RAMNIRANJAN JHUNJHUNWALA COLLEGE GHATKOPAR (W), MUMBAI - 400 086

DEPARTMENT OF INFORMATION TECHNOLOGY

2022 -2023

T.Y. B. Sc. (I.T) SEM V

RJ SUIT P503

ADVANCED WEB PROGRAMMING

Name : Aditya. A. Jaiswal

Roll No. : 3030

Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE

Ghatkopar (W), Mumbai-400 086

Certificate

NAAC Accredited "A" Grade
ESTD 1963
RJC
ISO 9001:2000
Best College
UNIVERSITY of MUMBAI

This is to certify that Mr. / Ms. **JAISWAL ADITYA AJAY MAMTA** Roll No **3030** of **Third Year** B.Sc. (I.T.) class has completed the required number of experiments in the subject of **Advanced Web Programming** in the Department of Information Technology during the academic year **2022-2023** .

Professor In- Charge

Co-ordinator of IT Department
Prof. Bharati Bhole
Prof. Archana Bhide

RAMNIRANJAN JHUNJHUNWALA COLLEGE
I.T. DEPT.
MUMBAI-400086
GHATKOPAR(W).

College Seal & Date

Examiner

INDEX

Practical No	Practical Title	Date
1	**Working with basic C# and ASP.NET**	
a	Creating an application that obtains four different values from the user and displays the products.	20-06-2022
b	Create an application to demonstrate the string operations.	20-06-2022
c	Create an application that receives the (Student Id, Student Name, Course Name, Date of Birth) information from a set of students. The application should also display the information of all the students once the data entered.	24-06-2022
2	**Working with Object Oriented C# and ASP.NET**	
a	Create simple application to perform following operations: i. Finding factorial Value ii. Money Conversion iii. Quadratic Equation iv. Temperature Conversion	27-06-2022
b	Create simple application to demonstrate use of following concepts i. Function Overloading ii. Inheritance (all types) iii. Constructor overloading iv. Interfaces	01-07-2022
c	Create simple application to demonstrate use of following concepts i. Using Delegates and events ii. Exception handling	04-07-2022
3	**Working with Web Forms and Controls**	
a	Create a simple web page with various server controls to demonstrate setting and use of their properties.(Example: AutoPostBack)	11-07-2022
b	Demonstrate the use of Calendar control to perform following operations. a. Display Messages In A Calendar Control b. Display vacation in a calendar control c. Selected day in a calendar control using style d. Difference between two calendar control	11-07-2022
c	Demonstrate the use of Treeview control performs following operations. a. Treeview control and datalist. b. Treeview Operations	15-07-2022
4	**Working with Form Controls**	
a	Create a Registration form to demonstrate use of various Validation controls.	18-07-2022
b	Create a Web Form to demonstrate use of Adrotator Control.	18-07-2022
c	Create a Web Form to demonstrate use of User Controls.	18-07-2022
5	**Working with Navigation, Beautification and Master Page.**	

a	Create Web Form to demonstrate use of Website Navigation controls and Site Map. i. Menu Control. ii. Site Map Control	25-07-2022
b	Create a web application to demonstrate use of Master page with applying Styles and Themes for page beautification.	25-07-2022
c	Create a web application to demonstrate various states of ASP.NET Pages.	25-07-2022
6	**Working with Database**	
a	Create a web application bind data in a multiline textbox by querying in another textbox.	08-08-2022
b	Demonstrate the use of Data list link control.	08-08-2022
7	**Working with Database**	
a	Create a web application to display Data Binding using dropdown list control.	22-08-2022
b	Create a web application to display the phone no of an author using a database.	22-08-2022
c	Create a web application for inserting and deleting records from a database. (Using Execute-Non Query).	22-08-2022
8	**Working with Data Controls**	
a	Create a web application to demonstrate data binding using DetailsView and FormView Control.	29-08-2022
b	Create a web application to display Using Disconnected Data Access and Data binding using GridView.	29-08-2022
9	**Working with GridView control**	
a	Create a web application to demonstrate use of GridView button column and GridView events.	21-09-2022
b	Create a web application to demonstrate GridView paging and Create own table format using GridView.	21-09-2022
c	Create a web application to demonstrate use of GridView control template and GridView hyperlink.	21-09-2022
10	**Working with AJAX and XML**	
a	Create a web application to demonstrate reading and writing operations with XML.	12-09-2022
b	Create a web application to demonstrate use of various Ajax controls. i. Update panel ii. Timer Control iii. Update Progress Control	17-09-2022
11	**Program to Create and Use dll**	**15-09-2022**

AWP Journal

Practical 1: Working with basic C# and ASP .NET

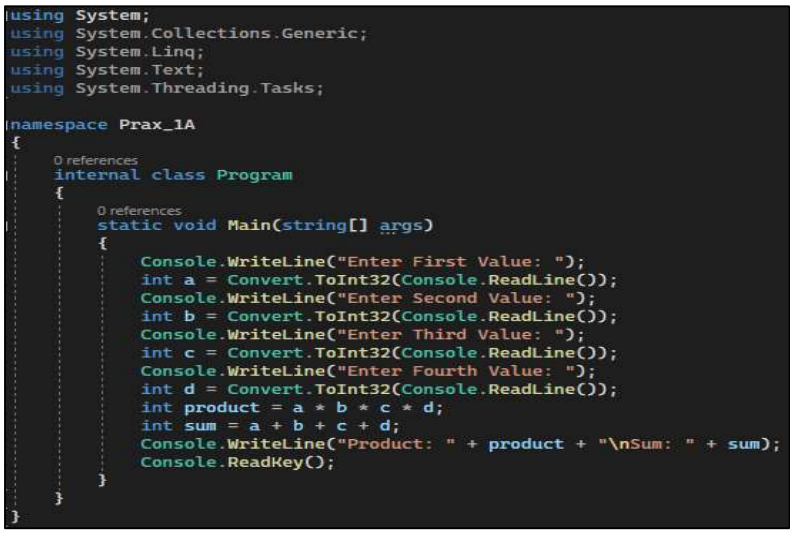
a) Create an application that obtains four int values from the user and displays the product.

A console application, in the context of C#, is an application that takes input and displays output at a command line console with access to three basic data streams: standard input, standard output and standard error.

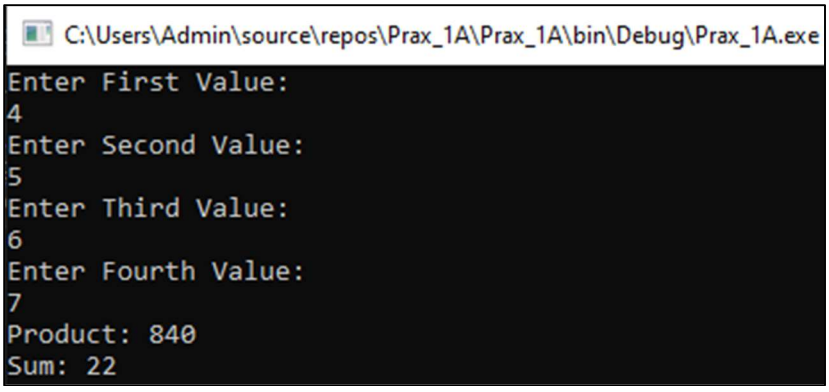
Here we are taking Four integers values as input from the user and displaying the product of all integers as output on the console .

Source code:

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

namespace Prax_1A
{
    internal class Program
    {
        static void Main(string[] args)
        {
                    Console.WriteLine("Enter First Value: ");
                    int a = Convert.ToInt32(Console.ReadLine());
                    Console.WriteLine("Enter Second Value: ");
                    int b = Convert.ToInt32(Console.ReadLine());
                    Console.WriteLine("Enter Third Value: ");
                    int c = Convert.ToInt32(Console.ReadLine());
                    Console.WriteLine("Enter Fourth Value: ");
                    int d = Convert.ToInt32(Console.ReadLine());
                    int product = a * b * c * d;
                    int sum = a + b + c + d;
                    Console.WriteLine("Product: " + product + "\nSum: " + sum);
                    Console.ReadKey();
         }
    }
}
```



Output:

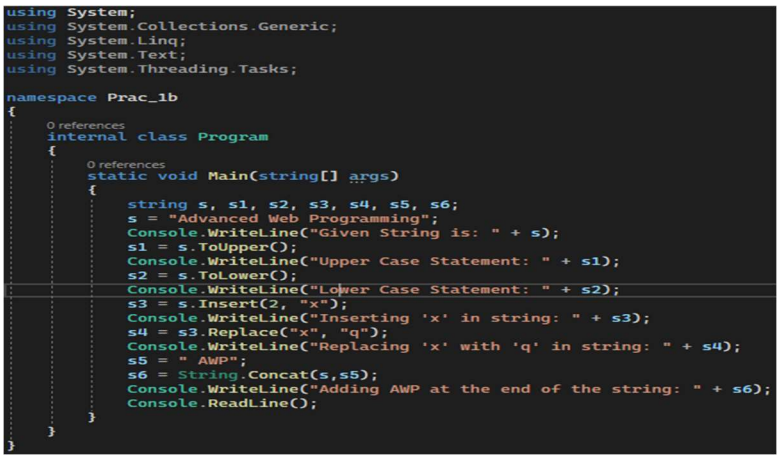


b) Create an application to demonstrate string operations.

In C#string is an object of System.String class that represents a sequence of characters. We can perform many operations on strings such as concatenation, comparison, getting substring, search, trim, replacement etc.

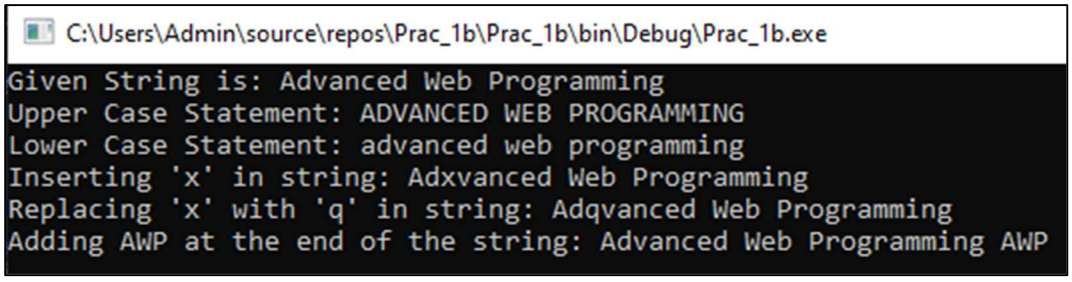
In C#string is a keyword which is an alias for System.String class. That is why string and String are equivalent. We are free to use any naming convention.

Source code:



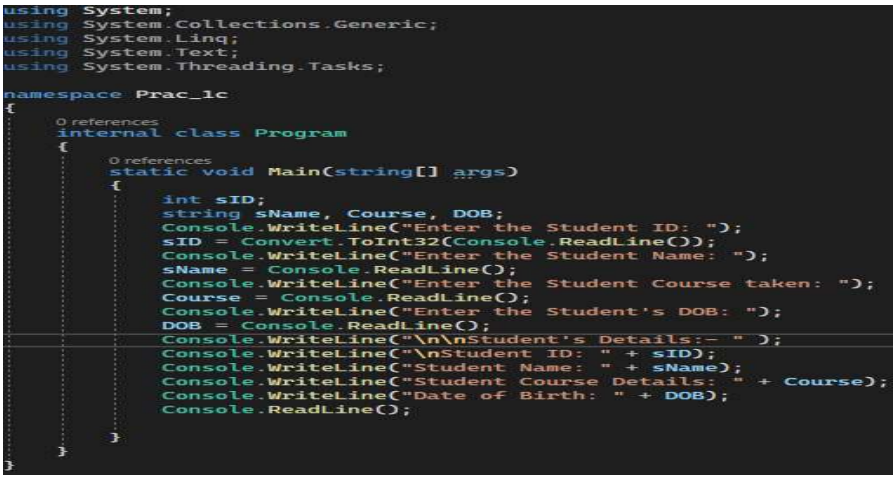
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Prac_1b
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string s, s1, s2, s3, s4, s5, s6;
            s = "Advanced Web Programming";
            Console.WriteLine("Given String is: " + s);
            s1 = s.ToUpper();
            Console.WriteLine("Upper Case Statement: " + s1);
            s2 = s.ToLower();
            Console.WriteLine("Lower Case Statement: " + s2);
            s3 = s.Insert(2, "x");
            Console.WriteLine("Inserting 'x' in string: " + s3);
            s4 = s3.Replace("x", "q");
            Console.WriteLine("Replacing 'x' with 'q' in string: " + s4);
            s5 = " AWP";
            s6 = String.Concat(s,s5);
            Console.WriteLine("Adding AWP at the end of the string: " + s6);
            Console.ReadLine();
        }
    }
}
```

Output:



```
C:\Users\Admin\source\repos\Prac_1b\Prac_1b\bin\Debug\Prac_1b.exe
Given String is: Advanced Web Programming
Upper Case Statement: ADVANCED WEB PROGRAMMING
Lower Case Statement: advanced web programming
Inserting 'x' in string: Adxvanced Web Programming
Replacing 'x' with 'q' in string: Adqvanced Web Programming
Adding AWP at the end of the string: Advanced Web Programming AWP
```

c) Create an application that receives the (Student Id, Student Name, Course Name, Date of Birth) information from a set of students. The application should also display the information of all the students once the data entered.



Source code:

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

namespace Prac_1c
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int sID;
            string sName, Course, DOB;
            Console.WriteLine("Enter the Student ID: ");
            sID = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter the Student Name: ");
            sName = Console.ReadLine();
            Console.WriteLine("Enter the Student Course taken: ");
            Course = Console.ReadLine();
            Console.WriteLine("Enter the Student's DOB: ");
            DOB = Console.ReadLine();
            Console.WriteLine("\n\nStudent's Details:- " );
            Console.WriteLine("\nStudent ID: " + sID);
            Console.WriteLine("Student Name: " + sName);
            Console.WriteLine("Student Course Details: " + Course);
            Console.WriteLine("Date of Birth: " + DOB);
            Console.ReadLine();

        }
    }
}
```

Output:

```
C:\Users\Admin\source\repos\Prac_1c\Prac_1c\bin\Debug\Prac_1c.exe
Enter the Student ID:
102
Enter the Student Name:
Aditya Jaiswal
Enter the Student Course taken:
BSc.IT
Enter the Student's DOB:
07-11-2002


Student's Details:-

Student ID: 102
Student Name: Aditya Jaiswal
Student Course Details: BSc.IT
Date of Birth: 07-11-2002
```

Practical 2: Working with Object Oriented C# and ASP .NET

OOP stands for "**Object Oriented Programming**".

Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

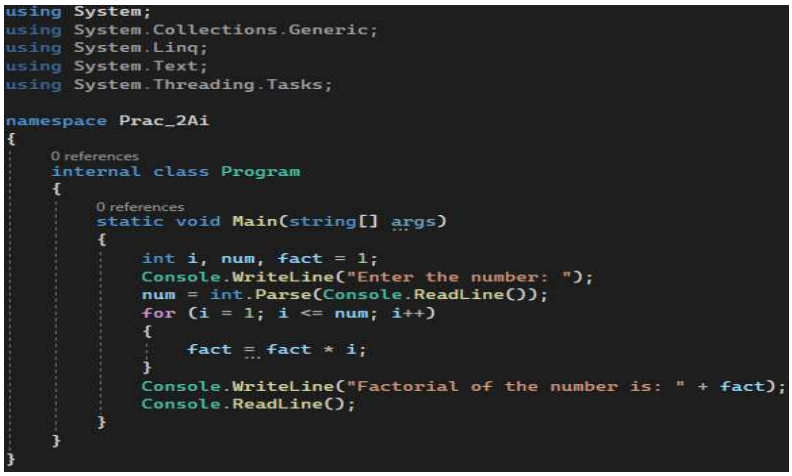
Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute. OOP provides a clear structure for the programs.
- OOP helps to keep the C# code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug. OOP makes it possible to create full reusable applications with less code and shorter development time.

a) Create simple application to perform following operations:

i) Factorial Numbers:

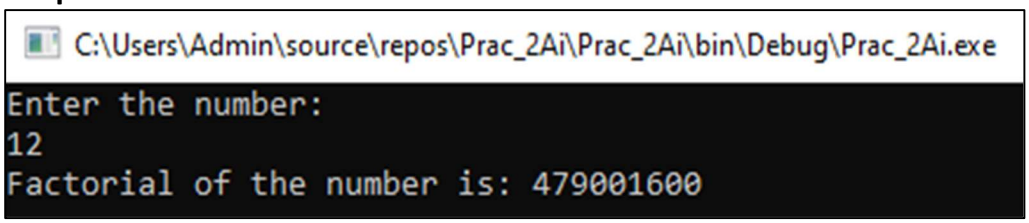
Factorial, in mathematics, is the product of all positive integers less than or equal to a given positive integer and denoted by that integer and an exclamation point. Thus, factorial seven is written 7!, meaning 1 × 2 × 3 × 4 × 5 × 6 × 7. Factorial zero is defined as equal to 1.

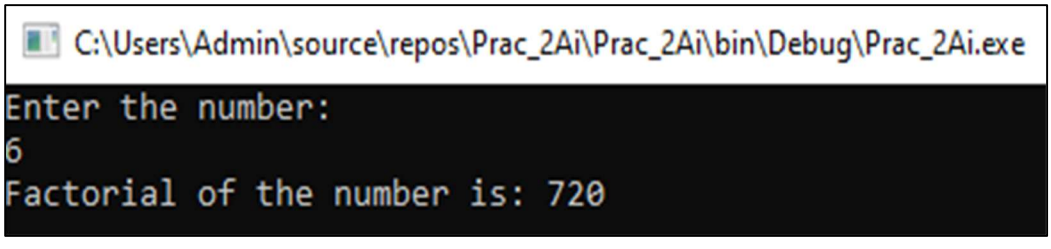


Source code:

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

namespace Prac_2Ai
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int i, num, fact = 1;
            Console.WriteLine("Enter the number: ");
            num = int.Parse(Console.ReadLine());
            for (i = 1; i <= num; i++)
            {
                fact = fact * i;
            }
            Console.WriteLine("Factorial of the number is: " + fact);
            Console.ReadLine();
        }
    }
}
```

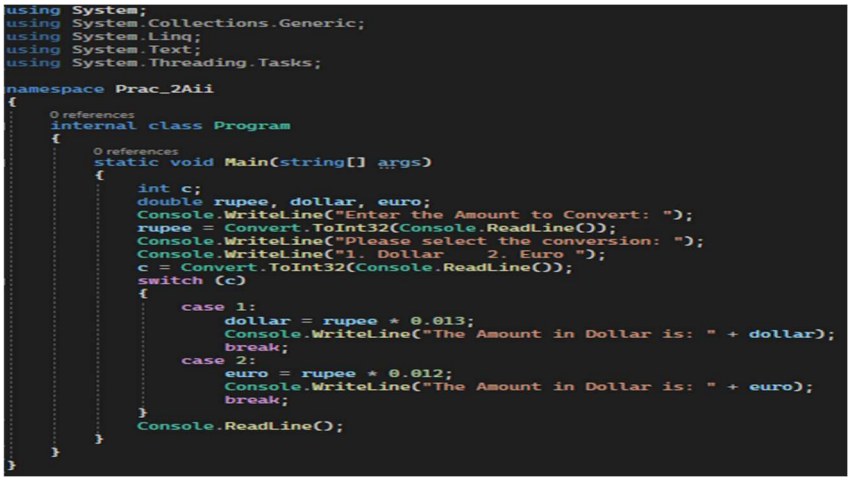
Output:





ii) Money Conversion:

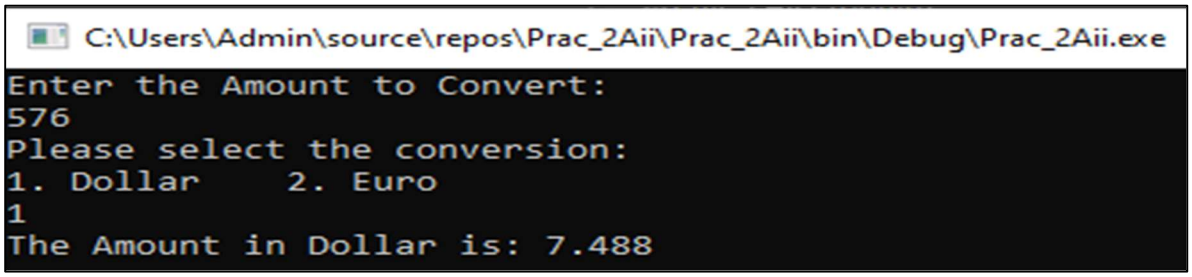
A currency converter is software code that is designed to convert one currency into another in order to check its corresponding value. The code is generally a part of a web site or it forms a mobile app and it is based on current market or bank exchange rates.

