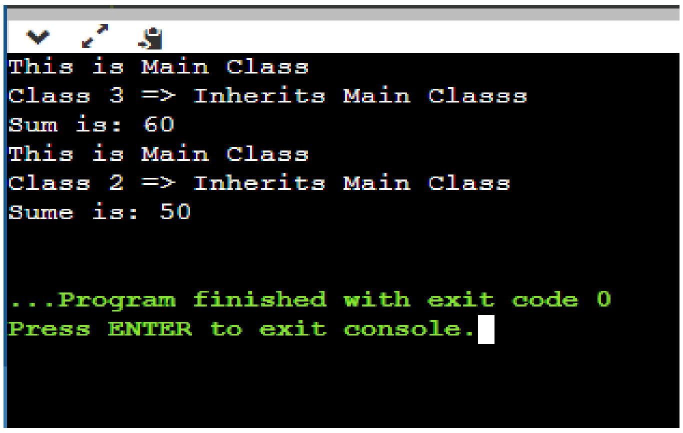
Step 11: Stop the program.

Source code:

```
using System;
namespace
Program
{
public class A
{
public void msg1(){
Console.WriteLine("This is Main Class");
}
public static int a = 20;
}

public class B : A{
public void msg2(){
Console.WriteLine("Class 2 => Inherits Main Class");
```

```
   int b = a + 30;
   Console.WriteLine("Sume is: "+b);
        }

 }
 public class C : A{ public void msg3(){
        Console.WriteLine("Class 3 => Inherits Main Classs ");
        int c = a + 40;
        Console.WriteLine("Sum is: "+ c);
    }
 }
 public class Final{
    public static void Main(string[] args){
        C obj1 = new C();
        obj1.msg1();
        obj1.msg3();
        B obj2 = new B();
        obj2.msg1();
        obj2.msg2();
    }
 }
```

Output:



```
This is Main Class
Class 3 => Inherits Main Classs
Sum is: 60
This is Main Class
Class 2 => Inherits Main Class
Sume is: 50


...Program finished with exit code 0
Press ENTER to exit console.
```

Result:

Thus, the above C# program to implement the concept of hierarchical inheritance has been executed successfully and output is verified.

Ex no:3(c)	MULTILEVEL INHERITANCE

Aim:

To create a console application in C# program to implement the concept of multilevel inheritance.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a parent class using class keyword.

Step 7: Inherit the child class using derived_class: base_class syntax.

Step 8: Create a class for object

Step 9: Call the function using object.

Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start Windowsdebugging from top level menu bar.

Step 11: The output prints in the Console window.

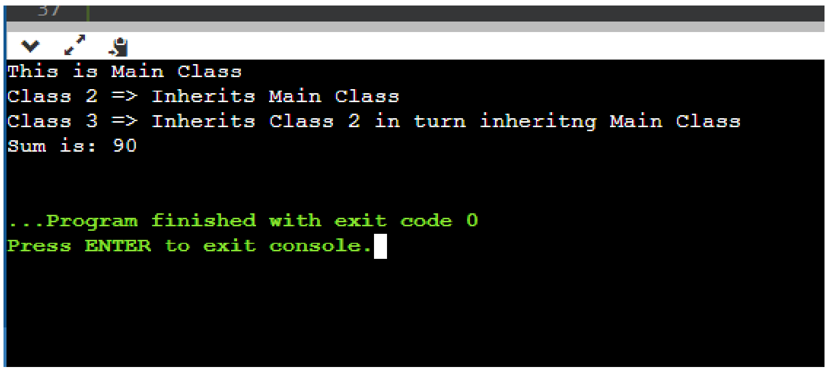
Step 12: Stop the program.

Source code:

```
using System;
 namespace Program
{
public class A
{
public void msg1(){ Console.WriteLine("This is Main Class");
}
public static int a = 20;
}
public class B : A{ public void msg2(){
Console.WriteLine("Class 2 => Inherits Main Class");
}
```

```
    public static int b = (A.a) + 30;
    }
    public class C : B{ public void msg3(){
    Console.WriteLine("Class 3 => Inherits Class 2 in turn inheritng Main Class ");
     int c = b + 40;
        Console.WriteLine("Sum is: "+ c);
     }
  }
  public class Final{
  public static void Main(string[] args){
   C obj1 = new C();
  obj1.msg1();
  obj1.msg2();
  obj1.msg3();
  }
    }
    }
```

Output:



```
This is Main Class
Class 2 => Inherits Main Class
Class 3 => Inherits Class 2 in turn inheritng Main Class
Sum is: 90


...Program finished with exit code 0
Press ENTER to exit console.
```

Result:

Thus, the above C# program to implement concept of multilevel inheritance has been executed successfully and output is verified.

Ex no:3(d)	INTERFACE

Aim:

To create a console application in C# program to implement the concept of interface.

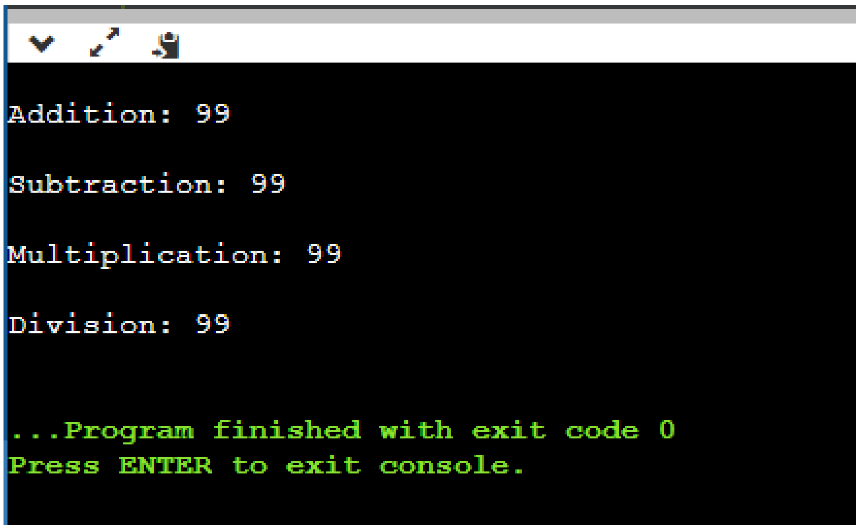
Algorithm:

```
Step 1: Start the Visual Studio 2022 framework.
Step 2: On the menu bar, choose File->New->Project.
Step 3: Choose Visual C# from templates and then choose console.
Step 4: Specify a name for your project and click OK button.
Step 5: This creates a new project in Solution Explorer.
Step 6: Create a parent class using class keyword.
Step 7: Inherit the child class using derived_class: base_class syntax.
Step 8: Create a class for object
Step 9: Call the function using object Step 10: Run the code & see the results by passing
        ctrl+F5. Choose debug->start Windowsdebugging from top level menu bar.
Step 10: The output prints in the Console window.
Step 11: Stop the program.
```

Source code:

```
using System;
namespace Program
{
interface addi {int plus (int a , int b);}
interface subt {int minus (int a1, int b1);}
interface mult {int cross (int a2, int b2);}
interface divi {int slash (int a3, int b3);}
public class operations : addi, subt, mult, divi { public int res1, res2, res3, res4;
public int plus(int a, int b){ return res1 = a + b;
}
public int minus(int a1, int b1){ return res2 = a1 - b1;
}
public int cross(int a2, int b2){ return res3 = a2 * b2;
}
public int slash(int a3, int b3){ return res4 = a3 / b3;
}
}
public class Final{
public static void Main(string[] args){ operations obj1 = new operations();
Console.WriteLine("\nAddition: " + obj1.plus(45, 54));
Console.WriteLine("\nSubtraction: " + obj1.minus(100, 1));
Console.WriteLine("\nMultiplication: " + obj1.cross(9, 11));
Console.WriteLine("\nDivision: " + obj1.slash(198, 2));
}
}
```

Output:



```
Addition: 99

Subtraction: 99

Multiplication: 99

Division: 99


...Program finished with exit code 0
Press ENTER to exit console.
```

Result:

Thus, the above C# program to implement the concept of interface has been executed successfully and output is verified.

Ex No: 4 a)	**METHOD OVERLOADING**
Date:	

Aim:

To create a console application in C# to implement the concept of Method Overloading.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a base class Building.

Step 7: Define the method area() to calculate the area of triangle.

Step 8: Override the method area() to calculate the area of cube.

Step 9: Invoke the methods using the object of the class.

Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start Windows debugging from top level menu bar.

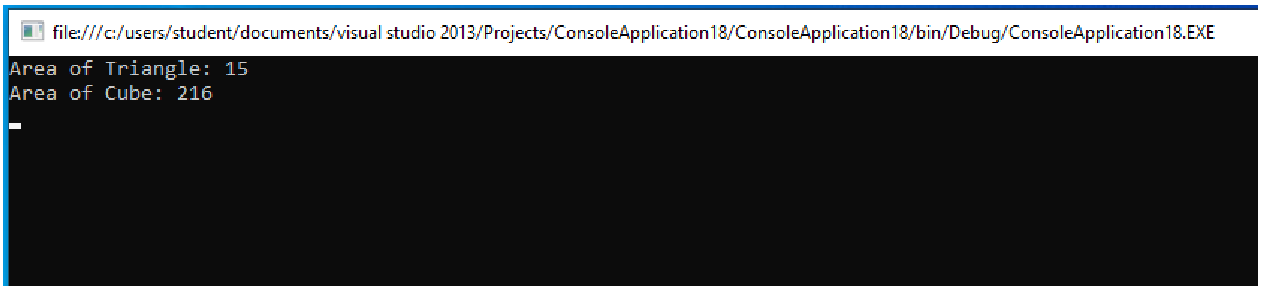
Step 11: The output prints in the console window.

Program:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication1
{
```

```
class Building
{
    public void area(int a){
         Console.WriteLine("Area of Cube:"+6 * a * a);
    }
    public void area(int a,int b){
       Console.WriteLine("Area of Triangle:"+0.5 * a * b);
    }
}
class Program
{
    static void Main(string[] args)
    {
        Cal c = new Cal();
        c.area(5);
        c.area(5,6);
        Console.ReadLine();
    }
}
```

OUTPUT:



file:///c:/users/student/documents/visual studio 2013/Projects/ConsoleApplication18/ConsoleApplication18/bin/Debug/ConsoleApplication18.EXE

```
Area of Triangle: 15
Area of Cube: 216
```

Result:

Thus, the console application in C# to implement the concept of Method Overloading has been executed and output was verified successfully.

Ex No: 4 b)	**METHOD OVERRIDING**
Date:	

Aim:

To create a console application in C# to implement the concept of Method Overriding.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a base class Shape and subclass Rectangle and Triangle.

Step 7: Define the method setDim() to assign the input variables.

Step 8: Create and define the method area() using virtual keyword in Shape class to display the statement Area of Triangle and Rectangle.

Step 9: Override the method area() in Rectangle and Triangle having the same parameters tocalculate the area of rectangle and area of triangle.

Step 10: Invoke the method area() of Shape class using its object.

Step 11: Invoke the methods of area() of Rectangle and Triangle class using its corresponding object.

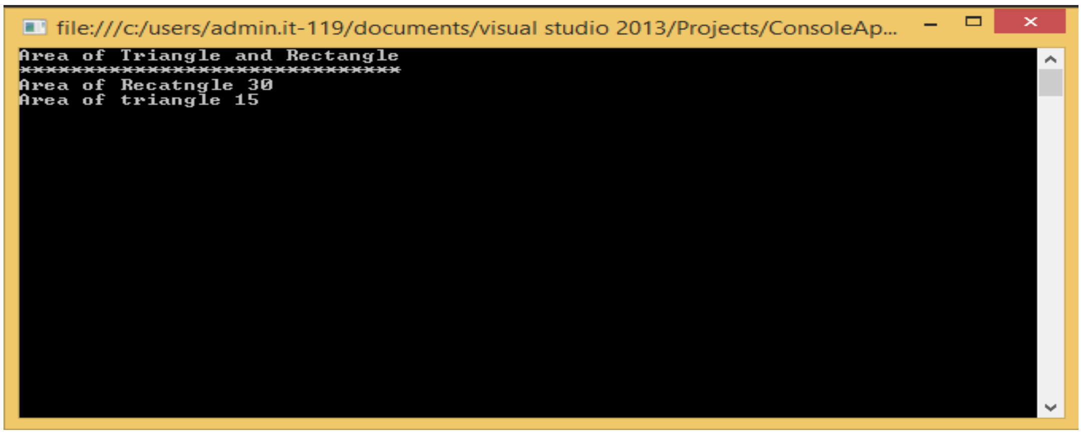
Step 12: Run the code & see the results by passing ctrl+F5. Choose debug->start windows debugging from top level menu bar.

Step 13: The output prints in the console window.

Program:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication2
{
    public class Shape
    {
        protected static int length;
        protected static int breadth;
        public void setDim(int l,int b)
        {
            length = l;
            breadth = b;
        }
        public virtual void area()
        {
            Console.WriteLine("Area of Triangle and Rectangle");
            Console.WriteLine("*****************************");
        }
    }
    public class Rectangle : Shape
    {
        public override void area()
        {
            Console.WriteLine("Area of Recatngle "+length * breadth);
        }
    }
    public class Triangle : Shape
    {
        public override void area()
        {
            Console.WriteLine("Area of triangle " + 0.5*length * breadth);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Shape s = new Shape();
            s.setDim(5, 6);
            s.area();
            Rectangle r = new Rectangle();
            Triangle t = new Triangle();
            r.area();
            t.area();
             Console.WriteLine("Area of Triangle:" + 1/2*c);
            Console.WriteLine("Area of Rectangle:" +d);
            Console.ReadLine();
        }
    }
}
```

Output:



file:///c:/users/admin.it-119/documents/visual studio 2013/Projects/ConsoleAp...

```
Area of Triangle and Rectangle
******************************
Area of Recatngle 30
Area of triangle 15
```

Result:

Thus, the console application in C# to implement the concept of Method Overriding has been executed and output was verified successfully.

Ex No: 4 c)	**OPERATOR OVERLOADING**
Date:	

Aim:

To create a console application in C# to implement the concept of Operator Overloading.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a class Program.

Step 7: Define the different operators like '+', '-', '*','/' to perform functions such as addition, subtraction, multiplication, and division respectively for the user defined class Complex.

Step 8: It adds the attributes of two Complex objects and returns the resultant Complex object to the main method where it invoked. Repeat Step 8 for subtraction, multiplication and division operations

Step 9: Create the method setData() to set values for real and imaginary. Perform the calculation and display the real part and imaginary part of the operations performed.

Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start windows debugging from top level menu bar.

Step 11: The output prints in the console window.

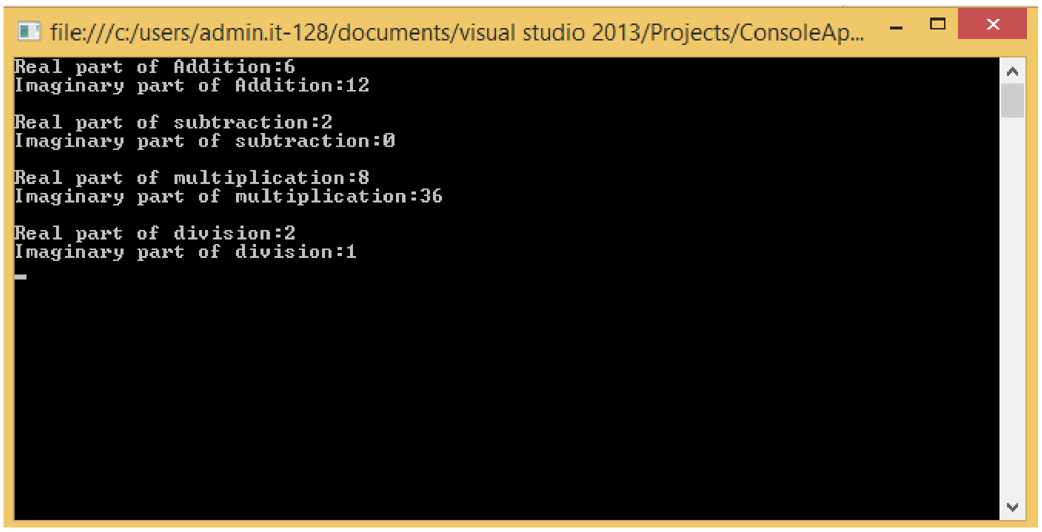
Program:

```
using System;
using System.Collections.Generic;

using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
class Program
    {
        static void Main(string[] args)
        {
           Complex a = new Complex();
           Complex b = new Complex();
           Complex c = new Complex();
           Complex d = new Complex();
           Complex e = new Complex();
           Complex f = new Complex();
           a.setData(4, 6);
           b.setData(2, 6);
           c = a + b;
           d = a - b;
           e = a * b;
           f = a / b;
        }
    }
 public class Complex
 {
    private int real, img;
    public void setData(int r,int i)
    {
        real = r;
        img = i;
    }
    public static Complex operator +(Complex a, Complex b)
    {
        Complex x = new Complex();
        x.real = a.real + b.real;
        x.img = a.img + b.img;
        Console.WriteLine("Real part of Addition:" + x.real);
        Console.WriteLine("Imaginary part of Addition:" + x.img);
        Console.ReadLine();
```

```
            return x;
        }
        public static Complex operator -(Complex a, Complex b)
        {
            Complex x = new Complex();
            x.real = a.real - b.real;
            x.img = a.img - b.img;
            Console.WriteLine("Real part of Subtraction:" + x.real);
            Console.WriteLine("Imaginary part of Subtraction:" + x.img);
            Console.ReadLine();
            return x;
        }
        public static Complex operator *(Complex a, Complex b)
        {
            Complex x = new Complex();
            x.real = a.real * b.real;
            x.img = a.img * b.img;
            Console.WriteLine("Real part of Multiplication:" + x.real);
            Console.WriteLine("Imaginary part of Multiplication:" + x.img);
            Console.ReadLine();
            return x;
        }
        public static Complex operator /(Complex a, Complex b)
        {
            Complex x = new Complex();
            x.real = a.real / b.real;
            x.img = a.img / b.img;
            Console.WriteLine("Real part of Division:" + x.real);
            Console.WriteLine("Imaginary part of Division:" + x.img);
            Console.ReadLine();
            return x;
        }

}
```

Output:



file:///c:/users/admin.it-128/documents/visual studio 2013/Projects/ConsoleAp...

```
Real part of Addition:6
Imaginary part of Addition:12

Real part of subtraction:2
Imaginary part of subtraction:0

Real part of multiplication:8
Imaginary part of multiplication:36

Real part of division:2
Imaginary part of division:1
_
```

Result:

Thus, the console application in C# to implement the concept of Operator Overloading has been executed and output was verified successfully.

Ex No: 4 d)	**EXCEPTION HANDLING**
Date:	

Aim:

To create a console application in C# to display the eligibility of the placement students using exception handling.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a class Program. Create a try() block to define a block of code to be tested for errors while executed.

Step 7: Inside the try block get the input elements from the user. If the average is greater than 50 print as batch 1. Else if average is between 35 and 50, then print it as batch 2. Else if average is between 20 and 35, print it as batch 3. Else print it as not eligible for placement.

Step 8: The catch() block is used to handle the exceptions in the program. When 0 is entered, it throws an exception denoting “Division by 0 is not possible”.

Step 9: Display “Thank You” in the finally block. The finally block executes once the try catch block terminates.

Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start windows debugging from top level menu bar.