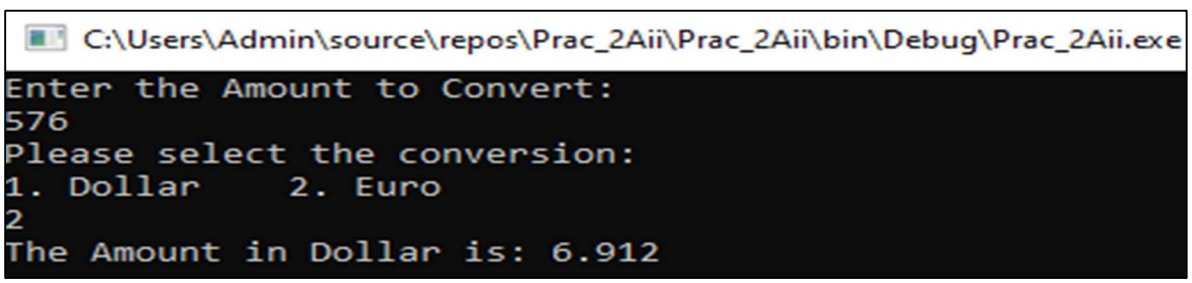


Source code:

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

namespace Prac_2Aii
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int c;
            double rupee, dollar, euro;
            Console.WriteLine("Enter the Amount to Convert: ");
            rupee = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Please select the conversion: ");
            Console.WriteLine("1. Dollar    2. Euro ");
            c = Convert.ToInt32(Console.ReadLine());
            switch (c)
            {
                case 1:
                    dollar = rupee * 0.013;
                    Console.WriteLine("The Amount in Dollar is: " + dollar);
                    break;
                case 2:
                    euro = rupee * 0.012;
                    Console.WriteLine("The Amount in Dollar is: " + euro);
                    break;
            }
            Console.ReadLine();
        }
    }
}
```

Output:





iii) Quadratic Equation:

In algebra, a quadratic equation (from the Latin quadratus for "square") is any equation that can be rearranged in standard form as where x represents an unknown, and a, b, and c represent known numbers, where a ≠ 0. If a = 0, then the equation is linear, not quadratic, as there is no term. The numbers a, b, and c are the coefficients of the equation and may be distinguished by calling them, respectively, the quadratic coefficient, the linear coefficient and the constant or free term.

Source code:

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

namespace Prac_2Aiii
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double a, b, c;
            double deter, x1, x2;
            Console.WriteLine("Enter the value of A: ");
            a = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Enter the value of B: ");
            b = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Enter the value of : ");
            c = Convert.ToDouble(Console.ReadLine());
            if (a == 0)
            {
                x1 = -c / b;
                Console.WriteLine("The roots are Linear: ", x1);
            }
            else
            {
                deter = (b * b) - (4 * a * c);
                if (deter > 0)
                {
                    Console.WriteLine("The Roots are Real And Distinct Roots.");
                    x1 = (-b / (2 * a)) + (Math.Sqrt(deter) / (2 * a));
                    x2 = (-b / (2 * a)) - (Math.Sqrt(deter) / (2 * a));
                    Console.WriteLine("The Roots are : " + x1 + " and " + x2);
                }
                else if (deter == 0)
                {
                    Console.WriteLine("The Roots are Repeated.");
                    x1 = -b / (2 * a);
                    Console.WriteLine("The Roots is: " + x1);
                }
                else
                {
                    Console.WriteLine("The Roots are Imaginary Roots.");
                    x1 = -b / (2 * a);
                    x2 = ((Math.Sqrt((4 * a * c) - (b * b))) / (2 * a));
                    Console.WriteLine("The Root 1: " + x1 + "+i" + x2);
                    Console.WriteLine("The Root 2: " + x1 + "-i" + x2);
                }
            }
```

```
            Console.ReadLine();
        }
    }
}
```

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

namespace Prac_2Aiii
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            double a, b, c;
            double deter, x1, x2;
            Console.WriteLine("Enter the value of A: ");
            a = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Enter the value of B: ");
            b = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Enter the value of : ");
            c = Convert.ToDouble(Console.ReadLine());
            if (a == 0)
            {
                x1 = -c / b;
                Console.WriteLine("The roots are Linear: ", x1);
            }
            else
            {
                deter = (b * b) - (4 * a * c);
                if (deter > 0)
                {
                    Console.WriteLine("The Roots are Real And Distinct Roots.");
                    x1 = (-b / (2 * a)) + (Math.Sqrt(deter) / (2 * a));
                    x2 = (-b / (2 * a)) - (Math.Sqrt(deter) / (2 * a));
                    Console.WriteLine("The Roots are : " + x1 + " and " + x2);
                }
                else if (deter == 0)
                {
                    Console.WriteLine("The Roots are Repeated.");
                    x1 = -b / (2 * a);
                    Console.WriteLine("The Roots is: " + x1);
                }
                else
                {
                    Console.WriteLine("The Roots are Imaginary Roots.");
                    x1 = -b / (2 * a);
                    x2 = ((Math.Sqrt((4 * a * c) - (b * b))) / (2 * a));
                    Console.WriteLine("The Root 1: " + x1 + "+i" + x2);
                    Console.WriteLine("The Root 2: " + x1 + "-i" + x2);
                }
            }
            Console.ReadLine();
        }
    }
}
```

Output:

```
C:\Users\Admin\source\repos\Prac_2Aiii\Prac_2Aiii\b
Enter the value of A:
2
Enter the value of B:
-8
Enter the value of :
-24
The Roots are Real And Distinct Roots.
The Roots are : 6 and -2
```

```
C:\Users\Admin\source\repos\Prac_2Aiii\
Enter the value of A:
1
Enter the value of B:
4
Enter the value of :
5
The Roots are Imaginary Roots.
The Root 1: -2+i1
The Root 2: -2-i1
```

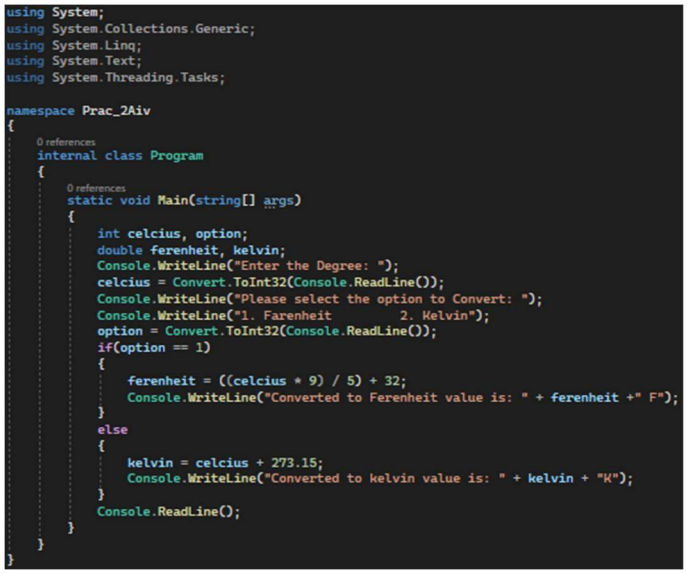
```
C:\Users\Admin\source\repos
Enter the value of A:
4
Enter the value of B:
-4
Enter the value of :
1
The Roots are Repeated.
The Roots is: 0.5
```

iv) Temperature Conversion:

There are many different units of measuring temperature. The three most important ones are Fahrenheit (F), Kelvin (K), and Celsius (C).

Looking to convert temperatures? Our newly launched temperature unit conversion calculator will do all of the maths for you!

The basic formula for converting Fahrenheit into Celsius is
(°F − 32) × 5/9 = °C.
To convert Fahrenheit degrees into Kelvins,
(°F − 32) × 5/9 + 273.15 = K.
To convert Kelvins into Celsius degrees, the formula is
K − 273.15 = °C,
while the formula of converting Kelvins into Fahrenheit degrees is
(K − 273.15) × 9/5 + 32= °F.



Source code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Prac_2Aiv
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int celcius, option;
            double ferenheit, kelvin;
            Console.WriteLine("Enter the Degree: ");
            celcius = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Please select the option to Convert: ");
            Console.WriteLine("1. Farenheit         2. Kelvin");
            option = Convert.ToInt32(Console.ReadLine());
            if(option == 1)
            {
                ferenheit = ((celcius * 9) / 5) + 32;
                Console.WriteLine("Converted to Ferenheit value is: " + ferenheit +" F");
            }
            else
            {
                kelvin = celcius + 273.15;
                Console.WriteLine("Converted to kelvin value is: " + kelvin + "K");
            }
            Console.ReadLine();
        }
    }
}
```

Output:

```
C:\Users\Admin\source\repos\Prac_2Aiv\Prac_2Aiv\bin\Debug\Prac_2Aiv.exe
Enter the Degree:
50
Please select the option to Convert:
1. Farenheit         2. Kelvin
1
Converted to Ferenheit value is: 122 F
```

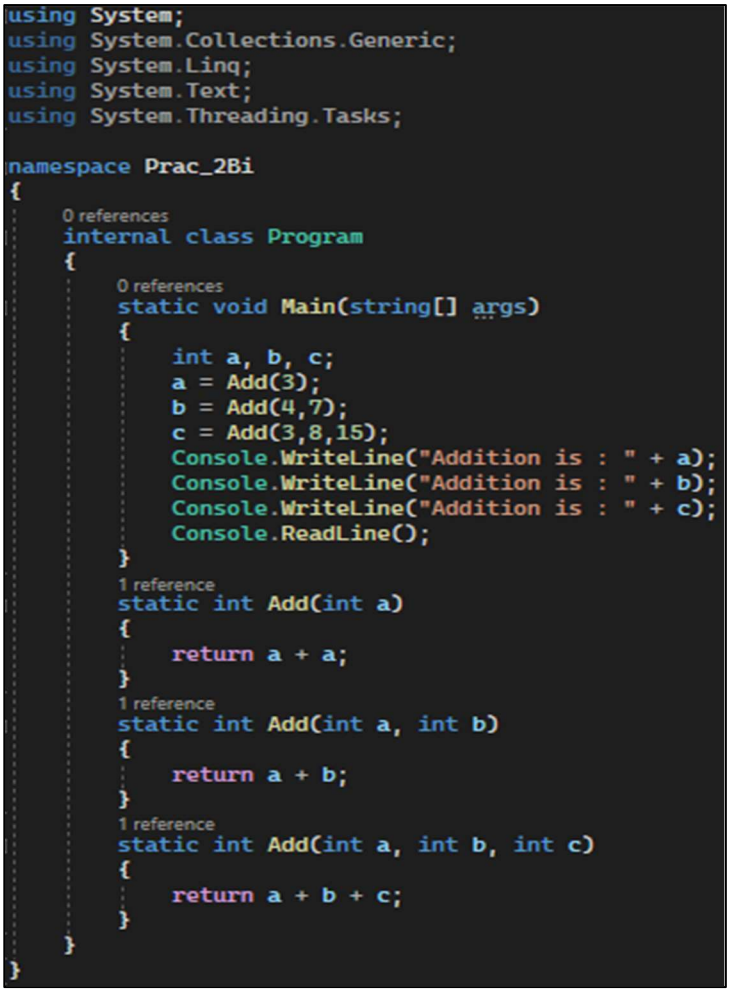
```
C:\Users\Admin\source\repos\Prac_2Aiv\Prac_2Aiv\bin\Debug\Prac_2Aiv.exe
Enter the Degree:
50
Please select the option to Convert:
1. Farenheit         2. Kelvin
2
Converted to kelvin value is: 323.15K
```

b) Create simple application to demonstrate use of following concepts:

i) Function Overloading:

Method Overloading is the common way of implementing polymorphism. It is the ability to redefine a function in more than one form. A user can implement function overloading by defining two or more functions in a class sharing the same name. C# can distinguish the methods with different method signatures.

i.e. the methods can have the same name but with different parameters list (i.e. the number of the parameters, order of the parameters, and data types of the parameters) within the same class.



Source code:

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

namespace Prac_2Bi
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int a, b, c;
            a = Add(3);
            b = Add(4,7);
            c = Add(3,8,15);
            Console.WriteLine("Addition is : " + a);
            Console.WriteLine("Addition is : " + b);
            Console.WriteLine("Addition is : " + c);
            Console.ReadLine();
        }
        static int Add(int a)
        {
            return a + a;
        }
        static int Add(int a, int b)
        {
            return a + b;
        }
        static int Add(int a, int b, int c)
        {
            return a + b + c;
        }
    }
}
```

Output:

```
C:\Users\Admin\source\repos\Prac_2Bi\Prac_2Bi\bin\Debug\Prac_2Bi.exe
Addition is : 6
Addition is : 11
Addition is : 26
```

ii) Inheritance (all types):

Acquiring (taking) the properties of one class into another class is called inheritance. Inheritance provides reusability by allowing us to extend an existing class.

The reason behind OOP programming is to promote the reusability of code and to reduce complexity in code and it is possible by using inheritance.

The following are the types of inheritance in C#.

- ❖ Single Inheritance
- ❖ Multiple Inheritance
- ❖ Hierarchical inheritance
- ❖ Multilevel Inheritance

The inheritance concept is based on a base class and derived class. Let us see the definition of a base and derived class.

Base class - is the class from which features are to be inherited into another class.

Derived class - it is the class in which the base class features are inherited.

a) Single Inheritance:

It is the type of inheritance in which there is one base class and one derived class.

Source code:

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

namespace Prac_2Bia
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int a, b, num;
            Cubic c= new Cubic();
            Console.WriteLine("Enter the value: ");
            num = Convert.ToInt32(Console.ReadLine());
            a = c.squ(num);
            b = c.cube(num);
            Console.WriteLine("\nSquare of " + num + " is : " + a);
            Console.WriteLine("Cube of " + num + " is : " + b);
            Console.ReadLine();
        }
        class Square
        {
            public int squ(int a)
            {
                return a * a;
            }
        }
        class Cubic : Square
        {
            public int cube(int a)
            {
                return squ(a) * a;
            }
        }
    }
}
```

Output:

```
C:\Users\Admin\source\repos\Prac_2Bia\Prac_2Bia\bin\Debug\Prac_2Bia.exe
Enter the value:
12

Square of 12 is : 144
Cube of 12 is : 1728
```

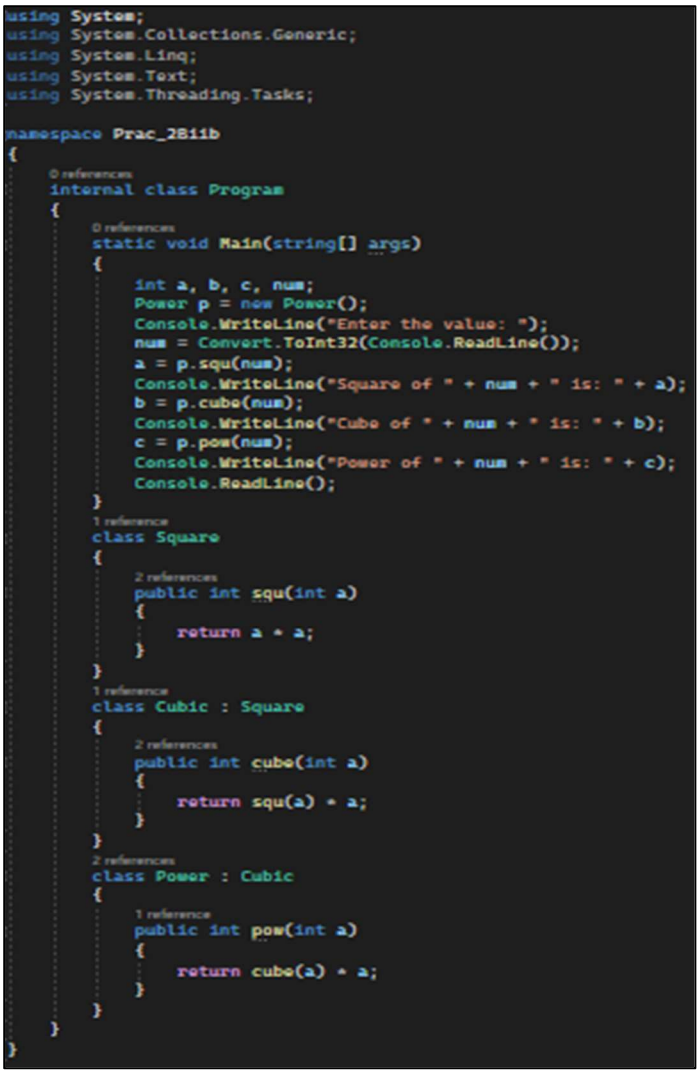
```
C:\Users\Admin\source\repos\Prac_2Bia\Prac_2Bia\bin\Debug\Prac_2Bia.exe
Enter the value:
7

Square of 7 is : 49
Cube of 7 is : 343
```

b) Multiple Inheritance:

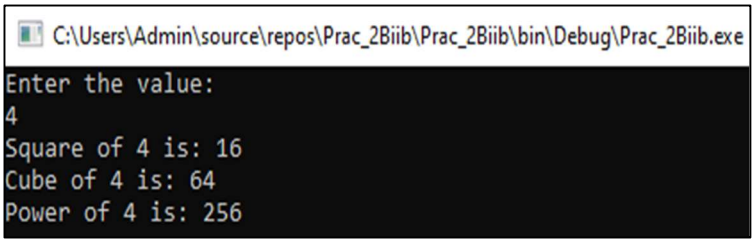
C# does not support multiple inheritances of classes. To overcome this problem we can use interfaces, we will see more about interfaces in my next article in detail.

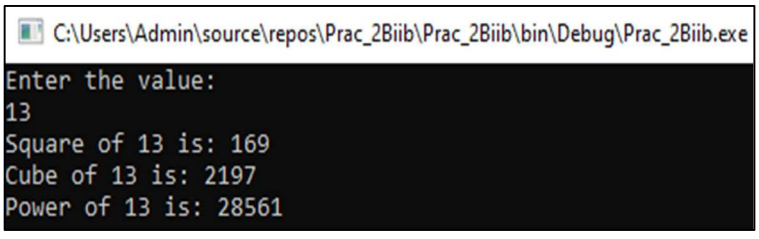
Source code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Prac_2Biib
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int a, b, c, num;
            Power p = new Power();
            Console.WriteLine("Enter the value: ");
            num = Convert.ToInt32(Console.ReadLine());
            a = p.squ(num);
            Console.WriteLine("Square of " + num + "
is: " + a);
            b = p.cube(num);
            Console.WriteLine("Cube of " + num + " is:
" + b);
            c = p.pow(num);
            Console.WriteLine("Power of " + num + "
is: " + c);
            Console.ReadLine();
        }
        class Square
        {
            public int squ(int a)
            {
                return a * a;
            }
        }
        class Cubic : Square
        {
            public int cube(int a)
            {
                return squ(a) * a;
            }

        }
        class Power : Cubic
        {
            public int pow(int a)
            {
                return cube(a) * a;
            }
        }
    }
}
```



Output:





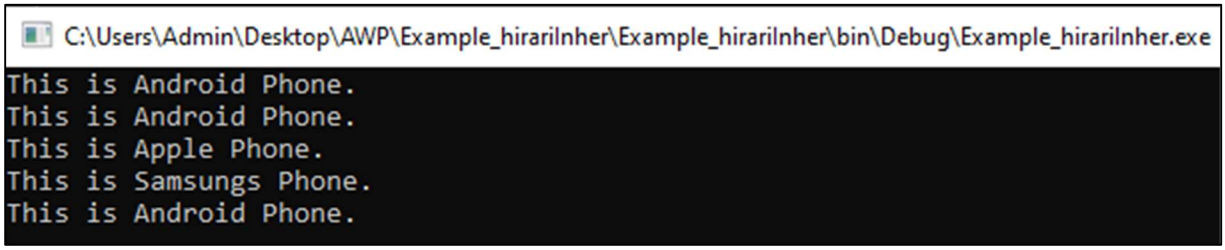
c) Hierarchical Inheritance:

This is the type of inheritance in which there are multiple classes derived from one base class. This type of inheritance is used when there is a requirement of one class feature that is needed in multiple classes.

Source code:

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

namespace Example_hirariInher
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Android a = new Android();
            a.and();
            Apple ap = new Apple();
            ap.and();
            ap.app();
            Samsung s = new Samsung();
            s.sams();
            s.and();
            Console.ReadLine();
        }
        class Android
        {
            public void and()
            {
                Console.WriteLine("This is Android Phone.");
            }
        }
        class Apple: Android
        {
            public void app()
            {
                Console.WriteLine("This is Apple Phone.");
            }
        }
        class Samsung: Android
        {
            public void sams()
            {
                Console.WriteLine("This is Samsungs Phone.");
            }
        }
    }
}
```

Output:



d) Multi-Level Inheritance:

When one class is derived from another derived class then this type of inheritance is called multilevel inheritance.