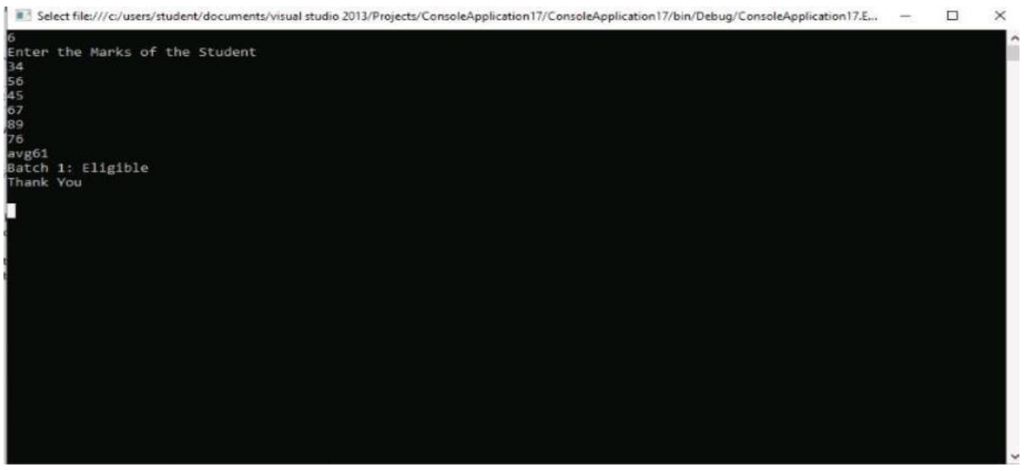
Step 12: The output prints in the console window.

Program:

```
using System;
using System.Collections.Generic;

using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication8
{
    class Program
    {
            static void Main(string[] args)
           {
            int n;
            n=Convert.ToInt32(Console.ReadLine());
            int[] mark =new int[n];
            int sum=0;
            Console.WriteLine("Enter the Marks of the Student");
             try{
            for (int i = 0; i < n; i++)
            {
                mark[i] = Convert.ToInt32(Console.ReadLine());
                sum += mark[i];
            }
            float avg = sum /n;
           if(avg<=100)
           {
                if(avg>50)
                {
                    Console.WriteLine("Batch 1");
                }
               else if(avg<=50 && avg>=35)
               {
                      Console.WriteLine("Batch 2");
                }
                else if(avg<35 && avg>=20)
               {
                       Console.WriteLine("Batch 3");
               }
```

```
          else if(avg<20 && avg >=10)
         {
               Console.WriteLine("Not eligible for placement");
         }
         else if(avg<10)
        {
              Console.WriteLine("Not eligible for Placement");
        }}}
catch
{
   Console.WriteLine("Divided By 0 Not Possible");
}
finally
{
   Console.WriteLine("Thank You");
}
    Console.ReadLine();
 }}
}
```

Output:



Select file:///c:/users/student/documents/visual studio 2013/Projects/ConsoleApplication17/ConsoleApplication17/bin/Debug/ConsoleApplication17.E...

```
6
Enter the Marks of the Student
34
56
45
67
89
76
avg61
Batch 1: Eligible
Thank You
```

Result:

Thus, the console application in C# to implement Exception handling was executed successfully.

Ex No: 4 e)	MULTITHREADING
Date:	

Aim:

To create a console application in C# to implement the concept of Multithreading.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a namespace ConsoleApplication and class MyThread.

Step 7: Create a method reverse() and sum() and invoke the methods through ThreadStart class.

Step 8: To perform reverse of a digit get the last digit of a number by performing modulo operation. Multiply the result by 10 and add the remainder to the number. Divide the result by 10 and return the number.

Step 9: To perform sum of a digit get the remainder of the number by performing modulo operation and keep on adding it. Divide the number by 10 so that all the digits in the number gets added up.

Step 10: Create MyThread class and create the object for it. Start the thread using start() method.

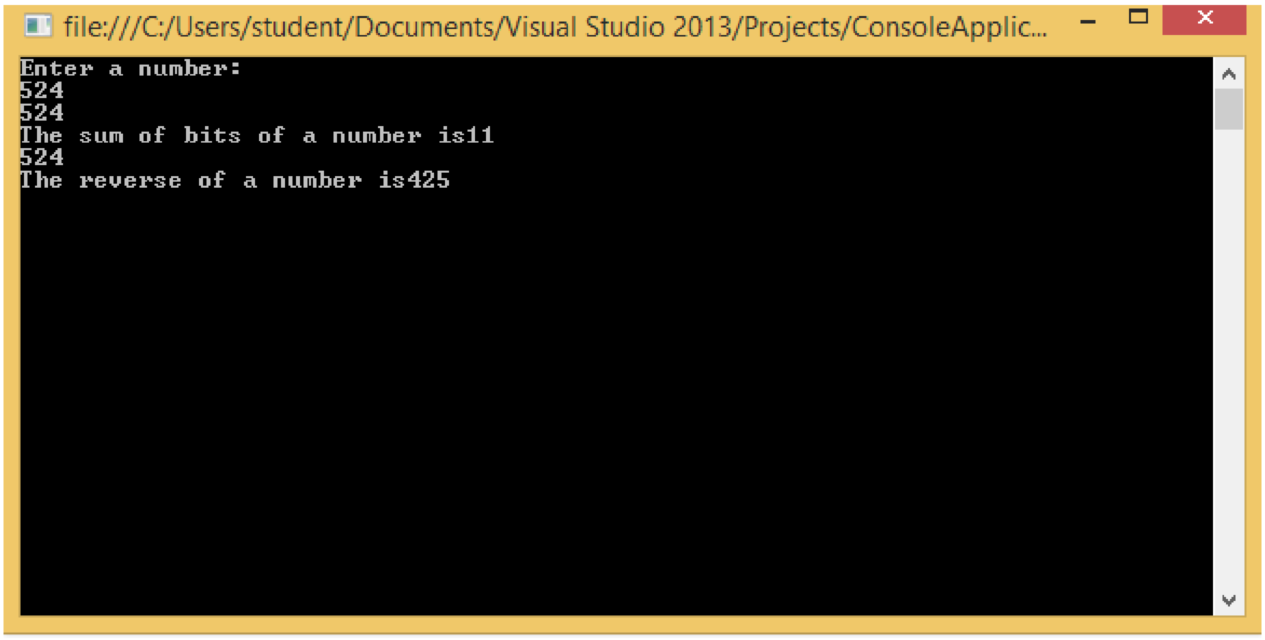
Step 11: Run the code & see the results by passing ctrl+F5. Choose debug->start windows debugging from top level menu bar.

Step 12: The output prints in the console window.

Program:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
namespace ConsoleApplication1 {
    public class MyThread
    {
        public void reverse()
        {
            int rev = 0, r;
           Console.WriteLine("Enter a number:");
           int n = int.Parse(Console.ReadLine());
           while(n>0)
            {
                r = n % 10;
                rev = rev * 10 + r;
                n = n / 10;
            }
            Console.WriteLine("The reverse of a number is" + rev);
            Console.ReadLine();
        }
        public void sum() {
            int sum = 0, r;
           int n = int.Parse(Console.ReadLine());
            while (n > 0)    {
                r = n % 10;
                sum += r;
                n = n / 10;
            }
            Console.WriteLine("The sum of bits of a number is" + sum);
            Console.ReadLine();
        }
    }
    class Program     {
        static void Main(string[] args)
        {
            MyThread mt=new MyThread();
            Thread t1 = new Thread(new ThreadStart(mt.reverse));
            Thread t2 = new Thread(new ThreadStart(mt.sum));
            t1.Start();
            t2.Start();
            //t2.Start();
            Console.ReadLine();
        }
        }
        }
```

Output:



file:///C:/Users/student/Documents/Visual Studio 2013/Projects/ConsoleApplic...

```
Enter a number:
524
524
The sum of bits of a number is11
524
The reverse of a number is425
```

Result:

Thus, the console application in C# to implement the concept of Multithreading was executed successfully.

Ex No: 5 a)	DELEGATES
Date:	

Aim:

To create a console application in C# to implement the concept of Delegates.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Create a class Delegate and define the methods AddNum() for addition, MultNum() formultiplication, SubNum() for subtraction, DivNum() for division.

Step 7: Create the method getNum() to input the values .

Step 8: Create an instance for the delegate and pass method as parameter.

Step 9: Invoke the method using the delegate object, in which the value is passed as parameter.

Step 10: Run the code & see the results by passing ctrl+F5. Choose debug->start windows debugging from top level menu bar.

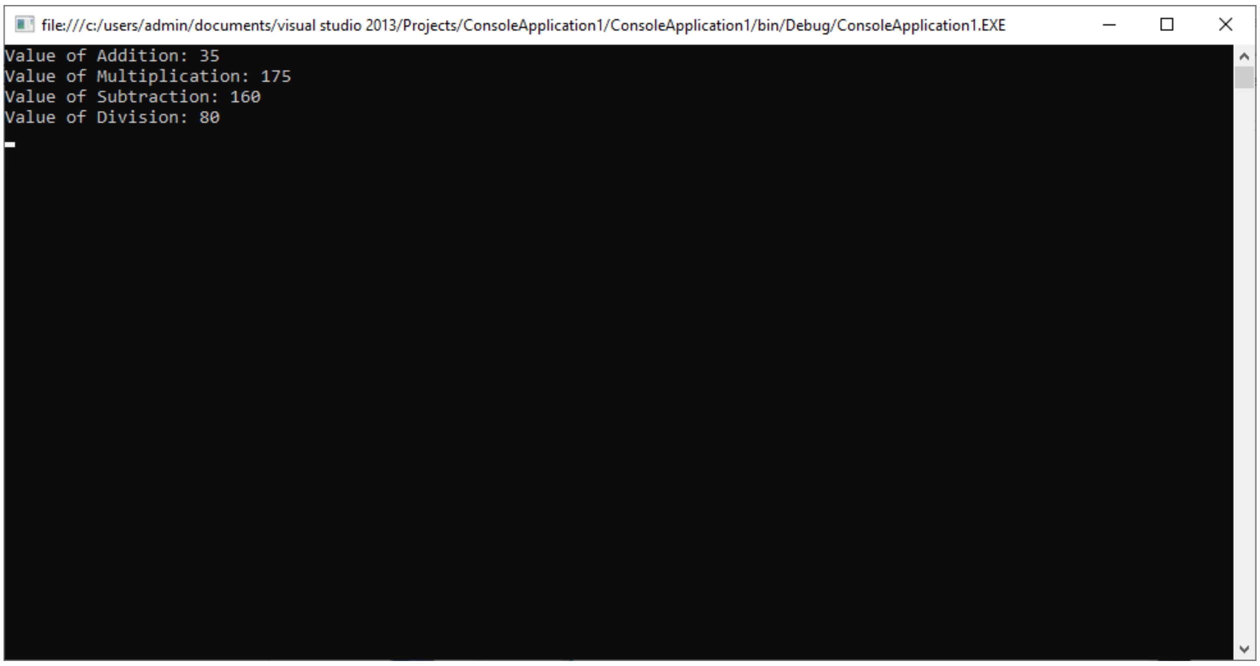
Step 11: The output prints in the console window.

Program:

```
using System;
using System.Collections.Generic;
using System.Linq;
using  System.Text;
using System.Threading;
delegate int NumberChanger(int n);
```

```
namespace DelegateAppl {
class TestDelegate {
static int num = 10;
public static int AddNum(int p) {
num += p;
return num;
}
public static int MultNum(int q) {
num *= q;
return num;
}
 public static int SubNum(int r)
{
    num -= r;
    return num;
}
public static int DivNum(int s)
{
        num/=s;
        return num;
    }
public static int getNum() {
            return num;
}
static void Main(string[] args) {
NumberChanger nc1 = new NumberChanger(AddNum);
NumberChanger nc2 = new NumberChanger(MultNum);
NumberChanger nc3 = new NumberChanger(SubNum);
NumberChanger nc4 = new NumberChanger(DivNum);
nc1(25);
Console.WriteLine("Value of Addition: {0}", getNum());
nc2(5);
Console.WriteLine("Value of Multiplication: {0}", getNum());
nc3(15);
Console.WriteLine("Value of Subtraction: {0}", getNum());
nc4(2);
Console.WriteLine("Value of Division: {0}", getNum());
Console.ReadKey();
}}}
```

Output:


file:///c:/users/admin/documents/visual studio 2013/Projects/ConsoleApplication1/ConsoleApplication1/bin/Debug/ConsoleApplication1.EXE

```
Value of Addition: 35
Value of Multiplication: 175
Value of Subtraction: 160
Value of Division: 80
```

Result:

Thus, the console application in C# to implement the concept of Delegates was executed successfully.

Ex No: 5 b)	**DELEGATES AND EVENT HANDLING**
Date:	

Aim:

To create a console application in C# to implement the concept of Events.

Algorithm:

Step 1: Start the Visual Studio 2022 framework.

Step 2: On the menu bar, choose File->New->Project.

Step 3: Choose Visual C# from templates and then choose console.

Step 4: Specify a name for your project and click OK button.

Step 5: This creates a new project in Solution Explorer.

Step 6: Declare a delegate SampleDelegate and an event SampleEvent.

Step 7: Create a Maths class as publisher, Operations class as subscriber and define AddOperation() and SubOperation() methods to call the methods Add() and Sub() to perform addition and subtraction operation.

Step 8: Inside Add() and Sub() methods, to raise an event call the event delegate to check subscription

Step 9: Subscribe to SampleEvent using += operator in Operations class and mention the name of handler SampleEventHandler to perform the required operations when an event is raised.

Step 10: The handler method SampleEventHandler in the Operations class must have samesignature of delegate SampleDelegate in Maths class.

Step 11: Run the code & see the results by passing ctrl+F5. Choose debug->start windows debugging from top level menu bar.

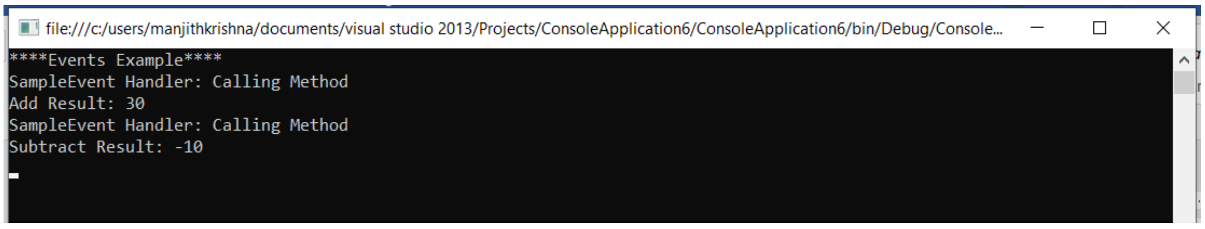
Step 12: The output prints in the console window.

Program:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication6
{
      class Maths
    {
        // Declare the delegate
        public delegate void SampleDelegate();
        //Declare an event
        public event SampleDelegate SampleEvent;
        public void Add(int a, int b)
        {
            // Calling event delegate to check subscription
            if (SampleEvent != null)
            {
                // Raise the event by using () operator
                SampleEvent();
                Console.WriteLine("Add Result: {0}", a + b);
            }
            else
            {
                Console.WriteLine("Not Subscribed to Event");
            }
        }
        public void Subtract(int x, int y)
        {
            // Calling event delegate to check subscription
            if (SampleEvent != null)
        {
            // Raise the event by using () operator
            SampleEvent();
            Console.WriteLine("Subtract Result: {0}", x - y);
        }
        else
        {
            Console.WriteLine("Not Subscribed to Event");
        }
    }
}
class Operations
{
    Maths m;
   public int a { get; set; }
   public int b { get; set; }
   public Operations(int x, int y)
    {
        m = new Maths();
        // Subscribe to SampleEvent event
        m.SampleEvent += SampleEventHandler;
        a = x;
        b = y;
    }
  // SampleEvent Handler
    public void SampleEventHandler()
    {
        Console.WriteLine("SampleEvent Handler: Calling Method");
    }
    public void AddOperation()
    {
        m.Add(a, b);
    }
    public void SubOperation()
```

```
        {
            m.Subtract(a, b);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("****Events Example****");
            Operations op = new Operations(10, 20);
            op.AddOperation();
            op.SubOperation();
            Console.ReadLine();
        }
    }}
```

Output:



file:///c:/users/manjithkrishna/documents/visual studio 2013/Projects/ConsoleApplication6/ConsoleApplication6/bin/Debug/Console...

```
****Events Example****
SampleEvent Handler: Calling Method
Add Result: 30
SampleEvent Handler: Calling Method
Subtract Result: -10
```

Result:

Thus, the console application in C# to implement the concept of Events was executed successfully.

Ex No: 6	**WINDOWS FORM APPLICATION USING UI CONTROLS IN .NET**
Date:	

Aim:

To create a login windows form application in C# using UI controls in .NET.

Algorithm:

Step 1: Open the Visual Studio then go to File -> New -> Project to create a new project and then select the language as Visual C# from the left menu.

Step 2: Click on Windows Forms App(.NET Framework) in the middle of current window. After that give the project name and click OK

Step 3: Add the controls to WinForms application by selecting Toolbox tab present in the extreme left side of Visual Studio where a list of controls will be displayed.

Step 4: Now drag and drop the textboxes for username and password, labels for username and password and buttons for login, exit and reset on created form for login windows application. By clicking on the particular dropped control, the properties of the control can be changed which is present in the right most corner of Visual Studio.

Step 5: Double click the button controls and add the code in the Form1.Designer.cs file present in the Solution Explorer Window.

Step 5.1: For exit button, use Close() method to close the windows form.

Step 5.2: For reset button, use Clear() method to clear the textboxes username and password.

Step 5.3: For login button, if the username and password matches then display ‘Welcome user’.

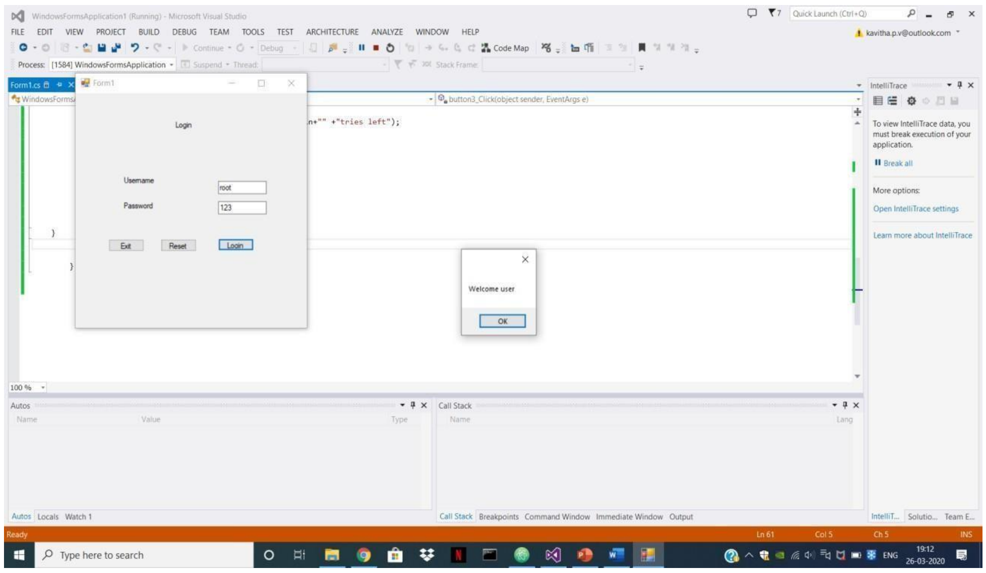
If it does not match, then it displays a message box containing ‘Wrong username and password’ with a maxcount of 3 tries left. When the maxcount becomes 3, then a message box is displayed with ‘Max try succceeded’.

Step 6: Run the program using F5 key or Play button present in the toolbar of Visual Studio. To stop the program use pause button present in the Tool Bar.