Source code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Example_multipleInher
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int h, w, area, cost, price;
            Price p = new Price();
            Console.WriteLine("Enter the height: ");
            h = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter the width: ");
            w = Convert.ToInt32(Console.ReadLine());
            p.setHeight(h);
            p.setWidth(w);
            area = p.getArea(h, w);
            Console.WriteLine("Area of Field is: " + area);
            Console.WriteLine("Enter the price of paint: ");
            price = Convert.ToInt32(Console.ReadLine());
            cost = p.getCost(area,price);
            Console.WriteLine("The Price is painting the field is: " + cost);
            Console.ReadLine();
        }
        class Shape
        {
            protected int height;
            protected int width;
            public void setHeight(int h)
            {
                height = h;
            }
            public void setWidth(int w)
            {
                width = w;
            }
        }
        public interface PaintCost
        {
            int getCost(int area, int price);
        }
        class Price: Shape, PaintCost
        {
            public int getArea(int h, int w)
            {
                return h * w;
            }
            public int getCost(int area, int price)
            {
                return area * price;
            }
        }
    }
}
```

Output:

```
C:\Users\Admin\Desktop\AWP\Example_multipleInher\Example_multipleInher\bin\Debug\Example_multipleInher.exe
Enter the height:
5
Enter the width:
4
Area of Field is: 20
Enter the price of paint:
50
The Price is painting the field is: 1000
```

iii) Constructor Overloading:

When more than one constructor with the same name is defined in the same class, they are called overloaded, if the parameters are different for each constructor. Let us see an example to learn how to work with Constructor Overloading in C#. In the example, we have two subjects and a string declaration for Student Name.

Source code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Example_ConstruOverl
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int n1, n2;
            Console.WriteLine("Enter the number 1: ");
            n1 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter the number 1: ");
            n2 = Convert.ToInt32(Console.ReadLine());
            string s;
            Console.WriteLine("Enter the string: ");
            s = Console.ReadLine();
            Add a = new Add(n1, n2);
            a.show();
            Add b = new Add(n1, s);
            b.show1();
            Add c = new Add(n2, s);
            c.show1();
            Console.ReadLine();

        }
        class Add
        {
            int x, y;
            double f;
            string s;
            public Add(int a, double b)
            {
                x = a;
                f = b;
            }
            public Add(int a, string b)
            {
                y = a;
                s = b;
            }
            public void show()
            {
                Console.WriteLine("1st Constructor Overloading: " + (x + f));
            }
            public void show1()
            {
                Console.WriteLine("2nd Constructor Overloading: " + (s + y));
            }
        }
    }
}
```

Output:

```
C:\Users\Admin\Desktop\AWP\Example_ConstruOverl\Example_ConstruOverl\bin\Debug\Example_ConstruOverl.exe
Enter the number 1:
30
Enter the number 1:
13
Enter the string:
Roll no. is:
1st Constructor Overloading: 43
2nd Constructor Overloading: Roll no. is: 30
2nd Constructor Overloading: Roll no. is: 13
```

iv) Interfaces:

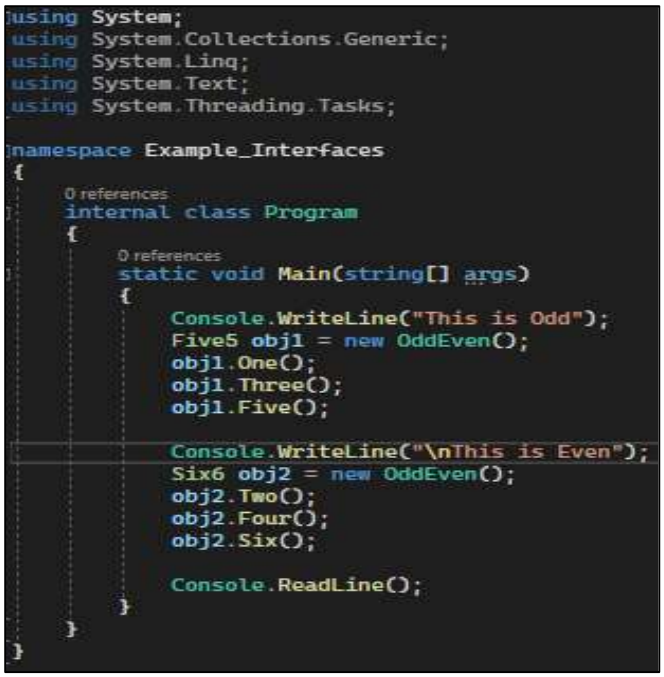
Interface in C# is a blueprint of a class. It is like abstract class because all the methods which are declared inside the interface are abstract methods. It cannot have method body and cannot be instantiated.

Its implementation must be provided by class or struct. The class or struct which implements the interface, must provide the implementation of all the methods declared inside the interface.

Source code:

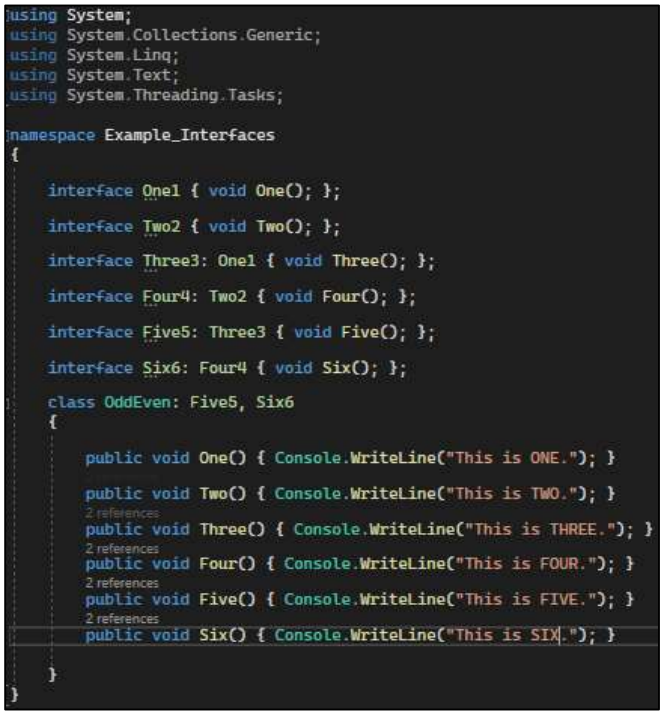
Program.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Example_Interfaces
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("This is Odd");
            Five5 obj1 = new OddEven();
            obj1.One();
            obj1.Three();
            obj1.Five();

            Console.WriteLine("\nThis is Even");
            Six6 obj2 = new OddEven();
            obj2.Two();
            obj2.Four();
            obj2.Six();

            Console.ReadLine();
        }
    }
}
```

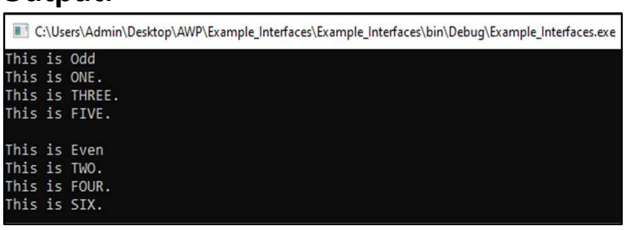


OddEven.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Example_Interfaces
{
    interface One1 { void One(); };
    interface Two2 { void Two(); };
    interface Three3: One1 { void Three(); };
    interface Four4: Two2 { void Four(); };
    interface Five5: Three3 { void Five(); };
    interface Six6: Four4 { void Six(); };
    class OddEven: Five5, Six6
    {
        public void One() { Console.WriteLine("This is ONE."); }
        public void Two() { Console.WriteLine("This is TWO."); }
        public void Three() { Console.WriteLine("This is
THREE."); }
        public void Four() { Console.WriteLine("This is FOUR."); }
        public void Five() { Console.WriteLine("This is FIVE."); }
        public void Six() { Console.WriteLine("This is SIX."); }

    }
}
```



Output:



c) Create simple application to demonstrate use of following concepts:

i) Using Delegates and events:

A Delegate is an abstraction of one or more function pointers. The .NET has implemented the concept of function pointers in the form of delegates. With delegates, you can treat a function as data. Delegates allow functions to be passed as parameters, returned from a function as a value and stored in an array. Delegates have the following characteristics:

They have a signature and a return type. A function that is added to delegates must be compatible with this signature.

Delegates can call methods synchronously and asynchronously.

Source code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Example_delegates
{
    internal class Program
    {
        delegate int numchg(int n);
        public static int num = 12;
        public static int Addnum(int a)
        {
            num = num + a;
            return num;
        }
        public static int Multinum(int a)
        {
            num = num * a;
            return num;
        }
        public static int getNum()
        {
            return num;
        }
        static void Main(string[] args)
        {
            int n;
            Console.WriteLine("Enter the number: ");
            n = Convert.ToInt32(Console.ReadLine());
            numchg nc = new numchg(Addnum);
            numchg nuch = new numchg(Multinum);

            nc(n);
            Console.WriteLine("The addition is: " + getNum());
            nuch(n);
            Console.WriteLine("The multiplication is: " + getNum());
            Console.ReadLine();
        }
    }
}
```

Output:

```
C:\Users\Admin\Desktop\AWP\Example_delegates\Example_delegates\bin\Debug\Example_delegates.exe
Enter the number:
15
The addition is: 27
The multiplication is: 405
```

```
C:\Users\Admin\Desktop\AWP\Example_delegates\Example_delegates\bin\Debug\Example_delegates.exe
Enter the number:
10
The addition is: 22
The multiplication is: 220
```

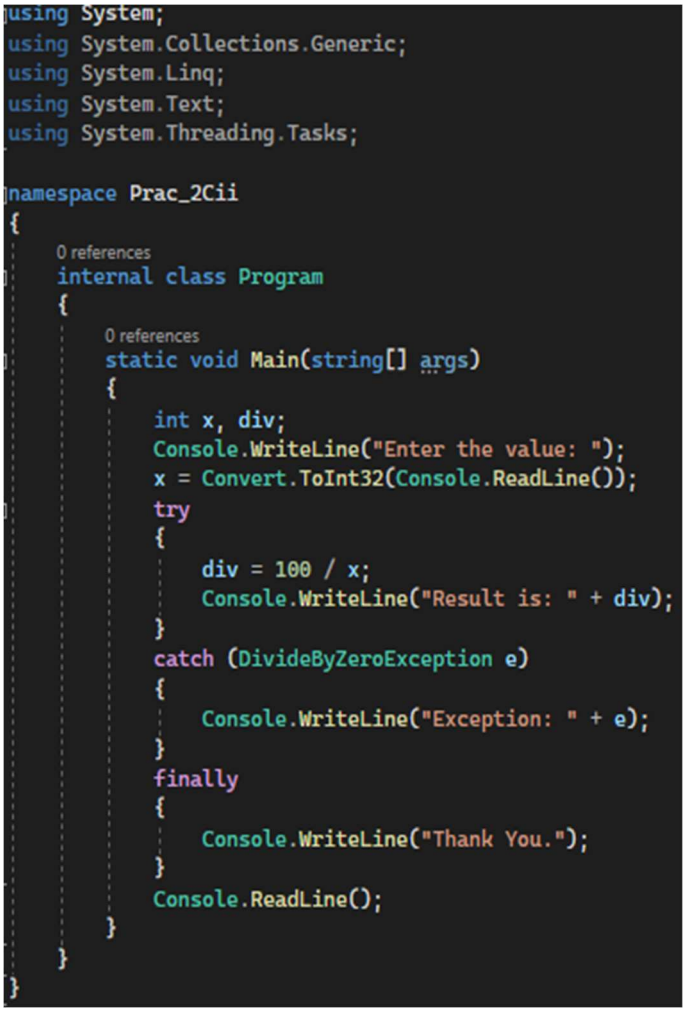
ii) Exception Handling:

Exception Handling in C# is a process to handle runtime errors. We perform exception handling so that normal flow of the application can be maintained even after runtime errors.

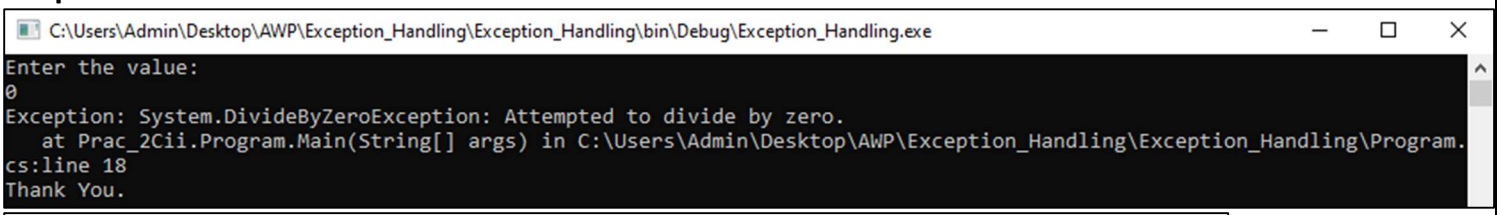
In C#, exception is an event or object which is thrown at runtime. All exceptions are derived from System.Exception class. It is a runtime error which can be handled. If we don't handle the exception, it prints an exception message and terminates the program.

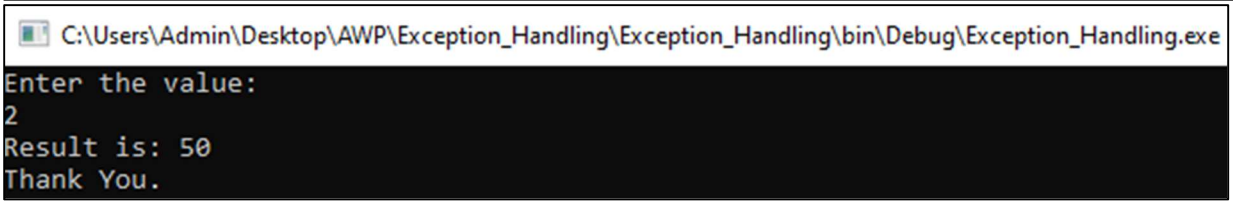
Source code:

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

namespace Prac_2Cii
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int x, div;
            Console.WriteLine("Enter the value: ");
            x = Convert.ToInt32(Console.ReadLine());
            try
            {
                div = 100 / x;
                Console.WriteLine("Result is: " + div);
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine("Exception: " + e);
            }
            finally
            {
                Console.WriteLine("Thank You.");
            }
            Console.ReadLine();
        }
    }
}
```



Output:





Practical 3: Working with Web Forms and Controls

a) Create a simple web page with various sever controls to demonstrate setting and use of their properties. (Example : AutoPostBack)

If we create a web Page, which consists of one or more Web Controls that are configured to use AutoPostBack (Every Web controls will have their own AutoPostBack property), the ASP.Net adds a special JavaScript function to the rendered HTML Page. This function is named _doPostBack() . When Called, it triggers a PostBack, sending data back to the web Server. ASP.NET also adds two additional hidden input fields that are used to pass information back to the server. This information consists of ID of the Control that raised the event and any additional information if needed. These fields will empty initially as shown below,

Source code:

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="AutoPostBack.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .auto-style1 {
            width: 125px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="Student Registration Form: "></asp:Label>
 <table style="width:100%;">
                <tr>
                    <td class="auto-style1">Enter Name: </td>
                    <td>
                        <asp:TextBox ID="TextBox1" runat="server" AutoPostBack="True"></asp:TextBox>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style1">Enter ID: </td>
                    <td>
                        <asp:TextBox ID="TextBox2" runat="server" AutoPostBack="True"></asp:TextBox>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style1">Languages Known:</td>
                    <td>
                        <asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True">
                            <asp:ListItem>Java</asp:ListItem>
                            <asp:ListItem>Python</asp:ListItem>
                            <asp:ListItem>C++</asp:ListItem>
                        </asp:RadioButtonList>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style1">Courses: </td>
                    <td>
                        <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True">
                            <asp:ListItem>BSc.CS</asp:ListItem>
                            <asp:ListItem>BSc.IT</asp:ListItem>
                            <asp:ListItem>BSc.BT</asp:ListItem>
                            <asp:ListItem>BSc.Science</asp:ListItem>
                        </asp:DropDownList>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style1">
                        <asp:Button ID="Button1" runat="server" Text="Submit" Width="76px"
BorderStyle="Solid" OnClick="Button1_Click" />
```

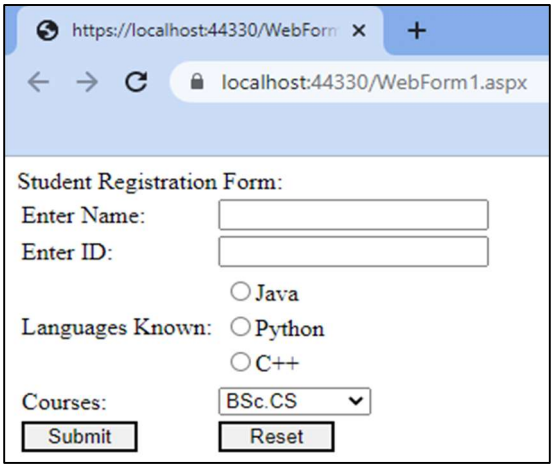
```
                    </td>
                    <td>
                        <asp:Button ID="Button2" runat="server" Text="Reset" Width="76px"
BorderStyle="Solid" />
                    </td>
                </tr>
            </table>
            <br />
            <asp:Label ID="stu" runat="server"></asp:Label>
            <br />
            <asp:Label ID="name" runat="server"></asp:Label>
            <br />
            <asp:Label ID="id" runat="server"></asp:Label>
            <br />
            <asp:Label ID="lang" runat="server"></asp:Label>
            <br />
            <asp:Label ID="course" runat="server"></asp:Label>
            <br />
            <asp:Label ID="thank" runat="server" Text="Thank You!"></asp:Label>
            <br />
            <br />
            <br />
            <br />
            <br />

        </div>
    </form>
</body>
</html>
```

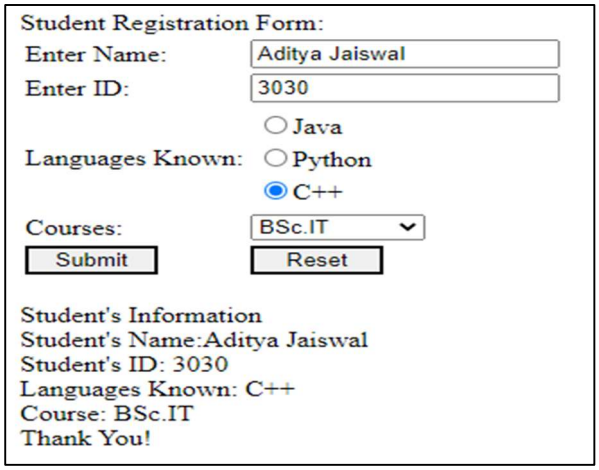
WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace AutoPostBack
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            stu.Text = "Student's Information";
            name.Text = "Student's Name:" + TextBox1.Text;
            id.Text = "Student's ID: " + TextBox2.Text;
            lang.Text = "Languages Known: " + RadioButtonList1.SelectedValue;
            course.Text = "Course: " + DropDownList1.SelectedItem;
            thank.Text = "Thank You!";
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace AutoPostBack
{
    2 references
    public partial class WebForm1 : System.Web.UI.Page
    {
        0 references
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        0 references
        protected void Button1_Click(object sender, EventArgs e)
        {
            stu.Text = "Student's Information";
            name.Text = "Student's Name:" + TextBox1.Text;
            id.Text = "Student's ID: " + TextBox2.Text;
            lang.Text = "Languages Known: " + RadioButtonList1.SelectedValue;
            course.Text = "Course: " + DropDownList1.SelectedItem;
            thank.Text = "Thank You!";
        }
    }
}
```

Output:





b) Demonstrate the use of Calendar control to perform following operations.

i) Display messages in a calendar control

ii) Display vacation in a calendar control

iii) Selected day in a calendar control using style

iv) Difference between two calendar dates

The calendar control is a functionally rich web control, which provides the following capabilities:
Displaying one month at a time
Selecting a day, a week or a month
Selecting a range of days
Moving from month to month
Controlling the display of the days programmatically
The basic syntax of a calendar control is:
<asp:Calender ID = "Calendar1" runat = "server">
</asp:Calender>

Source code:

WebForm1.aspx code:

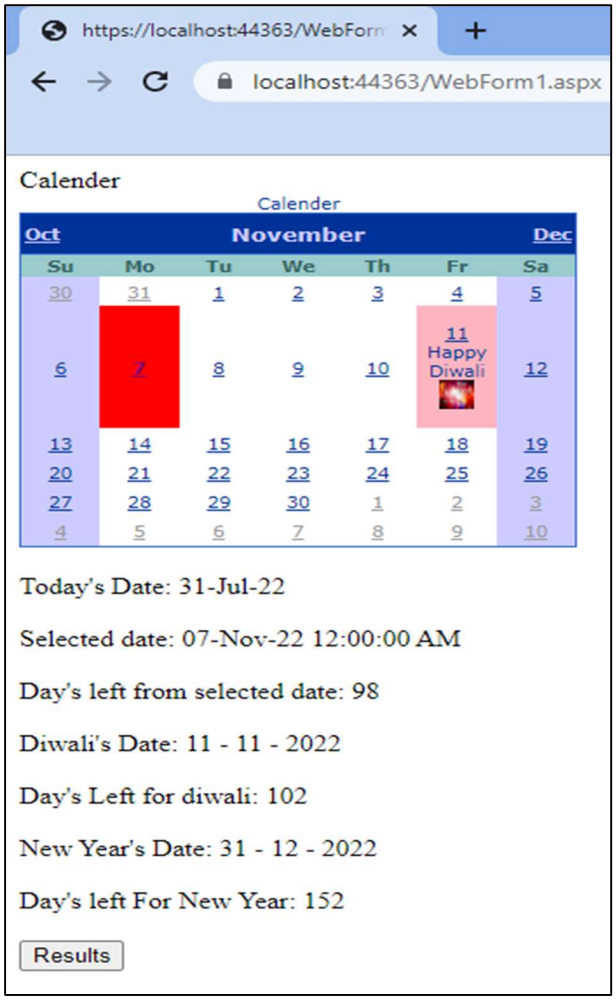
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="CalenderControl.WebForm1" %>

<!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="Calender"></asp:Label>
              <br />
              <asp:Calendar ID="Calendar1" runat="server" BackColor="White" BorderColor="#3366CC"
BorderWidth="1px" CellPadding="1" DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt"
ForeColor="#003399" Height="230px" OnDayRender="Calendar1_DayRender" Width="320px">
              <DayHeaderStyle BackColor="#99CCCC" ForeColor="#336666" Height="1px" />
              <NextPrevStyle Font-Size="8pt" ForeColor="#CCCCFF" />
              <OtherMonthDayStyle ForeColor="#999999" />
              <SelectedDayStyle BackColor="Red" Font-Bold="True" ForeColor="Purple" />
              <SelectorStyle BackColor="#99CCCC" ForeColor="#336666" />
              <TitleStyle BackColor="#003399" BorderColor="#3366CC" BorderWidth="1px" Font-Bold="True"
Font-Size="10pt" ForeColor="#CCCCFF" Height="25px" />
              <TodayDayStyle BackColor="Blue" ForeColor="White" />
              <WeekendDayStyle BackColor="#CCCCFF" />

              </asp:Calendar>
              <br />
              <asp:Label ID="Label2" runat="server" Text="Today's Date: "></asp:Label>
              <asp:Label ID="today" runat="server"></asp:Label>
              <br />
              <br />
              <asp:Label ID="Label7" runat="server" Text="Selected date: "></asp:Label>
              <asp:Label ID="select" runat="server"></asp:Label>
              <br />
              <br />
              <asp:Label ID="Label8" runat="server" Text="Day's left from selected date: "></asp:Label>
              <asp:Label ID="sday" runat="server"></asp:Label>
              <br />
              <br />
              <asp:Label ID="Label3" runat="server" Text="Diwali's Date: "></asp:Label>
              <asp:Label ID="diwali" runat="server"></asp:Label>
              <br />
              <br />
              <asp:Label ID="Label4" runat="server" Text="Day's Left for diwali: "></asp:Label>
              <asp:Label ID="daysleft" runat="server"></asp:Label>
```

```
            <br />
            <br />
            <asp:Label ID="Label5" runat="server" Text="New Year's Date: "></asp:Label>
            <asp:Label ID="newyear" runat="server"></asp:Label>
            <br />
            <br />
            <asp:Label ID="Label6" runat="server" Text="Day's left For New Year: "></asp:Label>
            <asp:Label ID="nydays" runat="server"></asp:Label>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Results" />
        </div>
        </form>
</body>
</html>
```

WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CalenderControl
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
        {

            if (e.Day.Date.Day == 11 & e.Day.Date.Month == 11)
            {
                e.Cell.BackColor = System.Drawing.Color.LightPink;
                Label lbl = new Label();
                lbl.Text = "<br/>Happy Diwali";
                e.Cell.Controls.Add(lbl);
                Image img = new Image();
                img.ImageUrl = "~/Images/diwali.jpg";
                img.Height = Unit.Pixel(20);
                img.Width = Unit.Pixel(20);
                e.Cell.Controls.Add(img);
            }


        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            Calendar1.Caption = "Calender";
            Calendar1.FirstDayOfWeek = FirstDayOfWeek.Sunday;
            Calendar1.NextPrevFormat = NextPrevFormat.ShortMonth;
            Calendar1.TitleFormat = TitleFormat.Month;
            today.Text = Calendar1.TodaysDate.ToShortDateString();
            select.Text = Calendar1.SelectedDate.ToString();
            DateTime now = Calendar1.SelectedDate;
            DateTime after = DateTime.Now;
            TimeSpan d = now - after;
            sday.Text = d.Days.ToString();
            diwali.Text = "11 - 11 - 2022";
            TimeSpan a = new DateTime(2022, 11, 11) - DateTime.Now;
            daysleft.Text = a.Days.ToString();
            newyear.Text = "31 - 12 - 2022";
            TimeSpan d1 = new DateTime(2022, 12, 31) - DateTime.Now;
            nydays.Text = d1.Days.ToString();
        }
    }
}
```

Output:



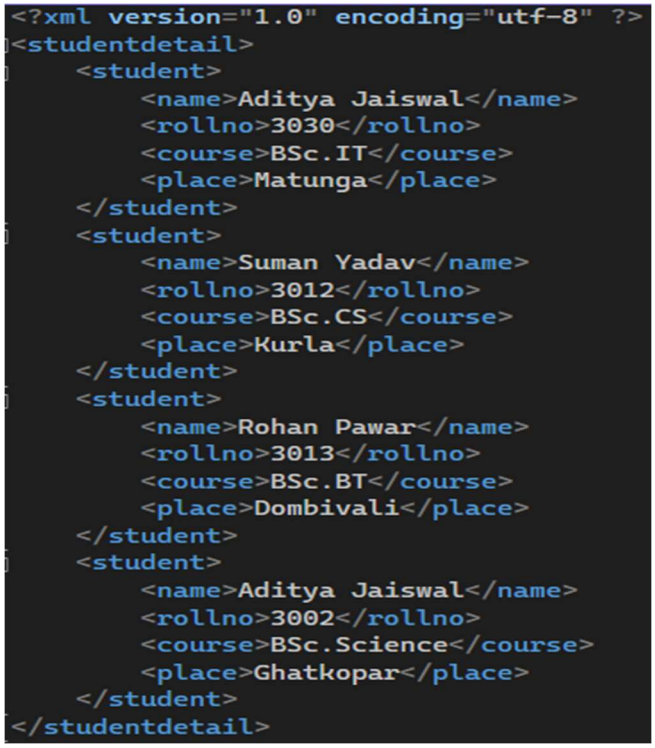
c) Demonstrate the use of Treeview control performs following operations.

i) Treeview Control and Datalist ii) Treeview Operations

The TreeView control is used to display a hierarchical representation of the same data in a tree structure. The top-level in the tree view is the root node with one or more child nodes. In addition, the root node can be contracted or expanded by clicking on the plus sign (+) button. It is also useful to provide the full path of the root node to the child node. Let's create a TreeView control in the VB.NET Windows form using the following steps.

Step 1: We have to find the TreeView control from the toolbox and then drag and drop the TreeView control onto the window form, as shown below.

Step 2: Once the TreeView is added to the form, we can set various properties of the TreeView by clicking on the TreeView control.



Source code:

XMLFile1.xml code:

```
<?xml version="1.0" encoding="utf-8" ?>
<studentdetail>
	<student>
		<name>Aditya Jaiswal</name>
		<rollno>3030</rollno>
		<course>BSc.IT</course>
		<place>Matunga</place>
	</student>
	<student>
		<name>Suman Yadav</name>
		<rollno>3012</rollno>
		<course>BSc.CS</course>
		<place>Kurla</place>
	</student>
	<student>
		<name>Rohan Pawar</name>
		<rollno>3013</rollno>
		<course>BSc.BT</course>
		<place>Dombivali</place>
	</student>
	<student>
		<name>Aditya Jaiswal</name>
		<rollno>3002</rollno>
		<course>BSc.Science</course>
		<place>Ghatkopar</place>
	</student>
</studentdetail>
```

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="TreeviewAndDatalist.WebForm1" %>
<!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="TreeView: "></asp:Label>

            <asp:TreeView ID="TreeView1" runat="server">
                <Nodes>
                    <asp:TreeNode Text="RJ College" Value="RJ College" NavigateUrl="https://www.rjcollege.edu.in/">
                        <asp:TreeNode Text="Junior College" Value="Junior College"
NavigateUrl="https://junior.rjcollege.edu.in/">
                            <asp:TreeNode Text="11th " Value="11th" NavigateUrl="https://studywoo.com/first-year-
junior-colleges/"></asp:TreeNode>
                            <asp:TreeNode Text="12th " Value="12th"
NavigateUrl="https://ythi.net/abbreviations/english/what-does-syjc-mean-what-is-the-full-form-of-syjc/"></asp:TreeNode>
```

```
                        </asp:TreeNode>
                        <asp:TreeNode Text="Degree College" Value="Degree College">
                            <asp:TreeNode Text="BSc.IT" Value="BSc.IT" NavigateUrl="https://leverageedublog.s3.ap-
south-1.amazonaws.com/blog/wp-content/uploads/2020/06/08065913/BSc-IT-Scope.jpg"></asp:TreeNode>
                            <asp:TreeNode Text="BSc.CS" Value="BSc.CS"
NavigateUrl="https://www.galaxyeduworld.com/storage/blogs/bsc-cs-blog-1.jpg"></asp:TreeNode>
                            <asp:TreeNode Text="BSc.BT" Value="BSc.BT"
NavigateUrl="https://www.blalbiotech.com/blog/wp-content/uploads/2022/05/bsc-biotechnology-
graduates.jpg"></asp:TreeNode>
                        </asp:TreeNode>
                    </asp:TreeNode>
                    <asp:TreeNode Text="Books" Value="Books">
                        <asp:TreeNode Text="Reading Books" Value="Reading Books"
NavigateUrl="https://parade.com/1167905/marynliles/quotes-about-reading-books/"></asp:TreeNode>
                        <asp:TreeNode Text="Reference Books" Value="Reference Books"
NavigateUrl="https://deligram.com/category/books-stationeries-books-reference-books"></asp:TreeNode>
                        <asp:TreeNode Text="Text Books" Value="Text Books"
NavigateUrl="https://www.canstockphoto.com/stack-of-textbooks-4657937.html"></asp:TreeNode>
                        <asp:TreeNode Text="Guide Books" Value="Guide Books"
NavigateUrl="https://www.collinsdictionary.com/dictionary/english/guidebook"></asp:TreeNode>
                    </asp:TreeNode>
                </Nodes>
            </asp:TreeView>
            <br />
            <asp:Label ID="Label2" runat="server" Text="DataList: "></asp:Label>
            <br />
            <asp:DataList ID="DataList1" runat="server">
                <ItemTemplate>
                    Name: <%# Eval("name") %>
                    <br />
                    Roll no: <%# Eval("rollno") %>
                    <br />
                    Courses: <%# Eval("course") %>
                    <br />
                    Place: <%# Eval("place") %>
                    <br />
                    <br />
                </ItemTemplate>
            </asp:DataList>
            <br />
            <br />
        </div>
    </form>
</body>
</html>
```

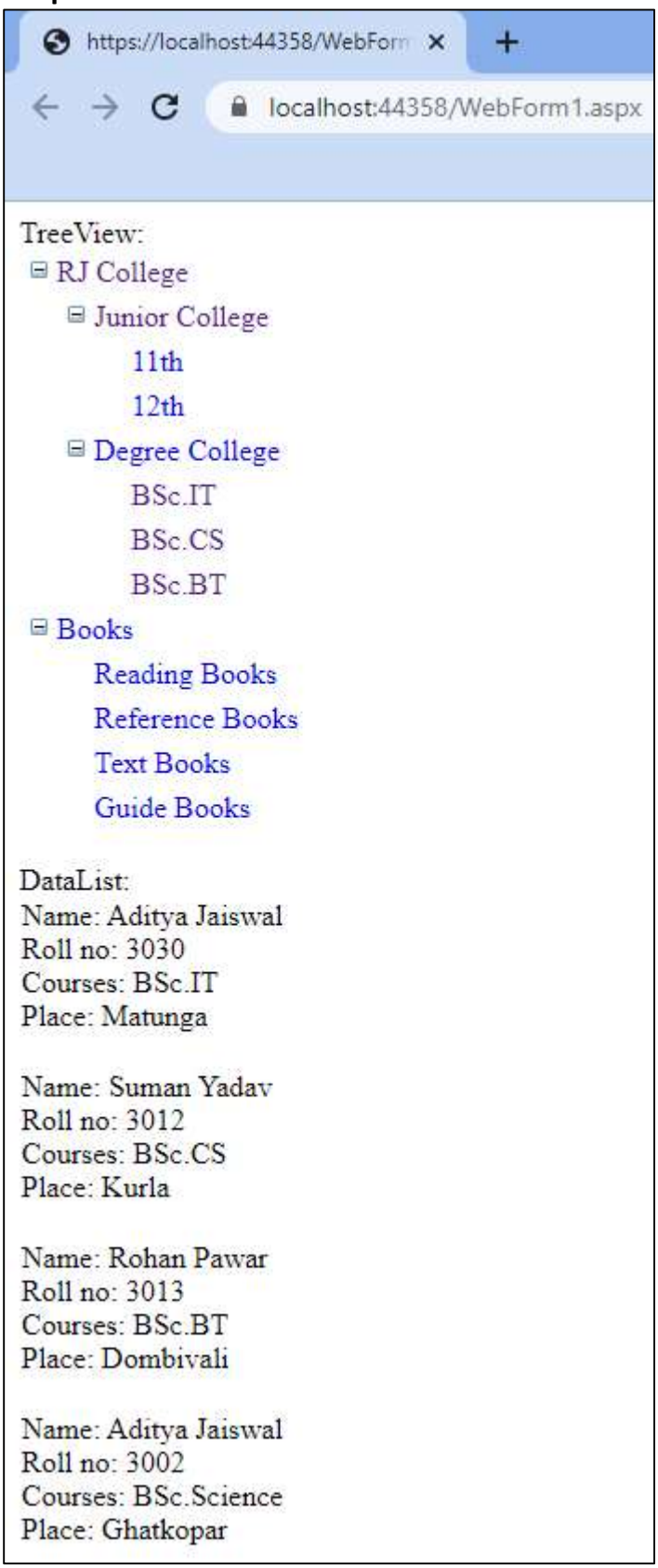
WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
namespace TreeviewAndDatalist
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                bindData();
            }
        }

        protected void bindData()
        {
            DataSet ds = new DataSet();
            ds.ReadXml(Server.MapPath("XMLFile1.xml"));
            if (ds != null && ds.HasChanges())
            {
                DataList1.DataSource = ds;
                DataList1.DataBind();
            }
            else
            {
                DataList1.DataBind();
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

namespace TreeviewAndDatalist
{
    2 references
    public partial class WebForm1 : System.Web.UI.Page
    {
        0 references
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                bindData();
            }
        }

        1 reference
        protected void bindData()
        {
            DataSet ds = new DataSet();
            ds.ReadXml(Server.MapPath("XMLFile1.xml"));
            if (ds != null && ds.HasChanges())
            {
                DataList1.DataSource = ds;
                DataList1.DataBind();
            }
            else
            {
                DataList1.DataBind();

            }

        }
    }
}
```

```
            }

        }
    }
}
```

Output:


https://localhost:44358/WebForm
localhost:44358/WebForm1.aspx
TreeView:
RJ College
Junior College
11th
12th
Degree College
BSc.IT
BSc.CS
BSc.BT
Books
Reading Books
Reference Books
Text Books
Guide Books
DataList:
Name: Aditya Jaiswal
Roll no: 3030
Courses: BSc.IT
Place: Matunga
Name: Suman Yadav
Roll no: 3012
Courses: BSc.CS
Place: Kurla
Name: Rohan Pawar
Roll no: 3013
Courses: BSc.BT
Place: Dombivali
Name: Aditya Jaiswal
Roll no: 3002
Courses: BSc.Science
Place: Ghatkopar


LOREM IPSUM


B.Sc. BIOTECHNOLOGY
SCOPE, JOBS AND SALARY
IN 2022

Practical 4: Working with Form Controls

a) Create a Registration form to demonstrate use of various Validation controls.

ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

Source code:

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Validation.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .auto-style1 {
            width: 100%;
            height: 0px;
        }
        .auto-style2 {
            width: 160px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <br />
            <table class="auto-style1">
                <tr>
                    <td class="auto-style2">
                        <asp:Label ID="Label1" runat="server" Text="Enter Name: "></asp:Label>
                    </td>
                    <td>
                        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="TextBox1" ErrorMessage="PLease Enter the Name" ForeColor="Red"></asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style2">
                        <asp:Label ID="Label2" runat="server" Text="Age: "></asp:Label>
                    </td>
                    <td>
                        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
ControlToValidate="TextBox2" ErrorMessage="Please Enter the Age" ForeColor="Red"></asp:RequiredFieldValidator>
                        <asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="TextBox2"
ErrorMessage="Please enter the age between 20 to 60" ForeColor="Red" MaximumValue="60"
MinimumValue="20"></asp:RangeValidator>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style2">
                        <asp:Label ID="Label3" runat="server" Text="Email Id:"></asp:Label>
                    </td>
                    <td>
                        <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
ControlToValidate="TextBox3" ErrorMessage="Please Enter the email id" ForeColor="Red"></asp:RequiredFieldValidator>
                        <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
ControlToValidate="TextBox3" ErrorMessage="Please enter the correct email" ForeColor="Red" ValidationExpression="\w+([-
+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"></asp:RegularExpressionValidator>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style2">
                        <asp:Label ID="Label4" runat="server" Text="Password: "></asp:Label>
                    </td>
                    <td>
                        <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ControlToValidate="TextBox4" ErrorMessage="Please Enter the password" ForeColor="Red"></asp:RequiredFieldValidator>
                        <asp:CustomValidator ID="CustomValidator1" runat="server"
ClientValidationFunction="CustomValidator1_ServerValidate" ControlToValidate="TextBox4" ErrorMessage="Please enter only
8 characters" ForeColor="Red" OnServerValidate="CustomValidator1_ServerValidate"></asp:CustomValidator>
                    </td>
```