Program:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        Double count;
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            txtuser.Clear();
            txtpass.Clear();
            txtuser.Focus();
        }
        private void button3_Click(object sender, EventArgs e)
        {
            string user, pass;
            user = "root";
            pass = "123";
            if ((txtuser.Text == user) && (txtpass.Text == pass))
            {MessageBox.Show("Welcome user");}
            else
            {
            count = count + 1;
            double maxcount = 3;
             double remain;
            remain = maxcount - count;
            MessageBox.Show("Wrong username"+"\t" +remain+"" +"tries left");
            txtuser.Clear();
            txtpass.Clear();
            txtuser.Focus();
             if(count == maxcount)
            {MessageBox.Show("Max try succeeded");
           Application.Exit();}
    }}}
```

Output:



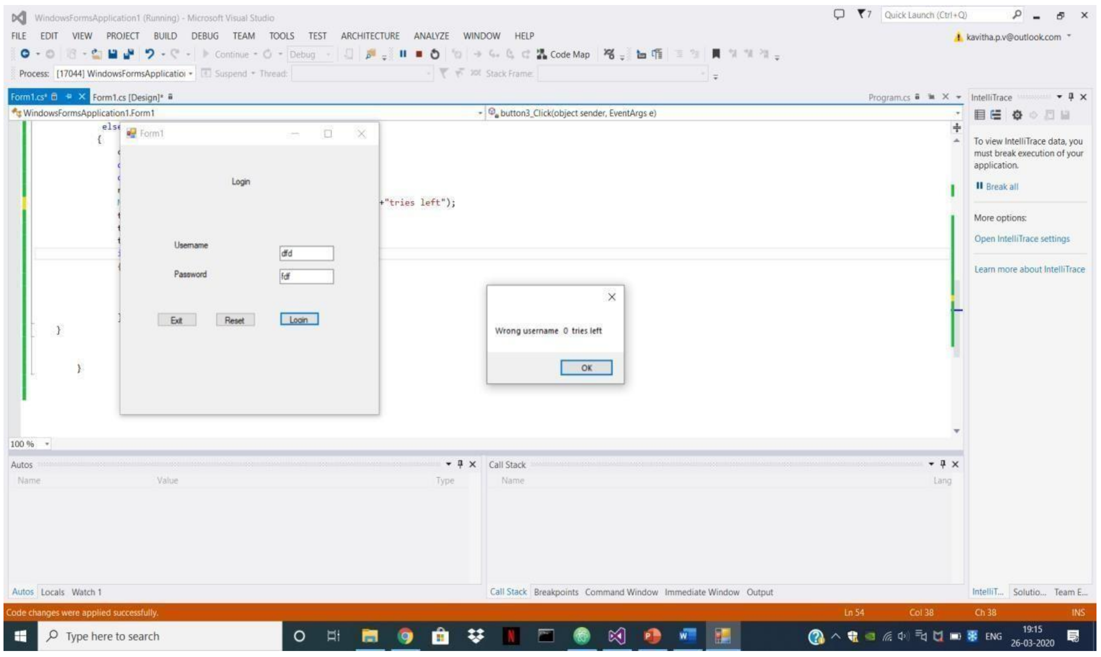
WindowsFormsApplication1 (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Form1
Login
Username
Password
Exit
Reset
Login
IntelliTrace
Autos
Call Stack
OK

Username: root

Password: 123

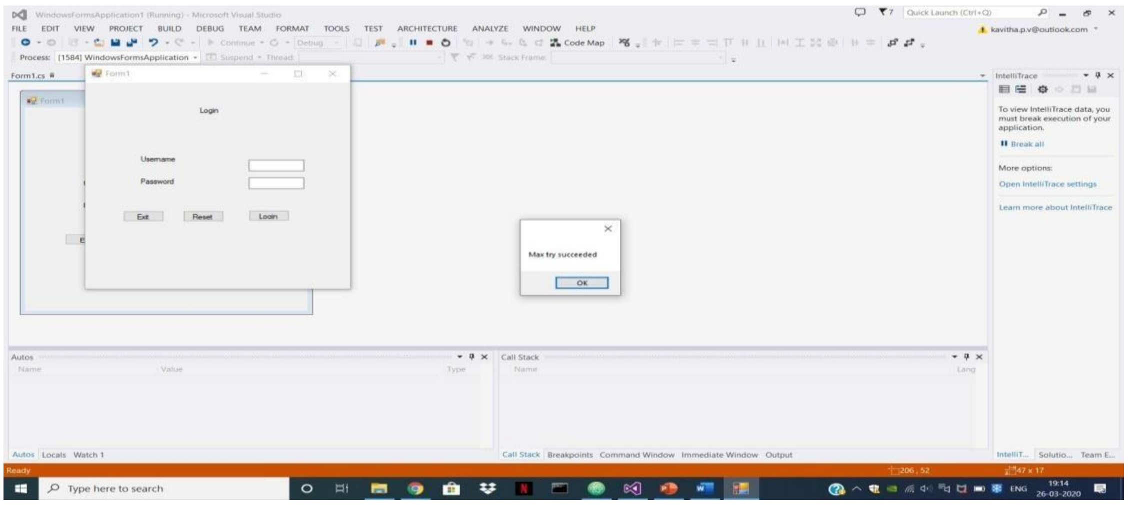
```
n+"" +"tries left");
```

Welcome user



```
+"tries left");
```

```
Wrong username 0 tries left
```



```
Max try succeeded
```

Result:

Thus, the creation of login windows application in C# using UI controls using .NET in Visual Studio framework was executed successfully.

Ex No: 7	WINDOWS FORM APPLICATION IN .NET USING ADO.NET
Date:	

Aim:

To create a windows form application in C# in .NET using ADO.Net connectivity for signup form in Visual Studio.

Algorithm:

Step 1: Open the Visual Studio then go to File -> New -> Project to create a new project and then select the language as Visual C# from the left menu.

Step 2: Click on Windows Forms App(.NET Framework) in the middle of current window. After that give the project name and click OK.

Step 3: Add the controls to WinForms application by selecting Toolbox tab present in the extreme left side of Visual Studio where a list of controls will be displayed.

Step 4: Now drag and drop the textboxes and labels for firstname, lastname, contact, username, password and confirm password, and button for submit on created form for signup windows application. By clicking on the particular dropped control, the properties of the control can be changed which is present in the right most corner of Visual Studio.

Step 5: Switch to Microsoft SQLServer Management Studio. Go to Database-> Right click -->New Database. Save the database name as UserregistrationDB and Click Ok.

Step 6: Click on newly database and now Right click on tables and add the columns as follows:

Column Name	Data Type
UserID	int
FirstName	varchar(50)
LastName	varchar(50)
Contact	varchar(50)
Address	varchar(250)
Username	varchar(50)
Password	varchar(50)

Step 7: Now Right click on UserID and then select Set Primary Key and set this same column as the identity specification for this table. To do this, go to Column Properties expand Identity Specification and set this as yes so that no values can be inserted into it. It can be automatically incremented by 1 upon new recording session. Save the table as ‘tblUser’.

Step 8: Right click on database UserregistrationDB and select New Query to create stored procedure. Type the following as stored procedure

```
CREATE PROC UserAdd
 @UserID  int,
  @FirstName varchar(50)
  @LastName varchar(50),
  @Contact varchar(50),
  @Address varchar(250),
  @UserName varchar(50),
  @Password varchar(50)
    AS
          INSERT INTO tblUser(FirstName,LastName,Contact,Address,UserName,Password)
         VALUES(@FirstName,@LastName,@Contact,@Address,@UserName,@Password)
```

Step 9: Click Execute button

Step 10: Double click on Submit button and add the code Form1.Designer.cs file present in the Solution Explorer Window.

Step 11: Now change the sored procedure as follows:

```
ALTER  PROC  UserAdd
@FirstName  varchar(50)
@LastName  varchar(50),
@Contact varchar(50),
 @Address varchar(250),
  @UserName varchar(50),
  @Password varchar(50)
    AS
            INSERT INTO tblUser(FirstName,LastName,Contact,Address,UserName,Password)
            VALUES
(@FirstName,@LastName,@Contact,@Address,@UserName,@Password)
```

and click Execute Button

Step 12: Give the connectionString in the form to connect database with application. Open the SQL Connection using SqlConnection.

Step 13: If the username and password is empty, then display a message box containing “Please fill mandatory fields” . If the password and confirm password does not match, then display a message box containing “ Password do not match”

Step 14: Use Open() method to open the database and create object for SqlCommand class to add the stored procedure and SqlConnection object as parameters.

Step15: Add the data entered by user to the database by the statement sqlCmd.Pararmeters.AddWithValue.

Step 16: Execute the query using ExecuteNonQuery(). Once the user enters the data, display a message box containing “Registration is successful”. Call the method Clear( ) to clear the textboxes.

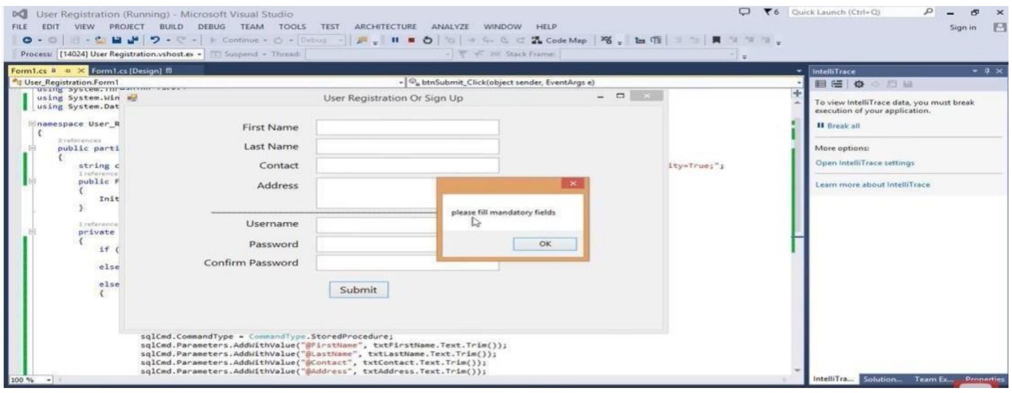
Step 17: Run the program using F5 key or Play button present in the toolbar of Visual Studio. To stop the program use pause button present in the Tool Bar.

Program:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace signup
{
   public partial class Form1 : Form
   {
      string connectionString = @"Data Source= IT-70\\SQLEXPRESS ; Initial
catalog=UserRegistrationDB; Integrated Security=True;";
      public Form1()
      {
         InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
         using (SqlConnection sqlCon = new SqlConnection(connectionString))
         {
            if (txtUsername.Text == "" || txtPassword.Text == "")
               MessageBox.Show("Please fill mandatory fields");
            else if (txtPassword.Text != txtConfirmpassword.Text)
               MessageBox.Show("Password do not match");
            else
            {
               sqlCon.Open();
               SqlCommand sqlCmd = new SqlCommand("UserAdd", sqlCon);
               sqlCmd.CommandType = CommandType.StoredProcedure;
               sqlCmd.Parameters.AddWithValue("@FirstName", txtFirstname.Text.Trim());
               sqlCmd.Parameters.AddWithValue("@LastName", txtLastname.Text.Trim());
               sqlCmd.Parameters.AddWithValue("@Contact", txtContact.Text.Trim());
               sqlCmd.Parameters.AddWithValue("@Username", txtUsername.Text.Trim());
```

```
                sqlCmd.Parameters.AddWithValue("@Password", txtPassword.Text.Trim());
                sqlCmd.ExecuteNonQuery();
                MessageBox.Show("Registration is successful");
                Clear();
            }
        }
    }
    void Clear()
    {
        txtFirstname.Text = txtLastname.Text = txtContact.Text = txtAddress.Text =
txtUsername.Text = txtPassword.Text = "";
    }
}}
```

Output:



User Registration (Running) - Microsoft Visual Studio

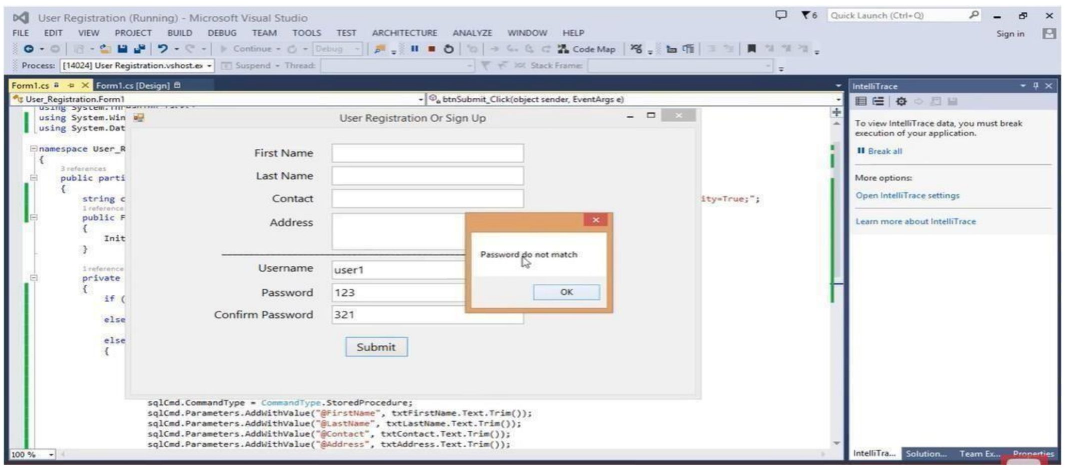
User Registration Or Sign Up

First Name	
Last Name	
Contact	
Address	
Username	
Password	
Confirm Password	

please fill mandatory fields

```
sqlCmd.CommandType = CommandType.StoredProcedure;
sqlCmd.Parameters.AddWithValue("@FirstName", txtFirstName.Text.Trim());
sqlCmd.Parameters.AddWithValue("@LastName", txtLastName.Text.Trim());
sqlCmd.Parameters.AddWithValue("@Contact", txtContact.Text.Trim());
sqlCmd.Parameters.AddWithValue("@Address", txtAddress.Text.Trim());
```

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Process: [14024] User Registration.vshost.exe
Form1.cs
Form1.cs [Design]
User_Registration.Form1
btnSubmit_Click(object sender, EventArgs e)
IntelliTrace
To view IntelliTrace data, you must break execution of your application.
Break all
More options:
Open IntelliTrace settings
Learn more about IntelliTrace
Submit
OK



User Registration (Running) - Microsoft Visual Studio

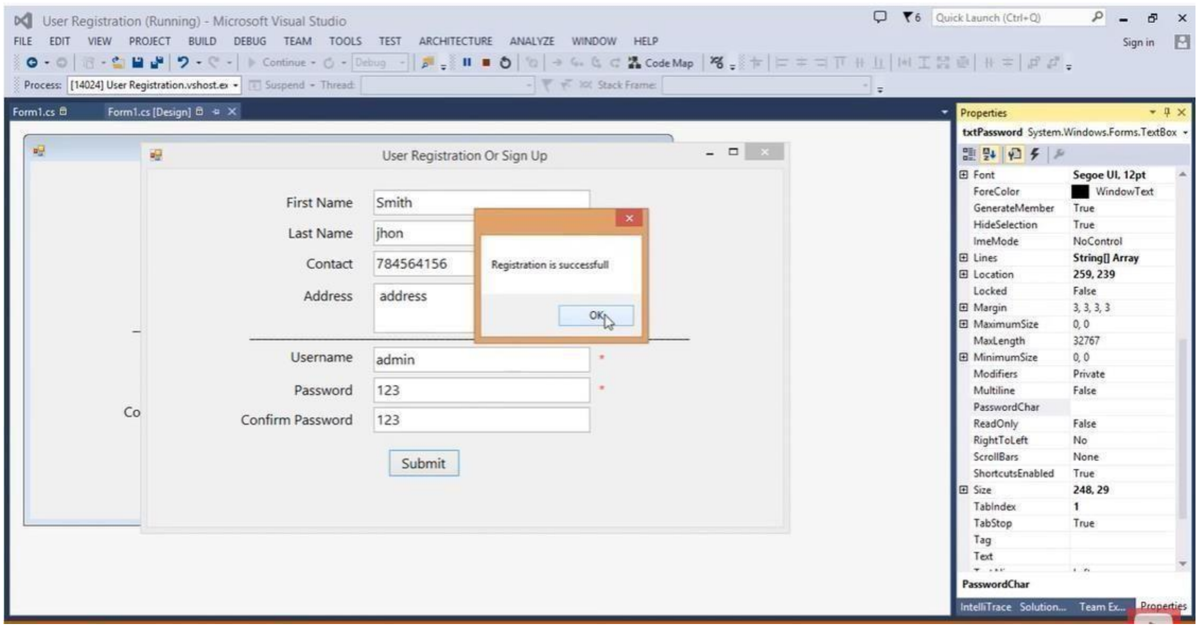
User Registration Or Sign Up

First Name	
Last Name	
Contact	
Address	
Username	user1
Password	123
Confirm Password	321

Password do not match

```
sqlCmd.CommandType = CommandType.StoredProcedure;
sqlCmd.Parameters.AddWithValue("@FirstName", txtFirstName.Text.Trim());
sqlCmd.Parameters.AddWithValue("@LastName", txtLastName.Text.Trim());
sqlCmd.Parameters.AddWithValue("@Contact", txtContact.Text.Trim());
sqlCmd.Parameters.AddWithValue("@Address", txtAddress.Text.Trim());
```

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Process: [14024] User Registration.vshost.exe
Form1.cs
Form1.cs [Design]
User_Registration.Form1
btnSubmit_Click(object sender, EventArgs e)
IntelliTrace
To view IntelliTrace data, you must break execution of your application.
Break all
More options:
Open IntelliTrace settings
Learn more about IntelliTrace
Submit
OK


User Registration (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue
Code Map
Process: [14024] User Registration.vshost.ex
Suspend
Thread:
Stack Frame:
Quick Launch (Ctrl+Q)
Sign in
Form1.cs
Form1.cs [Design]
IntelliTrace
Solution...
Team Ex...
Properties
Submit
OK

User Registration Or Sign Up

First Name	Smith
Last Name	jhon
Contact	784564156
Address	address
Username	admin
Password	123
Confirm Password	123

Registration is successfull

Properties

txtPassword System.Windows.Forms.TextBox

Font	Segoe UI, 12pt
ForeColor	WindowText
GenerateMember	True
HideSelection	True
ImeMode	NoControl
Lines	String[] Array
Location	259, 239
Locked	False
Margin	3, 3, 3, 3
MaximumSize	0, 0
MaxLength	32767
MinimumSize	0, 0
Modifiers	Private
Multiline	False
PasswordChar	
ReadOnly	False
RightToLeft	No
ScrollBars	None
ShortcutsEnabled	True
Size	248, 29
TabIndex	1
TabStop	True
Tag	
Text	

PasswordChar

Result:

Thus, the creation of windows form application in C# with ADO.Net connectivity for signup form in Visual Studio framework was executed successfully.

Ex No: 8	WEB FORM APPLICATION IN ASP.NET USING WEB CONTROLS
Date:	

Aim:

To develop a web form application in ASP.Net for login form using web controls in Visual Studio framework.

Algorithm:

Step 1: Open the Visual Studio then go to File -> New -> Project to create a new project and then select the language as Visual C# from the left menu.

Step 2: Click on ASP.NET Web Application in the middle of current window. After that give the project name and click OK.

Step 3: Select the Empty template and now the project appears. Right click on project in the solution explorer and click Add-> New Item->select Web Form and save the form name as 'Loginform.aspx'.

Step 4: Click the Design tab and drag and drop the UI controls like labels for username and password, text boxes for username and password, button for Submit, and another label for Incorrect Username and Password.

Step 5: By clicking on the particular dropped control, the properties of the control can be changed which is present in the right most corner of Visual Studio. Change the text and ID properties of UI controls. Set the visible property of third label to 'false'.