```
                </tr>
                <tr>
                    <td class="auto-style2">
                        <asp:Label ID="Label5" runat="server" Text="Confirm Password: "></asp:Label>
                    </td>
                    <td>
                        <asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server"
ControlToValidate="TextBox5" ErrorMessage="Please enter the password" ForeColor="Red"></asp:RequiredFieldValidator>
                        <asp:CompareValidator ID="CompareValidator1" runat="server" ControlToCompare="TextBox4"
ControlToValidate="TextBox5" ErrorMessage="Please enter the same Password" ForeColor="Red"></asp:CompareValidator>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style2">
                        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button" />
                    </td>
                    <td> </td>
                </tr>
            </table>
            <asp:ValidationSummary ID="ValidationSummary1" runat="server" ForeColor="Red" />
            <br />
            <br />
            <br />
        </div>
    </form>
</body>
</html>
```

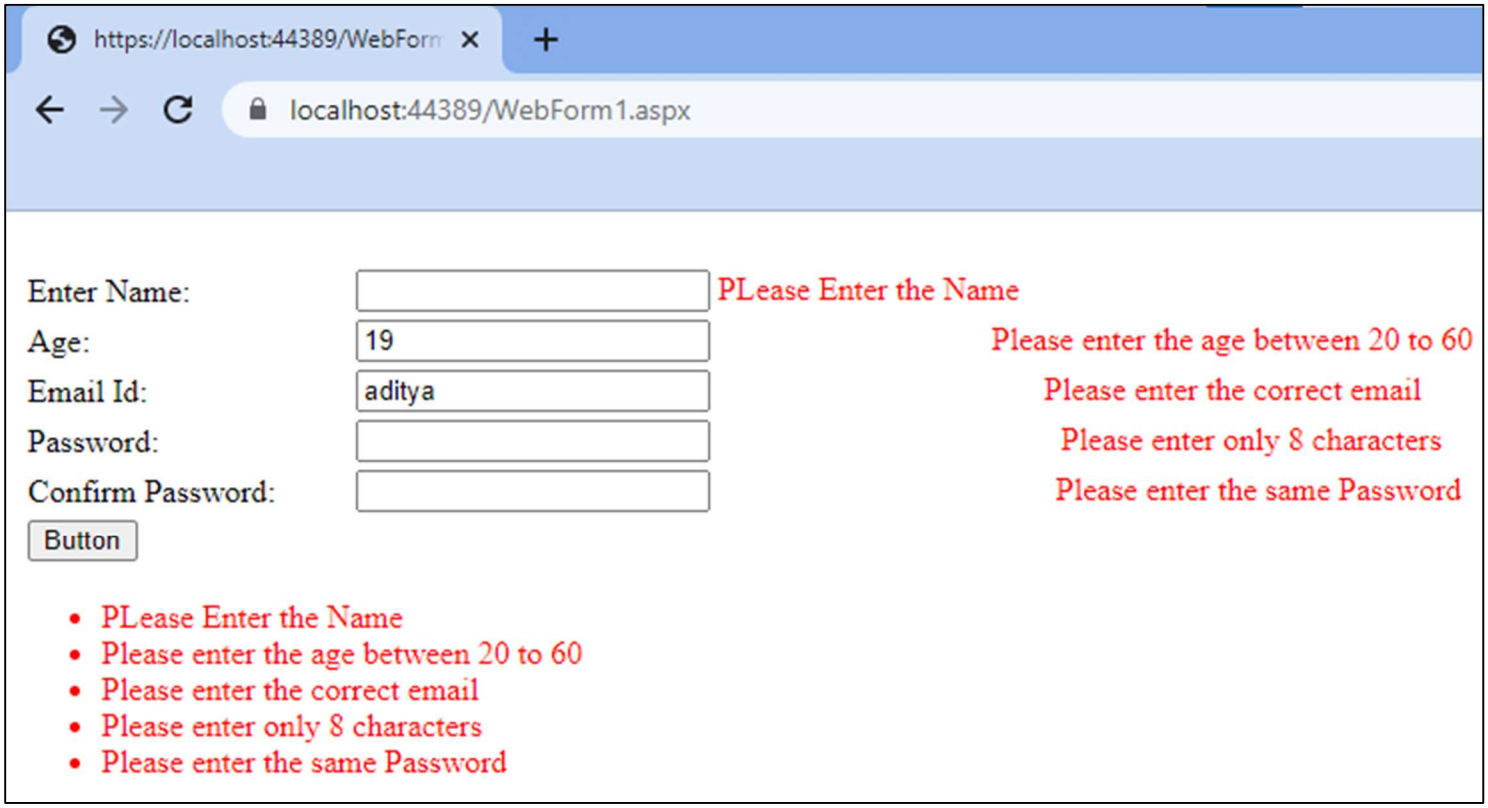
WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Validation
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs e)
        {
            if (e.Value.Length == 8)
            {
                e.IsValid = true;
            }
            else
            {
                e.IsValid = false;
            }
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            if (Page.IsValid)
            {
                Console.WriteLine("Thank You!");
            }
        }
    }
}
```

Web.config code:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2" />
    <httpRuntime targetFramework="4.7.2" />
  </system.web>
        <appSettings>
                <add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />
        </appSettings>
  <system.codedom>
    <compilers>
      <compiler language="c#;cs;csharp" extension=".cs"
type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
warningLevel="4" compilerOptions="/langversion:default /nowarn:1659;1699;1701" />
      <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
warningLevel="4" compilerOptions="/langversion:default /nowarn:41008 /define:_MYTYPE=\&quot;Web\&quot; /optionInfer+"
/>
    </compilers>
  </system.codedom>
</configuration>
```

Output:



b) Create Web Form to demonstrate use of Adrotator Control:

The AdRotator control randomly selects banner graphics from a list, which is specified in an external XML schedule file. This external XML schedule file is called the advertisement file.

The AdRotator control allows you to specify the advertisement file and the type of window that the link should follow in the AdvertisementFile and the Target property respectively.

Source code:

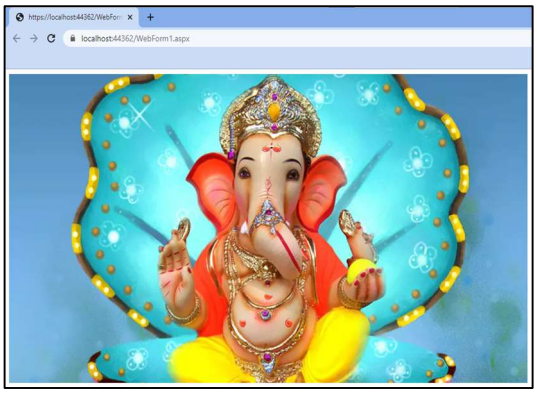
Webform1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="adrotator.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:AdRotator ID="AdRotator1" runat="server" AdvertisementFile="~/XMLFile1.xml" />
        </div>
    </form>
</body>
</html>
```

XMLFile1.xml code:

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
        <Ad>
                <ImageUrl>~/Images/diwali.jpg</ImageUrl>
                <NavigateUrl>https://images.indianexpress.com/2020/11/diwali-feature-1.jpg</NavigateUrl>
                <AltText>Happy Diwali</AltText>
                <Impression>2</Impression>
                <Keyword>Diwali</Keyword>
        </Ad>
        <Ad>
                <ImageUrl>~/Images/ganpati.jpg</ImageUrl>
                <NavigateUrl>https://www.bajajfinservmarkets.in/discover/wp-content/uploads/2019/09/2.png</NavigateUrl>
                <AltText>Ganpati Bappa Morya</AltText>
                <Impression>4</Impression>
                <Keyword>Ganesh Utsav</Keyword>
        </Ad>
        <Ad>
                <ImageUrl>~/Images/newyear.jpg</ImageUrl>
                <NavigateUrl>https://st2.depositphotos.com/1938335/5673/v/450/depositphotos_56730693-stock-
illustration-merry-christmas-and-happy-new.jpg</NavigateUrl>
                <AltText>Happy New Year</AltText>
                <Impression>6</Impression>
                <Keyword>Newyear</Keyword>
        </Ad>
</Advertisements>
```

Output:



c) Create Web Form to demonstrate use User Controls:

ASP.NET allows the users to create controls. These user defined controls are categorised into:

- User controls
- Custom controls

Source code:

WebUserControl1.aspx code:

```
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="WebUserControl1.ascx.cs"
Inherits="usercontrols.WebUserControl1" %>
<table>
    <tr>
        <td>
            <fieldset>
                <asp:Label ID="lblMessage" runat="server"></asp:Label>
            </fieldset>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="lblName" runat="server" Text="Name"></asp:Label>
        </td>
        <td>
            <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="lblAddress" runat="server" Text="Address"></asp:Label>
        </td>
        <td>
            <asp:TextBox ID="txtAddress" runat="server"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="lblPhone" runat="server" Text="Phone"></asp:Label>
        </td>
        <td>
            <asp:TextBox ID="txtPhone" runat="server"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="lblEmail" runat="server" Text="Email"></asp:Label>
        </td>
        <td>
            <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit_Click" />
        </td>
    </tr>
</table>
```

WebUserControl1.aspx.cs code:

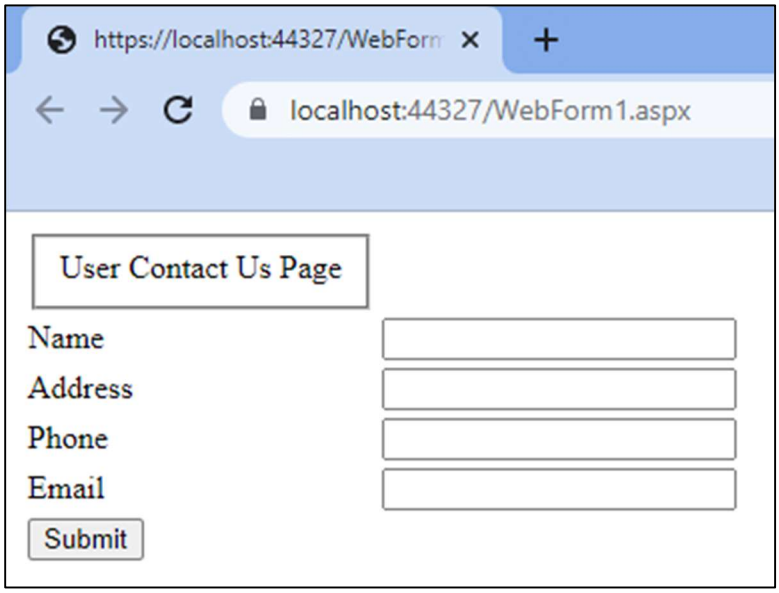
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace usercontrols
```

```
{
    public partial class WebUserControl1 : System.Web.UI.UserControl
    {
        private string _header;
        public string Header
        {
            get { return _header; }
            set { _header = value; }
        }
        protected void Page_Load(object sender, EventArgs e)
        {
            lblMessage.Text = _header;
        }

        protected void btnSubmit_Click(object sender, EventArgs e)
        {

        }
    }
}
```

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="usercontrols.WebForm1" %>

<%@ Register Src="~/WebUserControl1.ascx" TagPrefix="uc1" TagName="WebUserControl1" %>


<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <uc1:WebUserControl1 runat="server" id="WebUserControl1" Header="User Contact Us Page"
/>
        </div>
    </form>
</body>
</html>
```

Output:



Practical 5: Working with Navigation, Beautification and Master page:

a) Create Web Form to demonstrate use of Website Navigation controls and Site Map:

With the release of ASP.NET 2.0, a new set of controls were made available to help provide navigational elements to your web sites. These controls can be found in the Navigation tab of the Visual Studio Toolbox. These controls can be used to create menus and other navigational aids on ASP.NET Web pages.

Source code:

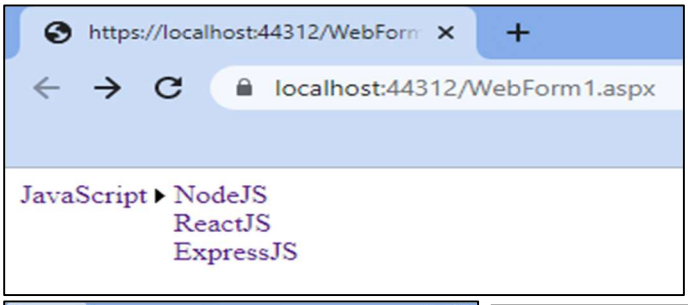
Web.Sitemap code:

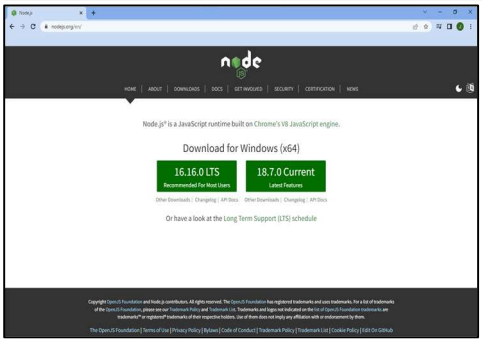
```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="https://www.javascript.com/" title="JavaScript"  description="JavaScript is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well.">
    <siteMapNode url="https://nodejs.org/en/" title="NodeJS"  description="Node. js (Node) is an open source development platform for executing JavaScript code server-side." />
    <siteMapNode url="https://reactjs.org/" title="ReactJS"  description="React (also known as React. js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components." />
      <siteMapNode url="https://expressjs.com/" title="ExpressJS"  description="ExpressJS is a web application framework that provides you with a simple API to build websites, web apps and back ends. With ExpressJS, you need not worry about low level protocols, processes, etc." />
  </siteMapNode>
</siteMap>
```

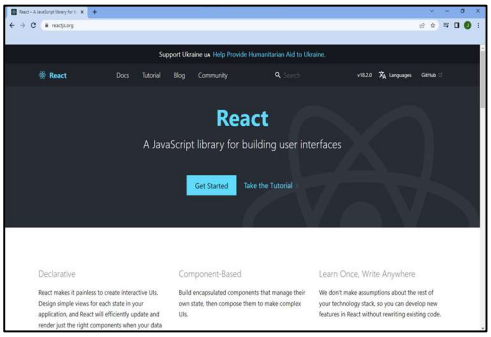
WebForm1.aspx code:

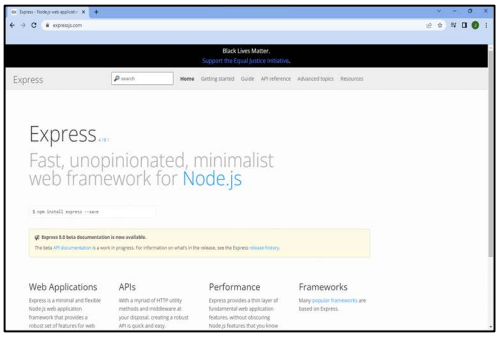
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="Sitemap.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
            <asp:Menu ID="Menu1" runat="server" DataSourceID="SiteMapDataSource1"></asp:Menu>
            <br />
        </div>
    </form>
</body>
</html>
```

Output:









b) Create a web application to demonstrate use of Master Page with applying Styles and Themes for page beautification.

It is an ASP.NET file with the. master extension with a predefined layout that can include static text, HTML elements and server controls. A Master page offers a template for one or more web forms. It defines placeholders for the content, which can be overridden by the content pages. The content pages contain only content.

Source code:

Master Page: Site1.master code:

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site1.master.cs" Inherits="Prac_5B.Site1" %>

<!DOCTYPE html>

<html>
<head runat="server">
    <title>Master | Home Page</title>
    <link href="StyleSheet1.css" rel="stylesheet" />
</head>
<body>
    <div class="headerdiv">
        <header class="header">
            <div class="mainheading"><h2>Diwali Shopping Site</h2></div>
            <div class="head">
                <ul>
                    <li class="pages"><a href="home.aspx">Home</a></li>
                    <li class="pages"><a href="about.aspx">About</a></li>
                    <li class="pages"><a href="contact.aspx">Contact</a></li>
                    <li class="pages"><a href="Site1.master">Master</a></li>
                </ul>
            </div>
        </header>
    </div>

    <form id="form1" runat="server">
        <div class="contentpage">
            <asp:ContentPlaceHolder
ID="ContentPlaceHolder1" runat="server">
                <p>This Is Master Page.</p><br />
                <img src="Images/master.jpg" />
            </asp:ContentPlaceHolder>
        </div>
    </form>
    <div>
        <footer>
            <div class="foot">
                <h5>Copyright @ C# Corner.</h5>
            </div>
        </footer>
    </div>

</body>
</html>
```

CSS for Master page:

StyleSheet1.css code:

```
body {
    margin: 0px;
    padding: 0px;
}

.headerdiv{
    text-align: center;
}

.headerdiv .head ul {
```

```
    display: flex;
    align-items: center;
    justify-content: center;
    margin-left: -30px;
}

.headerdiv .head ul li {
    list-style: none;
    display: block;
    font-size: 22px;
}

.headerdiv  .head ul li a {
    color: black;
    border-radius: 20px;
    text-decoration: none;
    padding: 3px 30px;
}

.headerdiv .head ul li a:hover {
     color: lightblue;
     background-color: lightgoldenrodyellow;
}


.contentpage{
    text-align: center;
    font-size: 40px;
    font-family: Algerian;
}

.contentpage img{
    height: 300px;
    width: 300px;
    margin-top: -30px
}



.foot h5{
    font-family: Calibri;
    font-size: 24px;
    text-align: center;
}
```

```
body {
    margin: 0px;
    padding: 0px;
}

.headerdiv{
    text-align: center;
}

.headerdiv .head ul {
    display: flex;
    align-items: center;
    justify-content: center;
    margin-left: -30px;
}

.headerdiv .head ul li {
    list-style: none;
    display: block;
    font-size: 22px;
}

.headerdiv  .head ul li a {
    color: black;
    border-radius: 20px;
    text-decoration: none;
    padding: 3px 30px;
}

.headerdiv .head ul li a:hover {
     color: lightblue;
     background-color: lightgoldenrodyellow;
}


.contentpage{
    text-align: center;
    font-size: 40px;
    font-family: Algerian;
}

.contentpage img{
    height: 300px;
    width: 300px;
    margin-top: -30px
}



.foot h5{
    font-family: Calibri;
    font-size: 24px;
    text-align: center;
}
```

Home.aspx code:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="home.aspx.cs" Inherits="Prac_5B.WebForm1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div>
        <p>This is Home Page.</p>
        <img src="Images/home.png" />
    </div>
</asp:Content>
```

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true" CodeBehind="home.aspx.cs"
    Inherits="Prac_5B.WebForm1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div>
        <p>This is Home Page.</p>
        <img src="Images/home.png" />
    </div>
</asp:Content>
```

About.aspx code:

```
<%@ Page Title="" Language="C#"
MasterPageFile="~/Site1.Master" AutoEventWireup="true" CodeBehind="about.aspx.cs"
Inherits="Prac_5B.WebForm2" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
```

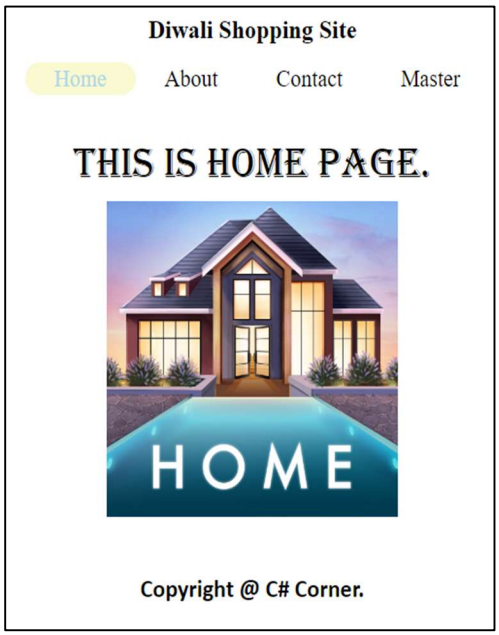
```
    <div>
        <p>This is About Page.</p>
        <img src="Images/about.jpg" />
    </div>
</asp:Content>
```

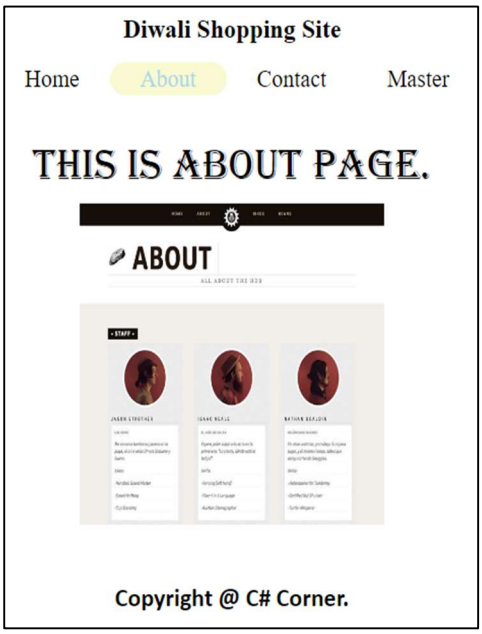
```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true" CodeBehind="about.aspx.cs"
    Inherits="Prac_5B.WebForm2" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div>
        <p>This is About Page.</p>
        <img src="Images/about.jpg" />
    </div>
</asp:Content>
```

Contact.aspx code:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="contact.aspx.cs" Inherits="Prac_5B.contact" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div class="contact-box">
        <p>This is Contact Us Page.</p>
                <div class="contacts" style="font-size: 22px; font-family: Calibri;">
                    <label for="name">Name: </label>
                    <input type="text" name="name" id="name" placeholder="Enter you name">
                </div>
                <div class="contacts" style="font-size: 22px; font-family: Calibri;">
                    <label for="email">Email: </label>
                    <input type="email" name="email" id="email" placeholder="Enter you email">
                </div>
                <div class="contacts" style="font-size: 22px; font-family: Calibri;">
                    <label for="phone">Phone Number: </label>
                    <input type="text" name="phone" id="phone" placeholder="Enter you phone number">
                </div>
                <div class="contacts" style="font-size: 22px; font-family: Calibri;">
                    <label for="message">Message: </label>
                    <textarea name="message" id="Message" cols="30" rows="10"></textarea>
                </div>
                <div class="content" style="font-size: 34px; font-family: Calibri;">
                    <div class="submit"><input type="submit" value="Submit"></div>
                </div>
        </div>
</asp:Content>
```

Output:





c) Create a web application to demonstrate various states of ASP.NET Pages:

i) View State:

View State is the method to preserve the Value of the Page and Controls between round trips. It is a Page-Level State Management technique. View State is turned on by default and normally serialises the data in every control on the page regardless of whether it is actually used during a post-back.