Step 6: Add another web form by Right click->Add->New Item-> select Web Form and save the form name as 'Welcomform.aspx'.

Step 7: Double click on Submit button and add the code in the Loginform .aspx.cs file present in the Solution Explorer Window.

Step 7.1: If the username entered by the user in the textbox matches with the password entered by the user, then display 'You are successfully logged in' in the new web form 'welcomform.aspx'.

Step 7.2: If the username entered by the user in the textbox does not matches with the password entered by the user, then a label is displayed with ‘Incorrect Username and Password’.

Step 8: Run the program using F5 key or Play button present in the toolbar of Visual Studio. To stop the program use pause button present in the Tool Bar.

Step 9: The output gets displayed in the web browser.

Program:

Loginform.aspx.cs

```
namespace WebApplication4
{
    public partial class LoginForm : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {}
        protected void butlogin_Click(object sender, EventArgs e)
        {
            if(txtUser.Text=="kavitha" && txtPassword.Text=="1234")
            {
                Response.Redirect("Welcomeform.aspx");
            }
            else
            {
                Label3.Visible = true;
            }
        }
    }
}
```

Welcomeform.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Welcomeform.aspx.cs"
Inherits="WebApplication4.Welcomeform" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
```

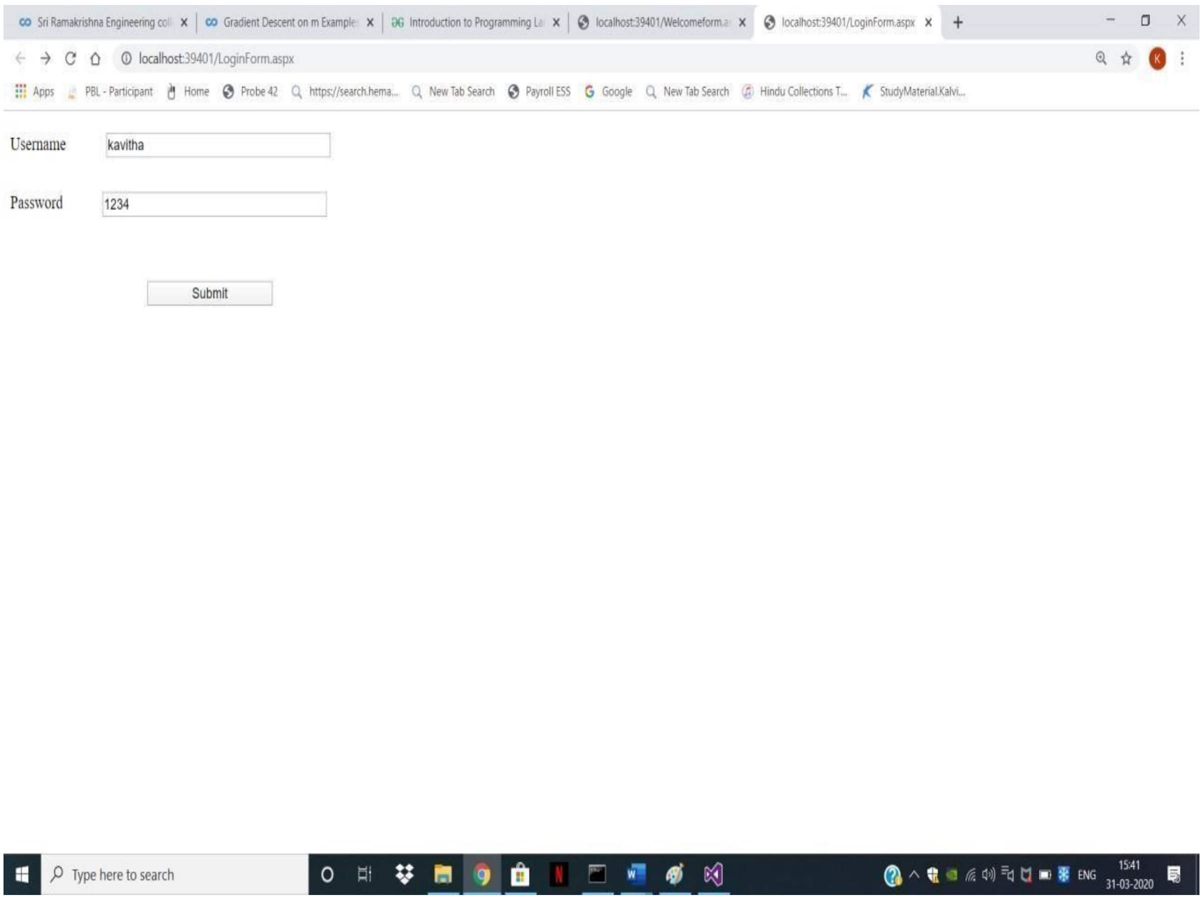
```
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
            <strong>You are successfully logged in!</strong></div>
    </form>
</body>
</html>
```

Loginform.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="LoginForm.aspx.cs"
Inherits="WebApplication4.LoginForm" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
            <asp:Label ID="Label1" runat="server" Text="Username"></asp:Label>
        <asp:TextBox ID="txtUser" runat="server" Height="16px" style="margin-left: 43px;
margin-top: 11px" Width="258px"></asp:TextBox>
        <br />
        <br />
        <asp:Label ID="Label2" runat="server" Text="Password"></asp:Label>
&nbsp;
        <asp:TextBox ID="txtPassword" runat="server" Height="16px" style="margin-left:
34px; margin-top: 11px" Width="258px"></asp:TextBox>
        <br />
        <br />
```

```
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
        <asp:Label ID="Label3" runat="server" style="font-weight: 700" Text="Incorrect
Username and Password" Visible="False"></asp:Label>
        <br />
        <br />
        <asp:Button ID="butlogin" runat="server" OnClick="butlogin_Click" style="margin-
left: 160px" Text="Submit" Width="146px" />
        <br />
         </div>
    </form>
</body>
</html>
```

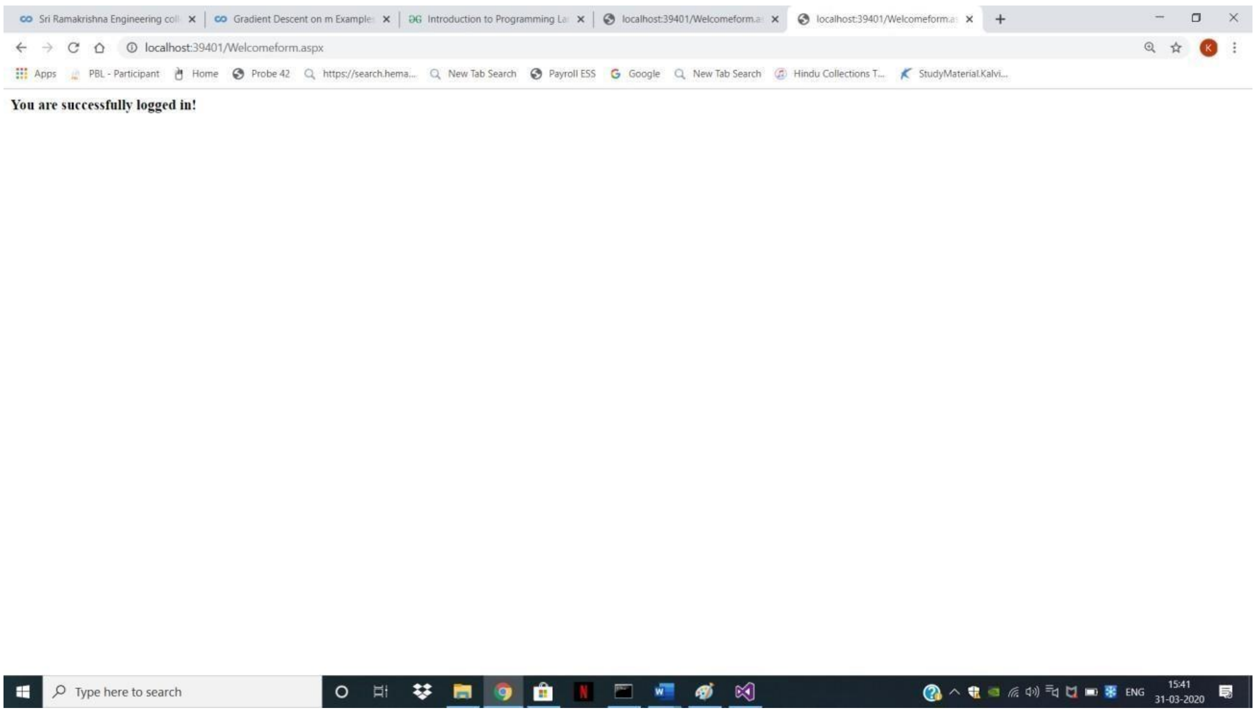
Output:



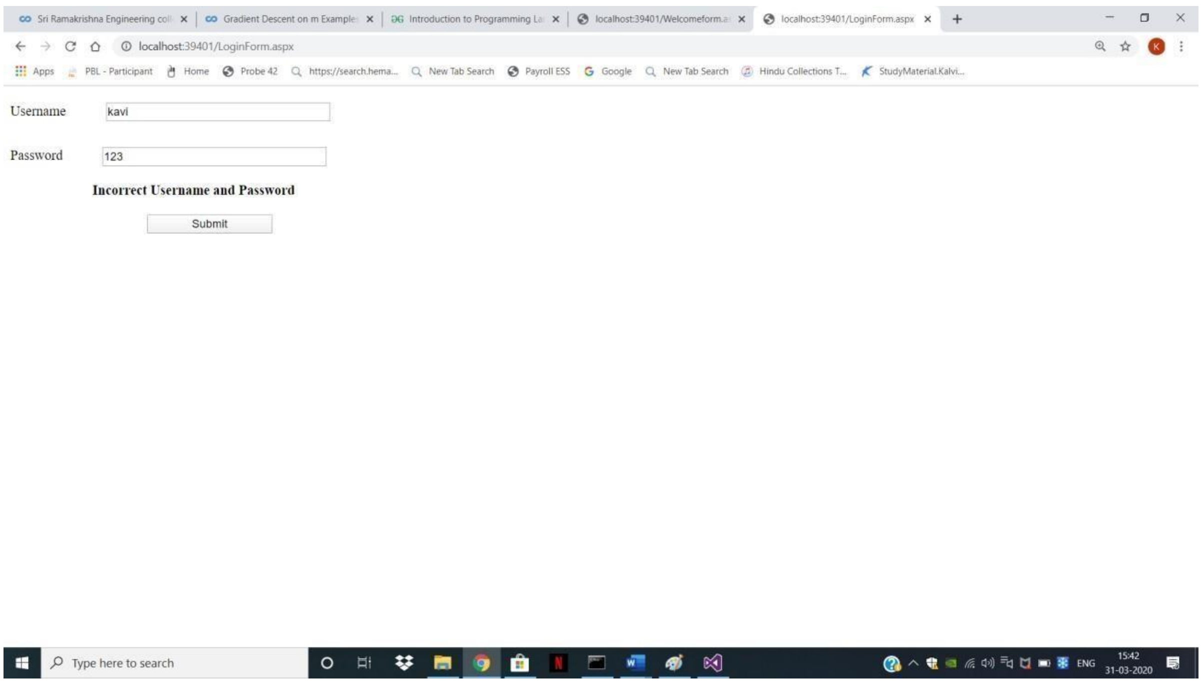
Username kavitha

Password 1234

Submit


localhost:39401/Welcomeform.aspx

You are successfully logged in!


localhost:39401/LoginForm.aspx

Username	kavi
Password	123

Incorrect Username and Password

Submit

Result:

Thus, the web form application in ASP.Net for login form using web controls in Visual Studio framework was developed and executed successfully.

Ex No: 9	**Web Application using Entity Framework**
Date:	

Aim:

To develop a web application using Entity Framework application in VisualStudio framework.

Algorithm:

Step 1: Open Visual Studio and select File → New → Project

Step 2: Select Installed → Templates → Visual C# → Windows from left pane and then in middle pane, select Console Application.

Step 3: Enter EFModelFirstDemo in the Name field

Step 4: To create model, first right-click on your console project in solution explorer and select Add → New Items..

Step 5: Dialog box will be opened, select ADO.NET Entity Data Model from middle pane and enter name ModelFirstDemoDB in the Name field.

Step 6: Click on Add button which will launch the Entity Data Model Wizard dialog.

Step 7: Select Empty EF Designer model and click Next button. The Entity Framework Designer opens with a blank model. Now we can start adding entities, properties and associations to the model.

Step 8: Right-click on the design surface and select Properties. In the Properties window, change the Entity Container Name to ModelFirstDemoDBContext.

Step 9: Right-click on the design surface and select Add New → Entity

Step 10: Enter Student as entity name and Student Id as property name and click Ok

Step 11: Right-click on the new entity on the design surface and select Add New → Scalar Property, enter Name as the name of the property.

Step 12: Enter FirstName and then add another two scalar properties such as LastName and EnrollmentDate

Step 13: Add two more Entities Course and Enrollment by following all the steps mentioned above and also add some Scalar properties as shown in the following step.

Step 14: For the three entities in Visual Designer, add some association or relationship between them.

Step 15: Right-click on the design surface and select Add New → Association

Step 16: Make one end of the relationship point to Student with a multiplicity of one and the other end point to Enrollment with a multiplicity of many.

Step 17: A Student has many Enrollments and Enrollment belongs to one Student. Ensure the Add foreign key properties to 'Post' Entity box is checked and click OK.

Step 18: Similarly, add one more association between Course and Enrollment.

To generate the database:

Step 1: Right-click on the design surface and select Generate Database from Model.

Step 2: Select existing database or create a new connection by clicking on New Connection**.**

Step 3: To create new Database, click on New Connection and enter Server name and database name and click next.

Step 4: Click Finish. This will add *.edmx.sql file in the project. Execute DDL scripts in Visual Studio by opening .sql file, then right-click and select Execute.

Step 5: Go to the server explorer, you will see that the database is created with three tables which are specified.

To swap model to generate code that makes use of the DbContext API:

Step 1: Right-click on an empty spot of your model in the EF Designer and select Add Code Generation Item.

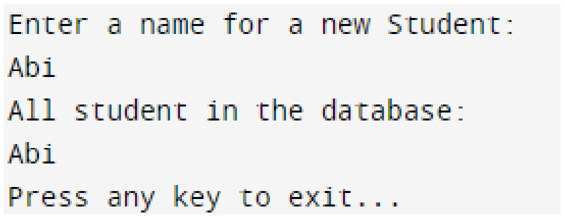
Step 2: Select EF 6.x DbContext Generator in middle pane and enter ModelFirstDemoModel in Name field.

Step 3: In the solution explorer that ModelFirstDemoModel.Context.tt and ModelFirstDemoModel.tt templates are generated.

Program:

```
using System;
using System.Linq;
namespace EFModelFirstDemo
{ class Program
{
    static void Main(string[] args) {
      using (var db = new ModelFirstDemoDBContext()) {
        // Create and save a new Student
        Console.Write("Enter a name for a new Student: ");
        var firstName = Console.ReadLine();

        var student = new Student {
          StudentID = 1,
          FirstName = firstName
        };
        db.Students.Add(student);
        db.SaveChanges();

        var query = from b in db.Students
          orderby b.FirstName select b;
        Console.WriteLine("All student in the database:");
        foreach (var item in query) {
          Console.WriteLine(item.FirstName);
        }
        Console.WriteLine("Press any key to exit...");
        Console.ReadKey();
      }
    }
  }
}
```

Output:



```
Enter a name for a new Student:
Abi
All student in the database:
Abi
Press any key to exit...
```

Result:

Thus, the web application using Entity Framework application in VisualStudio framework was developed and executed successfully.

Ex No: 10	.NET Core MVC Web Application
Date:	

Aim:

To develop a web application using .NET MVC application in VisualStudio framework.

Algorithm:

Step 1: Open the Visual Studio. Click File → New → Project menu option.

Step 2: Click on the left pane, select Templates → Visual C# → Web.

Step 3: Click on the middle pane, select ASP.NET Web Application.

Step 4: Enter the project name, MVCFirstApp, in the Name field and click ok to continue. You will see the following dialog which asks you to set the initial content for the ASP.NET project.

Step 5: Select the 'Empty' option and check the MVC checkbox in the Add folders and core references section. Click Ok, basic MVC project with minimal predefined content will be created.

To add Controller:

Step 6: Right-click on the controller folder in the solution explorer and select Add → Controller, the Add Scaffold dialog will be displayed.

Step 7: Select the MVC 5 Controller – Empty option and click 'Add' button. The Add Controller dialog will appear.

Step 8: Set the name to HomeController and click the Add button, a new C# file HomeController.cs in the Controllers folder will be opened.

Step 9: Modify the controller class by changing the action method called index and run this application and the result of the Index action method will be displayed on the browser.

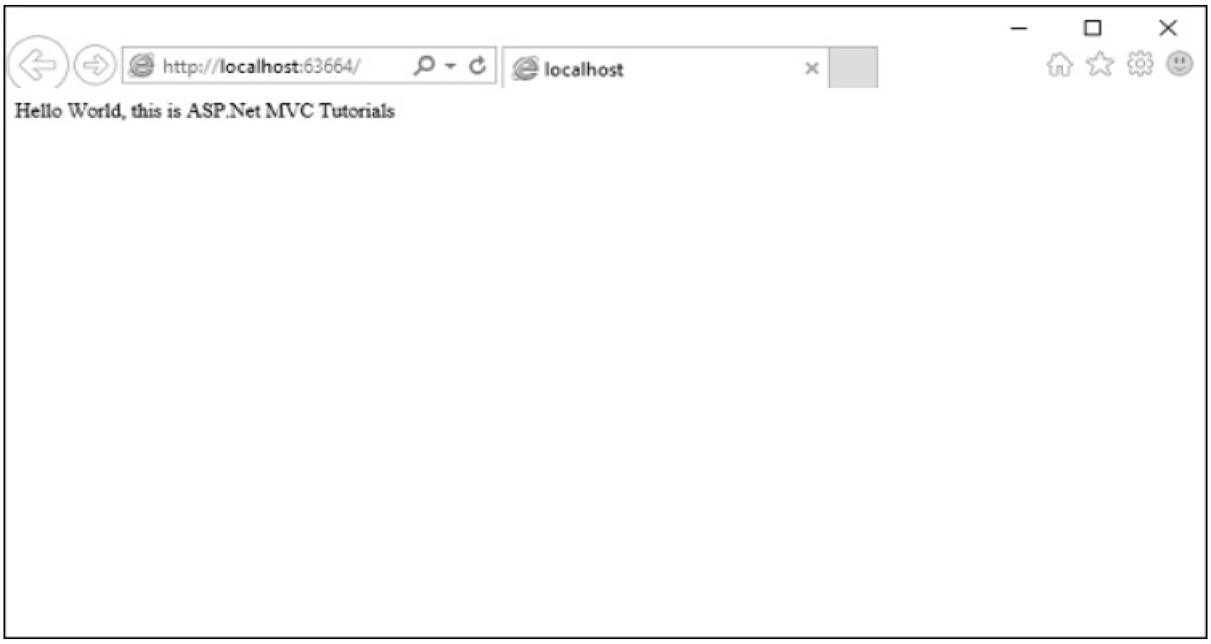
Step 10: The output gets displayed in the web browser.

Program:

Index.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace MVCFirstApp.Controllers {
  public class HomeController : Controller {
    // GET: Home
    public string Index(){
      return "Hello World, this is ASP.Net MVC ";
    }
  }
}
```

Output:



http://localhost:63664/

```
Hello World, this is ASP.Net MVC Tutorials
```

Result:

Thus, the web application using .NET MVC application in Visual Studio framework was developed and executed successfully.