Source code:

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac_5b.WebForm1" %>
<!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="Label3" runat="server" Text="View State Example:"></asp:Label>
            <br />
            <table>
                <tr>
                    <td>
                        <asp:Label ID="Label1" runat="server" Text="User Name: "></asp:Label>
                    </td>
                    <td>
                        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                    </td>
                </tr>
                <tr>
                    <td>
                        <asp:Label ID="Label2" runat="server" Text="Password:"></asp:Label>
                    </td>
                    <td>
                        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                    </td>
                </tr>
                <tr>
                    <td>
                        <asp:Button ID="Button1" runat="server"
OnClick="Button1_Click" Text="Submit" />
                    </td>
                    <td>
                        <asp:Button ID="Button2" runat="server"
OnClick="Button2_Click" Text="Restore" />
                    </td>
                </tr>
            </table>
        </div>
    </form>
</body>
</html>
```

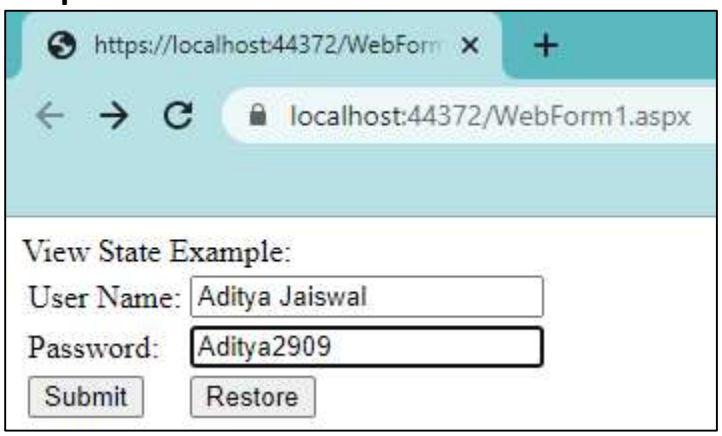
WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Prac_5b
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        public string a, b;
        protected void Button1_Click(object sender, EventArgs e)
        {
            ViewState["name"] = TextBox1.Text;
            ViewState["password"] = TextBox2.Text;
            TextBox1.Text = TextBox2.Text = string.Empty;
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
```

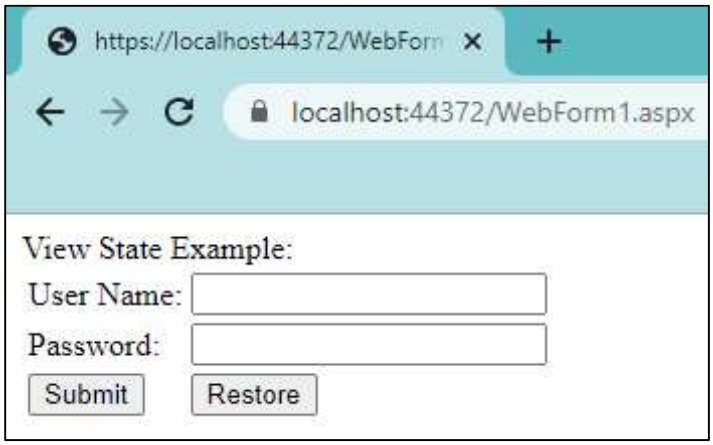
```
            if (ViewState["name"] != null)
            {
                TextBox1.Text = ViewState["name"].ToString();
            }
            if (ViewState["password"] != null)
            {
                TextBox2.Text = ViewState["password"].ToString();
            }
        }
        protected void Page_Load(object sender, EventArgs e)
        {
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Prac_5b
{
    2 references
    public partial class WebForm1 : System.Web.UI.Page
    {
        public string a, b;
        0 references
        protected void Button1_Click(object sender, EventArgs e)
        {
            ViewState["name"] = TextBox1.Text;
            ViewState["password"] = TextBox2.Text;
            TextBox1.Text = TextBox2.Text = string.Empty;
        }

        0 references
        protected void Button2_Click(object sender, EventArgs e)
        {
            if (ViewState["name"] != null)
            {
                TextBox1.Text = ViewState["name"].ToString();
            }
            if (ViewState["password"] != null)
            {
                TextBox2.Text = ViewState["password"].ToString();
            }
        }
        0 references
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

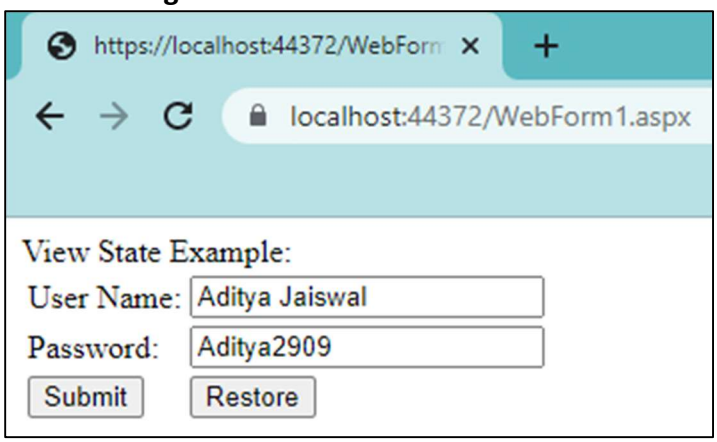
Output:



After Clicking on Submit Button:

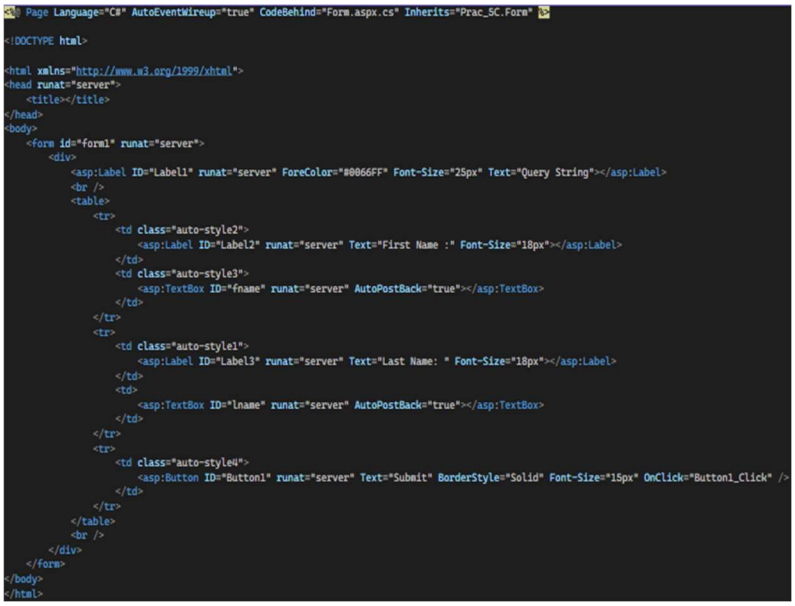


After Clicking on Restore Button:



ii) Query Strings:

A query string is one of the techniques in Web applications to send data from one webform to another through the URL. A query string consists of two parts, field and value, and each of pair separated by ampersand (&).



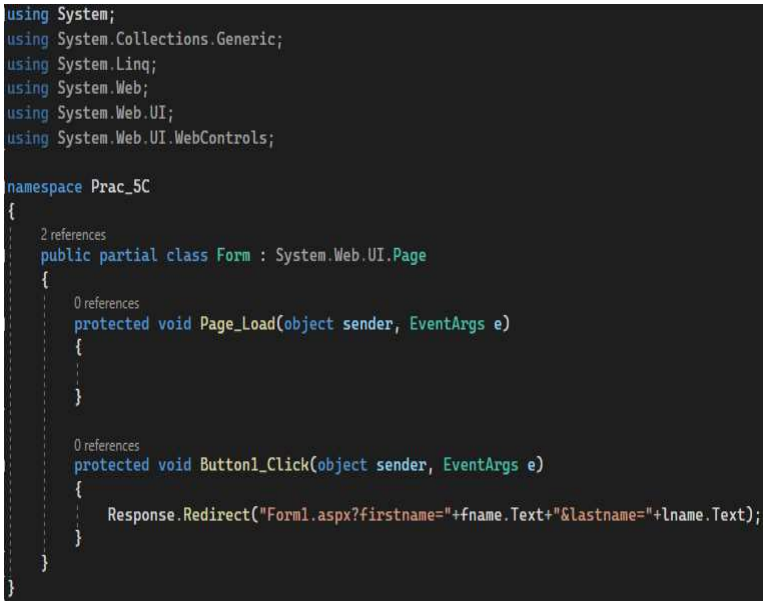
Source code:

Form.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Form.aspx.cs" Inherits="Prac_5C.Form" %>
<!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"
ForeColor="#0066FF" Font-Size="25px" Text="Query
String"></asp:Label>
            <br />
            <table>
                <tr>
                    <td class="auto-style2">
                        <asp:Label ID="Label2" runat="server" Text="First Name :" Font-Size="18px"></asp:Label>
                    </td>
                    <td class="auto-style3">
                        <asp:TextBox ID="fname" runat="server" AutoPostBack="true"></asp:TextBox>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style1">
                        <asp:Label ID="Label3" runat="server" Text="Last Name: " Font-Size="18px"></asp:Label>
                    </td>
                    <td>
                        <asp:TextBox ID="lname" runat="server" AutoPostBack="true"></asp:TextBox>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style4">
                        <asp:Button ID="Button1" runat="server" Text="Submit" BorderStyle="Solid" Font-Size="15px"
OnClick="Button1_Click" />
                    </td>
                </tr>
            </table>
        </div>
    </form>
</body>
</html>
```

Form.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Prac_5C
{
    public partial class Form : System.Web.UI.Page
    {
        protected void Page_Load(object sender,
EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
```



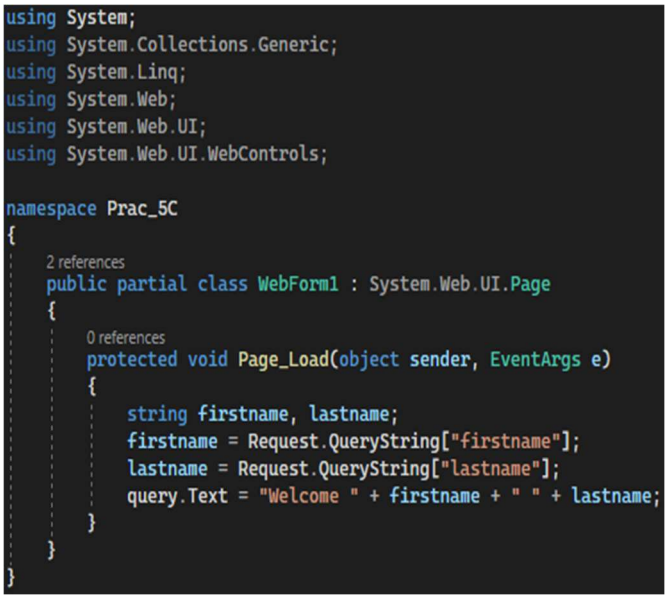
```
            Response.Redirect("Form1.aspx?firstname="+fname.Text+"&lastname="+lname.Text);
        }
    }
}
```

Form1.aspx code:

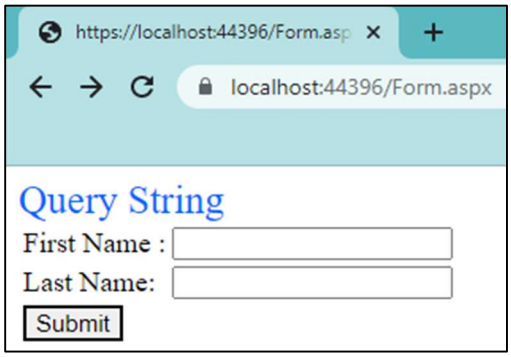
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Form1.aspx.cs"
Inherits="Prac_5C.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" ForeColor="#FF3300" Font-Size="30px"
Text="Query String"></asp:Label>
            <br />
            <asp:Label ID="query" runat="server" Font-Size="20px"></asp:Label>
        </div>
    </form>
</body>
</html>
```

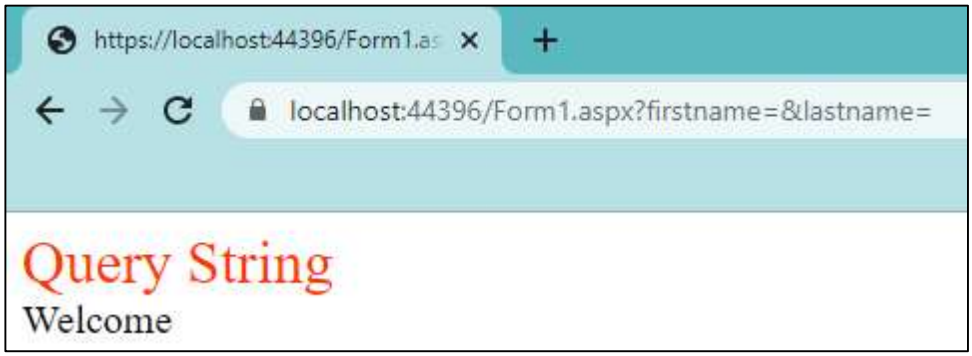
Form1.aspx.cs code:

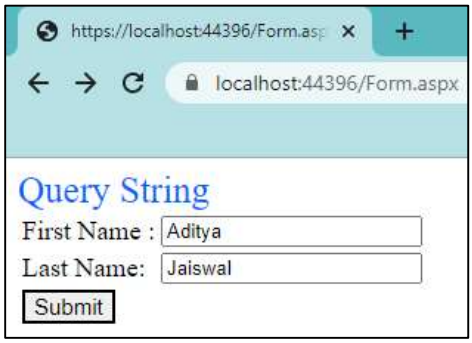
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Prac_5C
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender,
EventArgs e)
        {
            string firstname, lastname;
            firstname =
Request.QueryString["firstname"];
            lastname =
Request.QueryString["lastname"];
            query.Text = "Welcome " + firstname + " " + lastname;
        }
    }
}
```

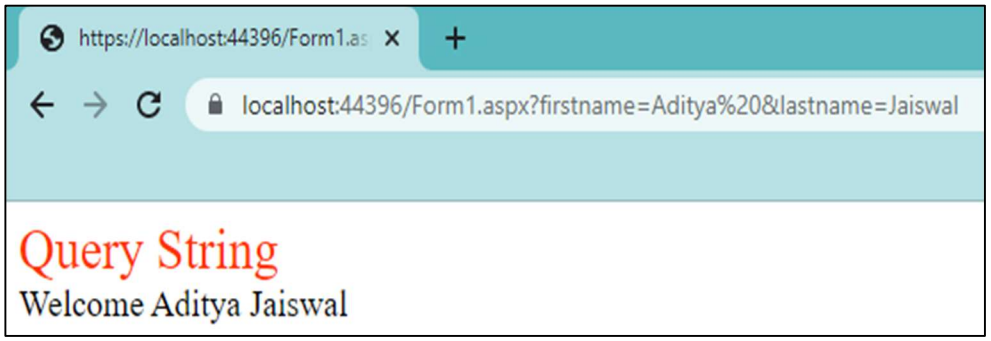


Output:

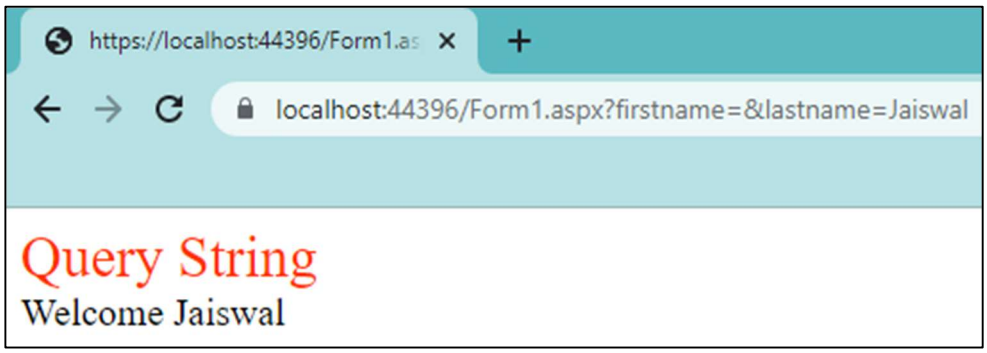

https://localhost:44396/Form.asp
localhost:44396/Form.aspx
Query String
First Name :
Last Name:
Submit


https://localhost:44396/Form1.as
localhost:44396/Form1.aspx?firstname=&lastname=
Query String
Welcome


https://localhost:44396/Form.asp
localhost:44396/Form.aspx
Query String
First Name : Aditya
Last Name: Jaiswal
Submit


https://localhost:44396/Form1.as
localhost:44396/Form1.aspx?firstname=Aditya%20&lastname=Jaiswal
Query String
Welcome Aditya Jaiswal


https://localhost:44396/Form.asp
localhost:44396/Form.aspx
Query String
First Name :
Last Name: Jaiswal
Submit


https://localhost:44396/Form1.as
localhost:44396/Form1.aspx?firstname=&lastname=Jaiswal
Query String
Welcome Jaiswal

Practical 6: Working with Database

a) Create a web application bind data in a multiline textbox by querying in another textbox.

Source code:

WebForm1.aspx code:

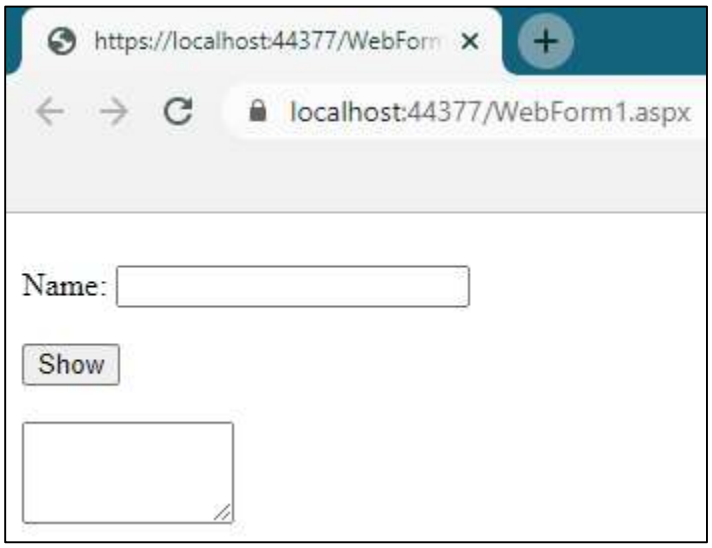
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac_6A.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <br />
            <asp:Label ID="name" runat="server" Text="Name:"></asp:Label>
            <asp:TextBox ID="value" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Show" OnClick="Button1_Click" />
            <br />
            <br />
            <asp:TextBox ID="Output" runat="server" TextMode="MultiLine" Height="45px" Width="100px"></asp:TextBox>
            <br />
            <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>" SelectCommand="SELECT * FROM [employee] WHERE ([Name] = @Name)">
                <SelectParameters>
                    <asp:ControlParameter ControlID="value" Name="Name" PropertyName="Text" Type="String" />
                </SelectParameters>
            </asp:SqlDataSource>
            <br />
            <br />

        </div>
    </form>
</body>
</html>
```

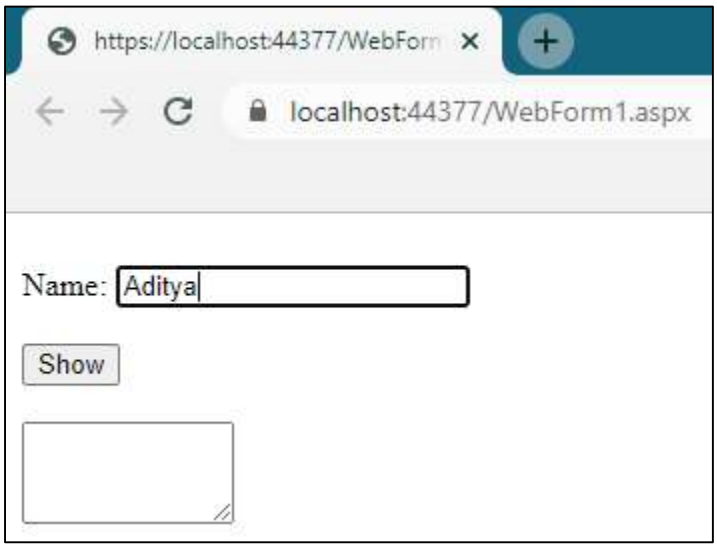
WebForm1.aspx.cs code

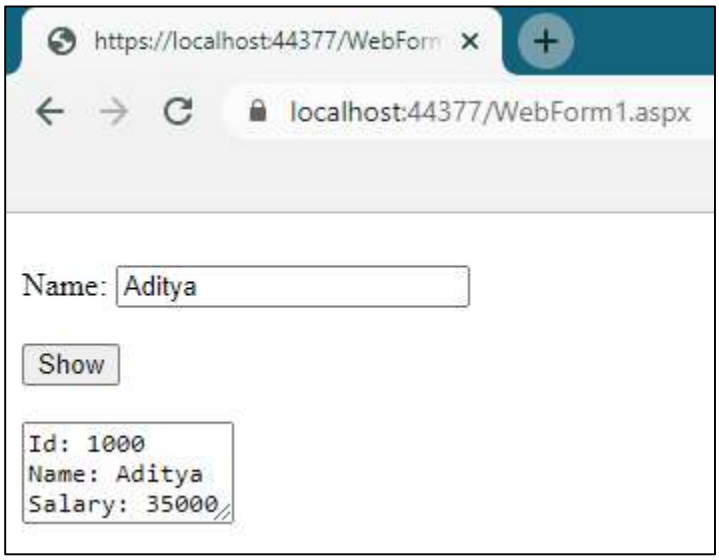
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.Sql;
using System.Data.SqlClient;

namespace Prac_6A
{
    public partial class WebForm1 :
System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=Company;Integrated Security=SSPI");
            con.Open();
            string str = "SELECT * FROM [employee] WHERE Name='" + value.Text + "'";
            SqlCommand cmd = new SqlCommand(str,con);
            SqlDataReader rdr = cmd.ExecuteReader();
            while (rdr.Read())
            {
                Output.Text = "Id: " + rdr[0].ToString() + "\nName: " + rdr[1].ToString() + "\nSalary: " +
rdr[2].ToString();
            }

        }
    }
```

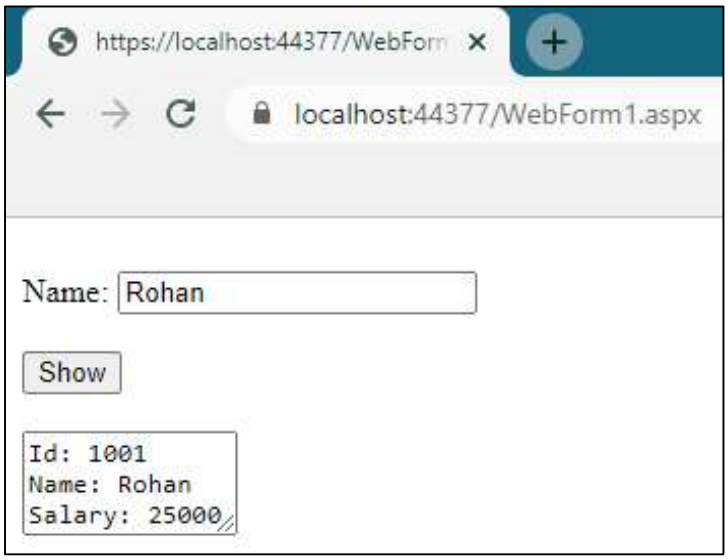
}

Output:


https://localhost:44377/WebForm
localhost:44377/WebForm1.aspx
Name:
Show


https://localhost:44377/WebForm
localhost:44377/WebForm1.aspx
Name: Aditya
Show

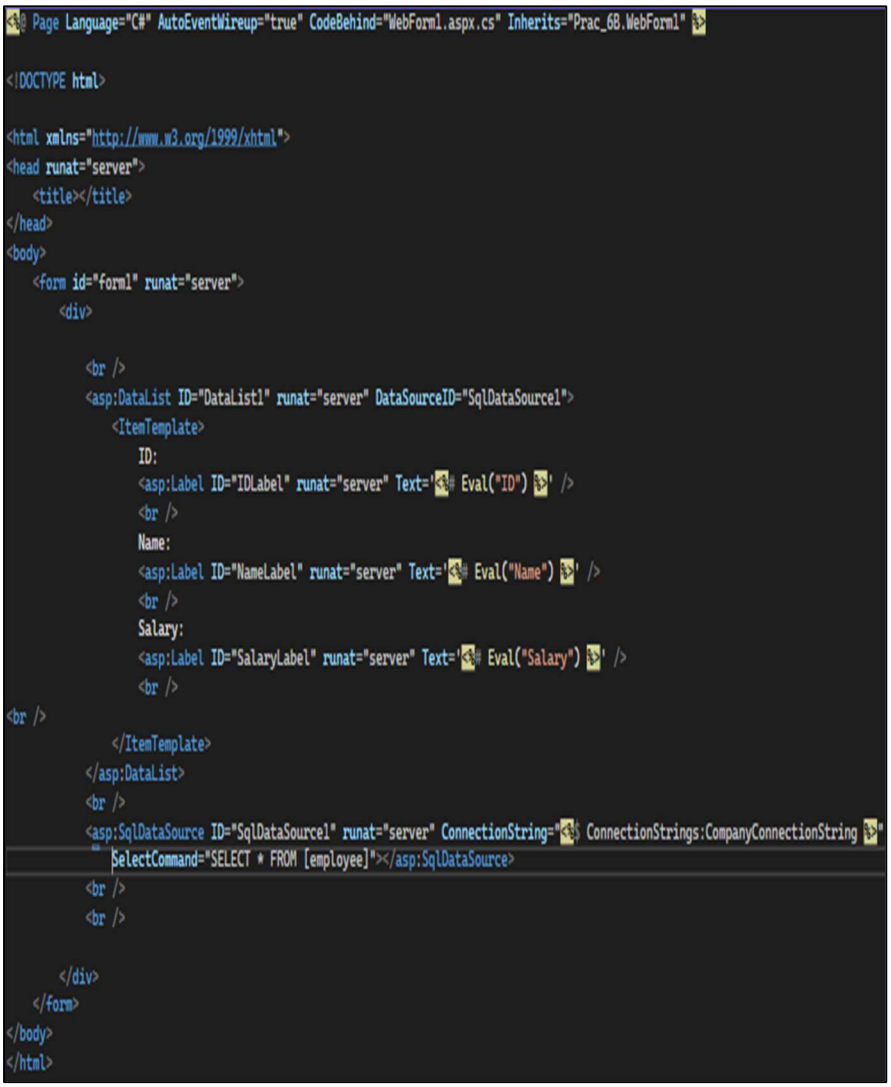

https://localhost:44377/WebForm
localhost:44377/WebForm1.aspx
Name: Aditya
Show
Id: 1000
Name: Aditya
Salary: 35000


https://localhost:44377/WebForm
localhost:44377/WebForm1.aspx
Name: Rohan
Show
Id: 1001
Name: Rohan
Salary: 25000

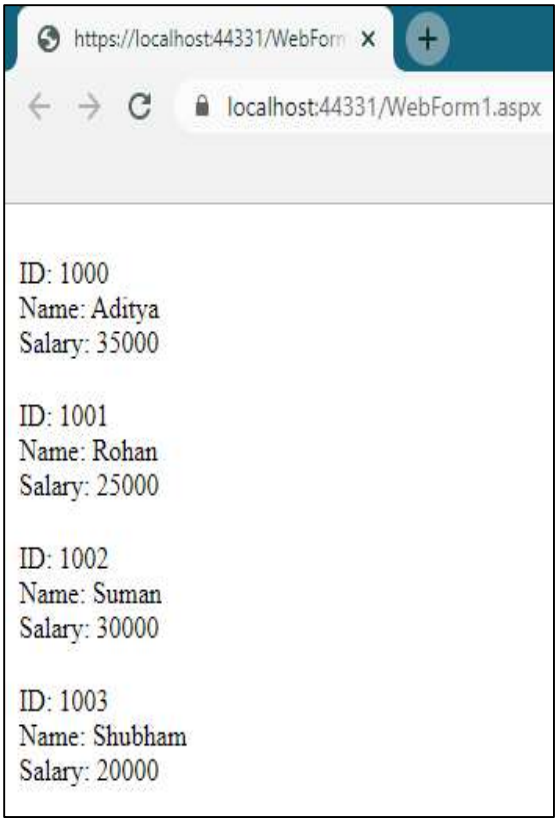
b) Demonstrate the use of Data list link control.

Source code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac_6B.WebForm1"
%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <br />
            <asp:DataList ID="DataList1" runat="server" DataSourceID="SqlDataSource1">
                <ItemTemplate>
                    ID:
                    <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>' />
                    <br />
                    Name:
                    <asp:Label ID="NameLabel" runat="server" Text='<%# Eval("Name") %>' />
                    <br />
                    Salary:
                    <asp:Label ID="SalaryLabel" runat="server" Text='<%# Eval("Salary") %>' />
                    <br />
<br />
                </ItemTemplate>
            </asp:DataList>
            <br />
            <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>" SelectCommand="SELECT * FROM
[employee]"></asp:SqlDataSource>
            <br />
            <br />
        </div>
    </form>
</body>
</html>
```



Output:



Practical 7: Working with Database

a) Create a web application to display Databinding using dropdown list control.

Data binding is the process that establishes a connection between the UI and the data it displays. If the binding has the correct settings and the data provides the proper notifications, when the data changes its value, the elements that are bound to the data reflect changes automatically.

Every ASP.NET web form control inherits the DataBind method from its parent Control class, which gives it an inherent capability to bind data to at least one of its properties. This is known as simple data binding or inline data binding.

Simple data binding involves attaching any collection (item collection) which implements the IEnumerable interface, or the DataSet and DataTable classes to the DataSource property of the control.

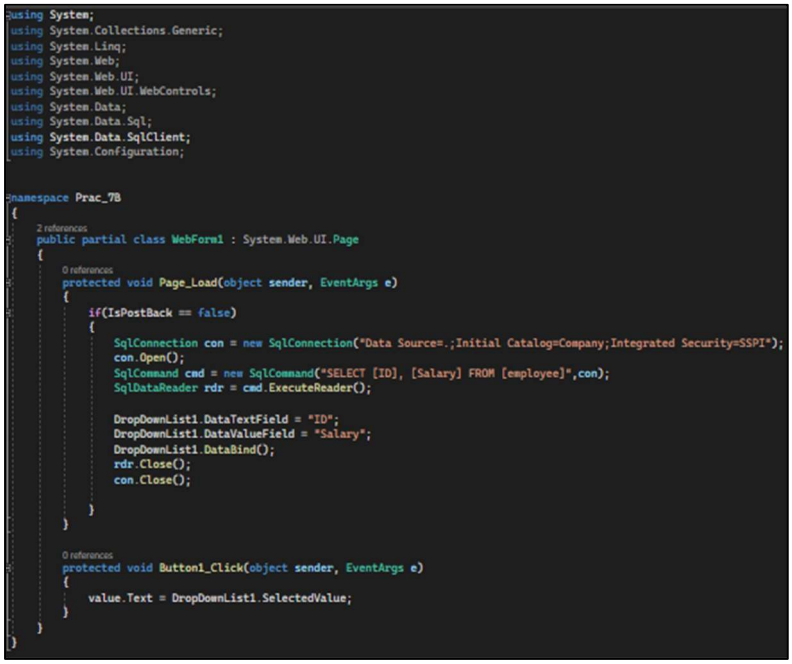
Source code:

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac_7B.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>" SelectCommand="SELECT [ID], [Salary] FROM
[employee]"></asp:SqlDataSource>
            <br />
            <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True" DataSourceID="SqlDataSource1"
DataTextField="ID" DataValueField="Salary">
            </asp:DropDownList>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Click" />
            <br />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Your Salary is:  "></asp:Label>
            <asp:Label ID="value" runat="server"></asp:Label>
        </div>
    </form>
</body>
</html>
```

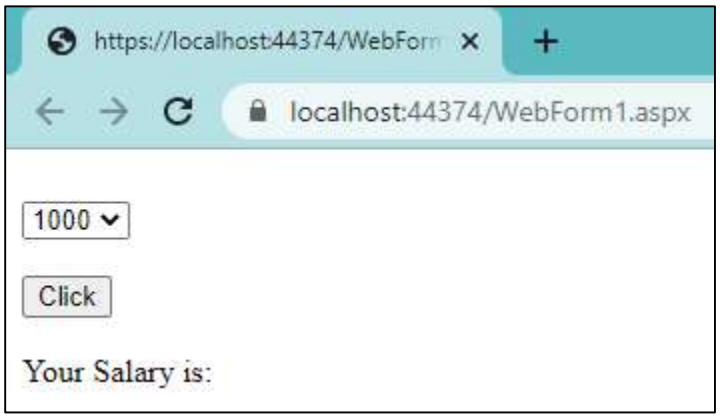
WebForm1.aspx.cs code:

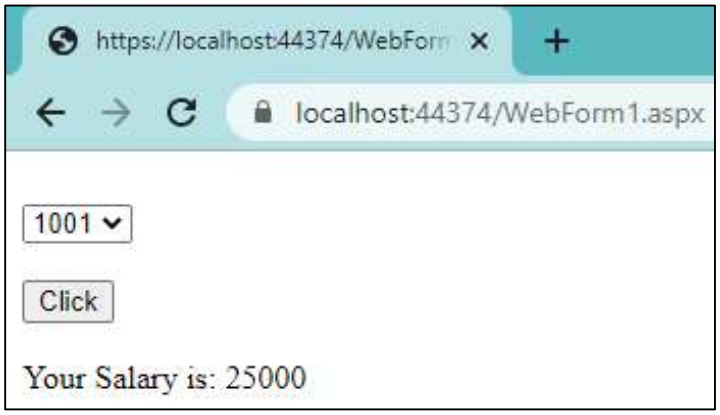
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
using System.Configuration;
namespace Prac_7B
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender,
EventArgs e)
        {
            if(IsPostBack == false)
            {
                SqlConnection con = new
SqlConnection("Data Source=.;Initial
Catalog=Company;Integrated Security=SSPI");
                con.Open();
                SqlCommand cmd = new SqlCommand("SELECT [ID], [Salary] FROM [employee]",con);
                SqlDataReader rdr = cmd.ExecuteReader();
                DropDownList1.DataTextField = "ID";
                DropDownList1.DataValueField = "Salary";
                DropDownList1.DataBind();
                rdr.Close();
                con.Close();
            }
        }
        protected void Button1_Click(object sender, EventArgs e)
```

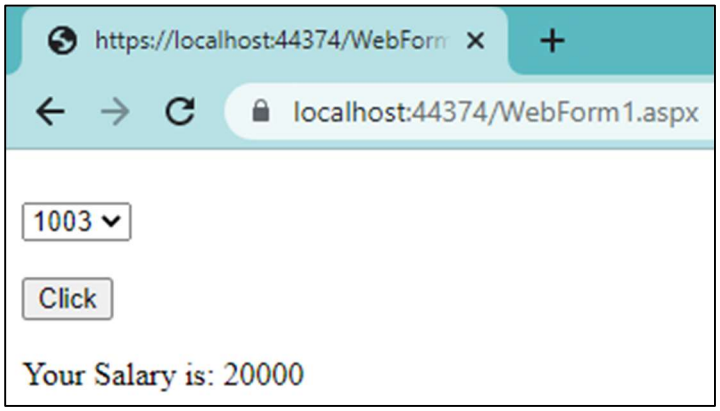


```
        {
            value.Text = DropDownList1.SelectedValue;
        }
    }
}
```

Output:







b) Create a web application for to display the phone no of an author using database.

Source code:

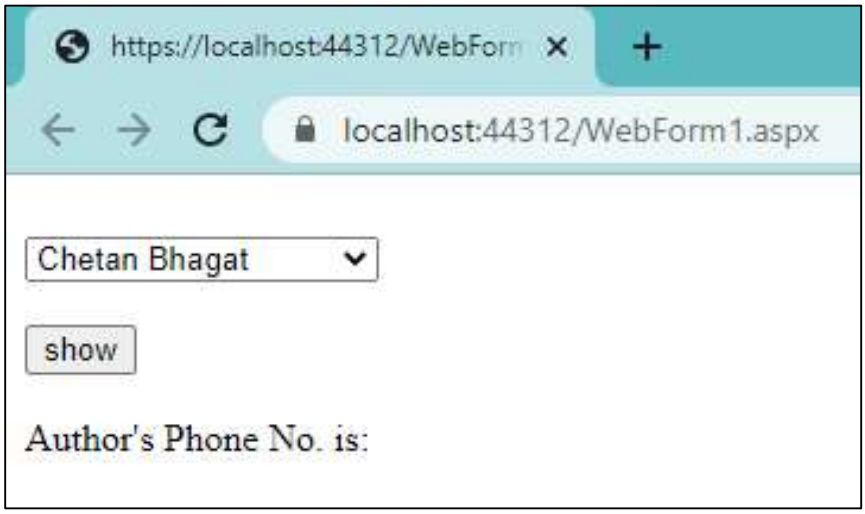
WebForm1.aspx code:

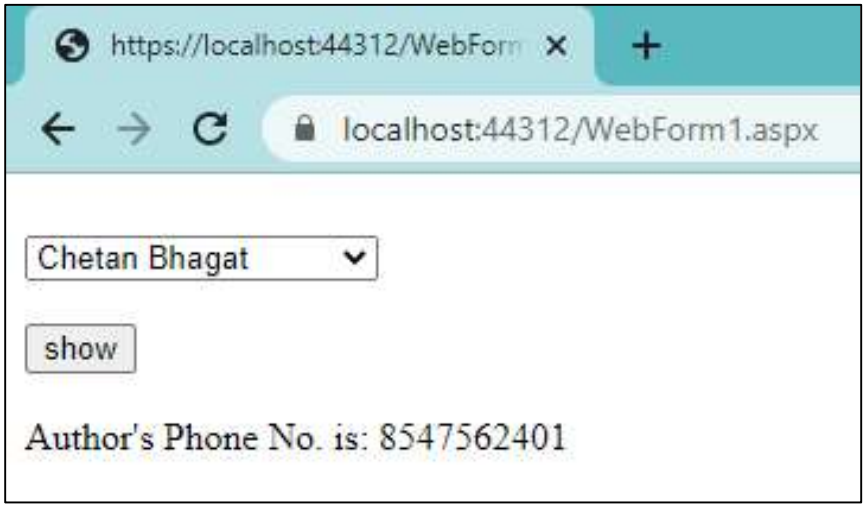
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac7_B.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <br />
            <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True" DataSourceID="SqlDataSource1"
                DataTextField="Author" DataValueField="Phone">
            </asp:DropDownList>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="show" />
            <br />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Author's Phone No. is: "></asp:Label>
            <asp:Label ID="phone" runat="server"></asp:Label>
            <br />
            <br />
            <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>"
                SelectCommand="SELECT [Author], [Phone] FROM [Authors]"></asp:SqlDataSource>
        </div>
    </form>
</body>
</html>
```

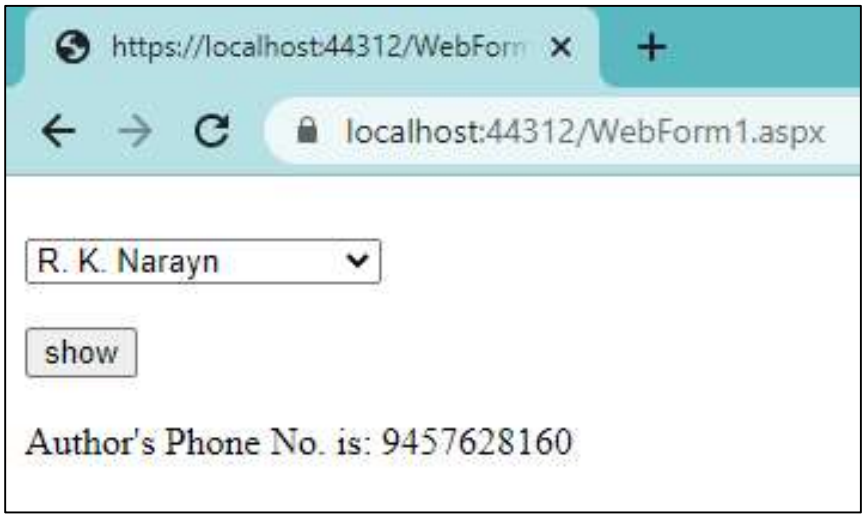
WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.Sql;
using System.Data.SqlClient;


namespace Prac7_B
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if(IsPostBack == false)
            {
                SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=Company;Integrated
Security=SSPI");
                con.Open();
                SqlCommand cmd = new SqlCommand("SELECT [Author], [Phone] FROM [Authors]", con);
                SqlDataReader rdr = cmd.ExecuteReader();
                DropDownList1.DataTextField = "Author";
                DropDownList1.DataValueField = "Phone";
                DropDownList1.DataBind();
                rdr.Close();
                con.Close();
            }
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            phone.Text = DropDownList1.SelectedValue;
        }
    }
}
```

Output:


https://localhost:44312/WebForm
localhost:44312/WebForm1.aspx
Chetan Bhagat
show
Author's Phone No. is:


https://localhost:44312/WebForm
localhost:44312/WebForm1.aspx
Chetan Bhagat
show
Author's Phone No. is: 8547562401


https://localhost:44312/WebForm
localhost:44312/WebForm1.aspx
R. K. Narayn
show
Author's Phone No. is: 9457628160

c) Create a web application for inserting and deleting record from a database. (Using Execute-Non Query).

The ExecuteNonQuery method is used to execute SQL Command or the stored procedure performs INSERT, UPDATE or Delete operations. It doesn't return any data from the database. Instead, it returns an integer specifying the number of rows inserted, updated or deleted.

Source code:

WebForm1.aspx code:

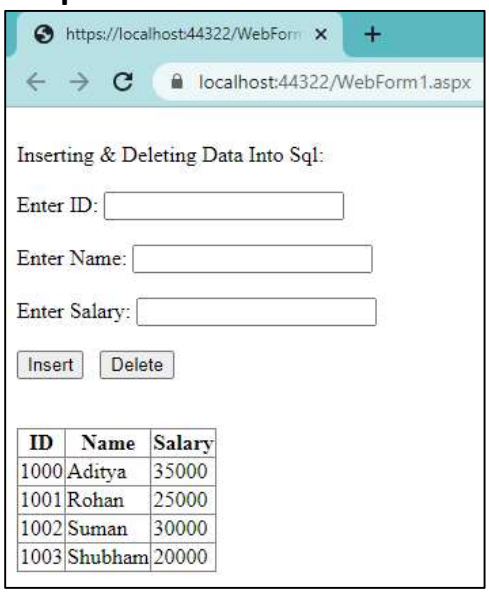
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac_7_C.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>" SelectCommand="SELECT * FROM [employee]"></asp:SqlDataSource>
            <br />
            <asp:Label ID="Label1" runat="server" Text="Inserting &amp;amp; Deleting Data Into Sql:"></asp:Label>
            <br />
            <br />
            <asp:Label ID="Label2" runat="server" Text="Enter ID: "></asp:Label>
            <asp:TextBox ID="id" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Label ID="Label3" runat="server" Text="Enter Name:  "></asp:Label>
            <asp:TextBox ID="name" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Label ID="Label4" runat="server" Text="Enter Salary:  "></asp:Label>
            <asp:TextBox ID="salary" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Button ID="Insert" runat="server" OnClick="Insert_Click" Text="Insert" />
            <asp:Button ID="Delete" runat="server" OnClick="Delete_Click" Text="Delete" />
            <br />
            <br />
            <asp:Label ID="value" runat="server"></asp:Label>
            <br />
            <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataSourceID="SqlDataSource1">
                <Columns>
                    <asp:BoundField DataField="ID" HeaderText="ID" SortExpression="ID" />
                    <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name" />
                    <asp:BoundField DataField="Salary" HeaderText="Salary" SortExpression="Salary" />
                </Columns>
            </asp:GridView>
            <br />
        </div>
    </form>
</body>
</html>
```

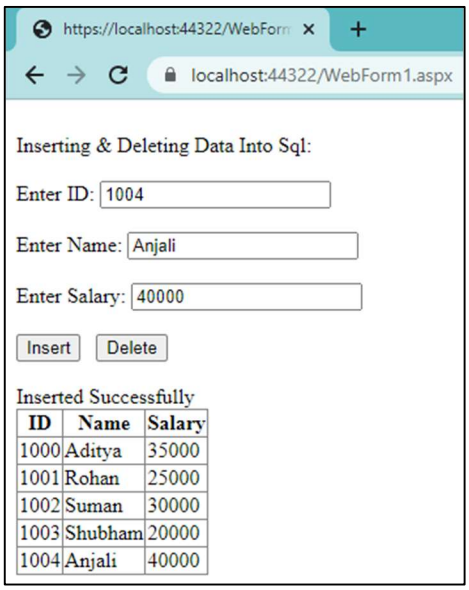
WebForm1.aspx.cs code:

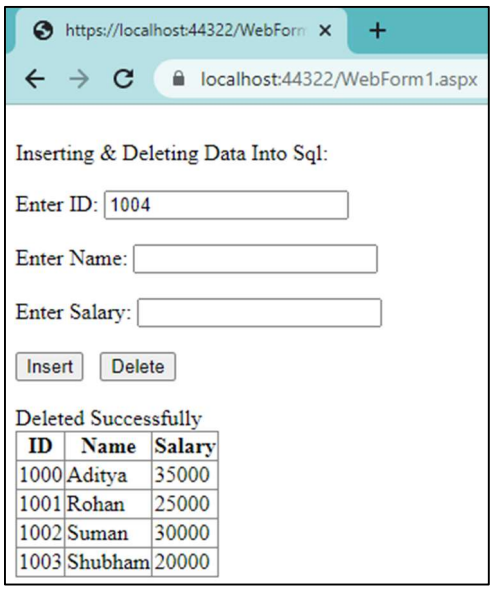
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
using System.Configuration;
namespace Prac_7_C
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Insert_Click(object sender, EventArgs e)
        {
```

```
            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=Company;Integrated Security=SSPI");
            con.Open();
            String str;
            str = "INSERT INTO [employee] (ID,Name,Salary) VALUES (" + id.Text + ",'" + name.Text + "'," + salary.Text
+ ")";
            SqlCommand cmd = new SqlCommand(str, con);
            cmd.CommandType = System.Data.CommandType.Text;
            cmd.ExecuteNonQuery();
            value.Text = "Inserted Successfully";
            SqlCommand cmd2 = new SqlCommand("SELECT * FROM employee",con);
            SqlDataReader rdr = cmd2.ExecuteReader();
            GridView1.DataBind();
            rdr.Close();
            con.Close();
        }
        protected void Delete_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=Company;Integrated Security=SSPI");
            con.Open();
            String str;
            str = "DELETE FROM employee WHERE ID=" + id.Text + "";
            SqlCommand cmd1 = new SqlCommand(str, con);
            SqlDataReader rdr2 = cmd1.ExecuteReader();
            value.Text = "Deleted Successfully";
            rdr2.Close();
            SqlCommand cmd2 = new SqlCommand("SELECT * FROM employee",con);
            SqlDataReader rdr = cmd2.ExecuteReader();
            GridView1.DataBind();
            rdr.Close();
            con.Close();
        }
    }
}
```

Output:







Practical 8: Working with data controls

a) Create a web application to demonstrate data binding using DetailsView and FormView Control.

i) DetailsView:

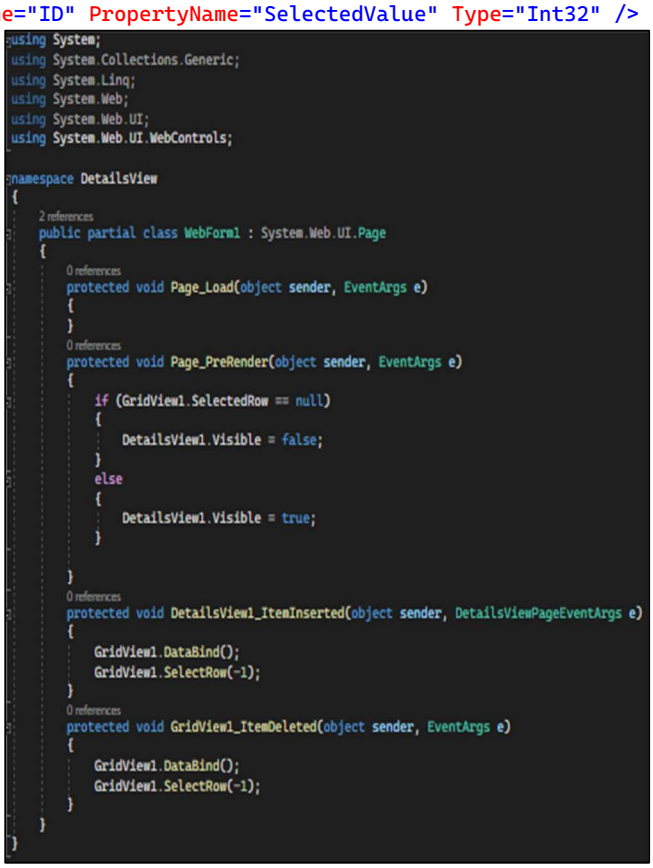
Source code:

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="DetailsView.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:GridView ID="GridView1" DataKeyNames="ID" runat="server" AutoGenerateColumns="False"
DataSourceID="SqlDataSource1">
                <Columns>
                    <asp:CommandField ShowSelectButton="True" />
                    <asp:BoundField DataField="ID" HeaderText="ID" SortExpression="ID" ReadOnly="True" />
                </Columns>
            </asp:GridView>
            <br />
            <br />
            <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>" SelectCommand="SELECT [ID] FROM [employee]"></asp:SqlDataSource>
            <br />
            <asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="125px" AutoGenerateRows="False"
DataKeyNames="ID" DataSourceID="SqlDataSource2">
                <Fields>
                    <asp:BoundField DataField="ID" HeaderText="ID" ReadOnly="True" SortExpression="ID" />
                    <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name" />
                    <asp:BoundField DataField="Salary" HeaderText="Salary" SortExpression="Salary" />
                </Fields>
            </asp:DetailsView>
            <br />
            <asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>" SelectCommand="SELECT * FROM [employee] WHERE ([ID] = @ID)">
                <SelectParameters>
                    <asp:ControlParameter ControlID="GridView1" Name="ID" PropertyName="SelectedValue" Type="Int32" />
                </SelectParameters>
            </asp:SqlDataSource>
        </div>
    </form>
</body>
</html>
```

WebForm1.aspx.cs code:

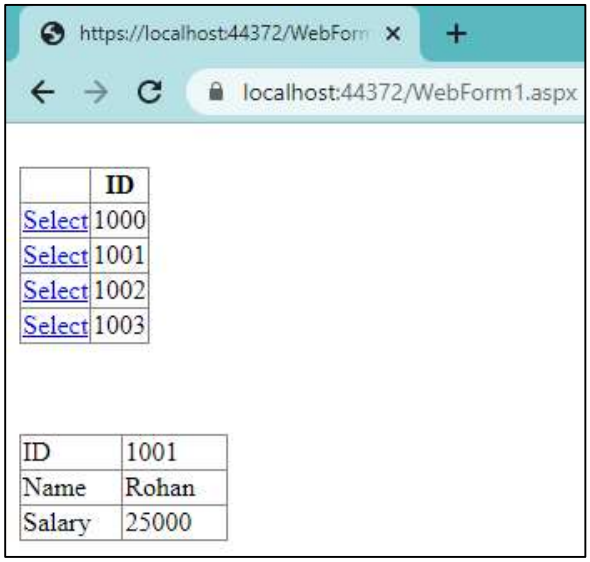
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace DetailsView
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Page_PreRender(object sender, EventArgs e)
        {
            if (GridView1.SelectedRow == null)
            {
                DetailsView1.Visible = false;
            }
            else
            {
                DetailsView1.Visible = true;
            }
        }
        protected void DetailsView1_ItemInserted(object sender, DetailsViewPageEventArgs e)
```



```
        {
            GridView1.DataBind();
            GridView1.SelectRow(-1);
        }
        protected void GridView1_ItemDeleted(object sender, EventArgs e)
        {
            GridView1.DataBind();
            GridView1.SelectRow(-1);
        }
    }
}
```

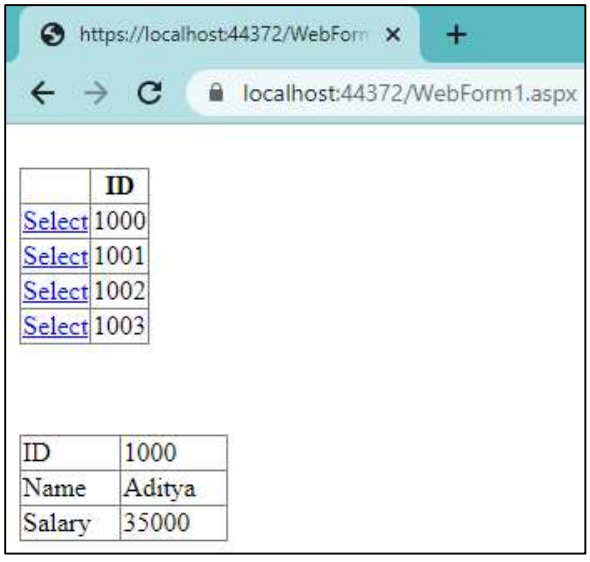
Output:

	ID
Select	1000
Select	1001
Select	1002
Select	1003



	ID
Select	1000
Select	1001
Select	1002
Select	1003

ID	1001
Name	Rohan
Salary	25000



	ID
Select	1000
Select	1001
Select	1002
Select	1003

ID	1000
Name	Aditya
Salary	35000

ii) FormView:

Source code:

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="FormView.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server" align="center">
        <div>
            <asp:FormView ID="FormView1" runat="server" BorderColor="Black" BorderStyle="Groove"
BorderWidth="1.5pt" Font-Italic="True" Font-Names="Pristina" Font-Size="20pt" DataKeyNames="ID"
DataSourceID="SqlDataSource1" AllowPaging="True">
                <FooterStyle BackColor="#66FF99" Font-Names="Sylfaen" />
                <FooterTemplate>
                    <asp:Button ID="Button1" runat="server" BackColor="#CC99FF"
BorderColor="#66CCFF" BorderStyle="Solid" Font-Names="Harrington" Font-Size="12pt" Font-
Strikeout="False" Text="Button" />
                    <br />
                    Thank You...!!!
                </FooterTemplate>
                <HeaderStyle BackColor="#66FFFF" Font-Names="Algerian" Font-Size="24pt" Font-
Underline="True" />
                <HeaderTemplate>
                    Product Details
                </HeaderTemplate>
                <ItemTemplate>
                    ID:
                    <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>'></asp:Label>
                    <br />
                    Name:
                    <asp:Label ID="NameLabel" runat="server" Text='<%# Bind("Name") %>'></asp:Label>
                    <br />
                    Salary:
                    <asp:Label ID="SalaryLabel" runat="server" Text='<%# Bind("Salary")
%>'></asp:Label>
                    <br />
                </ItemTemplate>
                <PagerTemplate>
                    Thank You..!!!
                </PagerTemplate>
            </asp:FormView>
            <br />
            <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>" SelectCommand="SELECT * FROM
[employee]"></asp:SqlDataSource>
            <br />
        </div>
    </form>
</body>
</html>
```


WebForm1.aspx
FormView: Overview
div
PRODUCT DETAILS
ID: 0
Name: abc
Salary: 0
Button
Thank You...!!!
Thank You..!!!
SqlDataSource - SqlDataSource1

Output:


https://localhost:44326/WebForm
localhost:44326/WebForm1.aspx
PRODUCT DETAILS
ID: 1000
Name: Aditya
Salary: 35000
Button
Thank You...!!!
Thank You..!!!

b) Create a web application to display Using Disconnected Data Access and Data binding using GridView.

Source code:

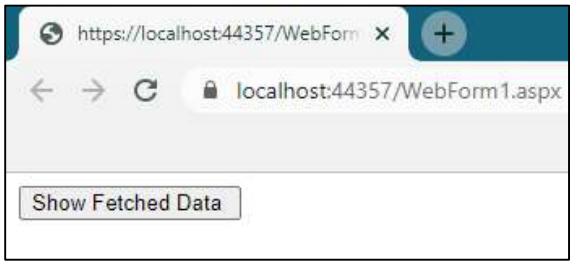
WebForm1.aspx code:

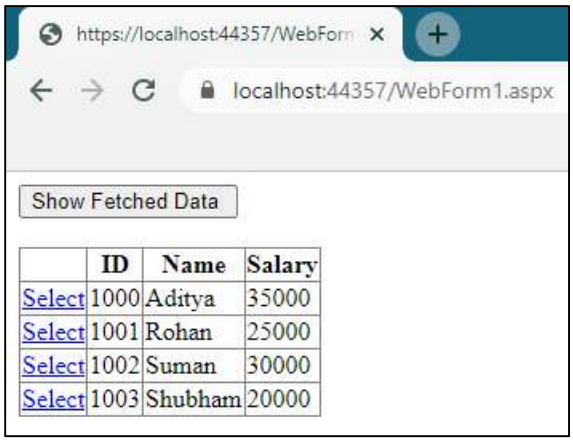
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac_8C.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Show Fetched Data " />
            <br />
            <br />
            <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="ID">
                <Columns>
                    <asp:CommandField ShowSelectButton="True" />
                    <asp:BoundField DataField="ID" HeaderText="ID" ReadOnly="True" SortExpression="ID" />
                    <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name" />
                    <asp:BoundField DataField="Salary" HeaderText="Salary" SortExpression="Salary" />
                </Columns>
            </asp:GridView>
            <br />
        <!--    <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:CompanyConnectionString %>" SelectCommand="SELECT * FROM [employee]"></asp:SqlDataSource>-->
            <br />
        </div>
    </form>
</body>
</html>
```

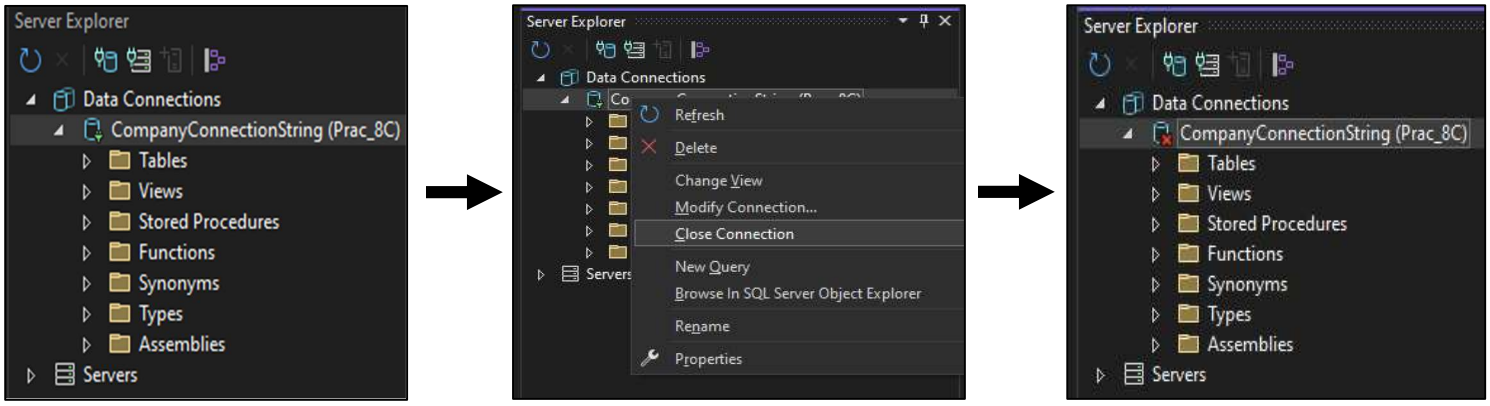
WebForm1.aspx.cs code:

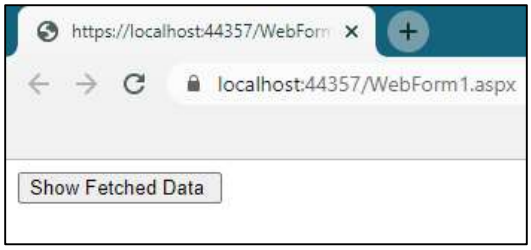
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.Sql;
using System.Data.SqlClient;
using System.Configuration;
using System.Data;

namespace Ajax_DisconnectedDataAccess
{
    public partial class Disconnected_DataAccess : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection("Data Source=.;Initial Catalog=Customer;Integrated Security=SSPI");
            GridView1.Visible = true;
            con.Open();
            string query = "SELECT * FROM [tblproduct]";
            SqlDataAdapter sda = new SqlDataAdapter(query,con);
            DataSet ds = new DataSet();
            sda.Fill(ds,"Product");
            GridView1.DataSource = ds;
            GridView1.DataBind();
            con.Close();

        }
    }
}
```

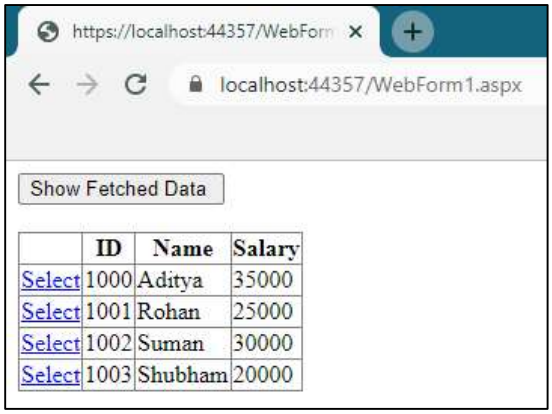
Output:


https://localhost:44357/WebForm
localhost:44357/WebForm1.aspx
Show Fetched Data


https://localhost:44357/WebForm
localhost:44357/WebForm1.aspx
Show Fetched Data
ID Name Salary
Select 1000 Aditya 35000
Select 1001 Rohan 25000
Select 1002 Suman 30000
Select 1003 Shubham 20000


Server Explorer
Data Connections
CompanyConnectionString (Prac_8C)
Tables
Views
Stored Procedures
Functions
Synonyms
Types
Assemblies
Servers
Refresh
Delete
Change View
Modify Connection...
Close Connection
New Query
Browse In SQL Server Object Explorer
Rename
Properties


https://localhost:44357/WebForm
localhost:44357/WebForm1.aspx
Show Fetched Data


https://localhost:44357/WebForm
localhost:44357/WebForm1.aspx
Show Fetched Data
ID Name Salary
Select 1000 Aditya 35000
Select 1001 Rohan 25000
Select 1002 Suman 30000
Select 1003 Shubham 20000

Practical 9: Working with GridView control

a) Create a web application to demonstrate use of GridView button column and GridView events.

Source code:

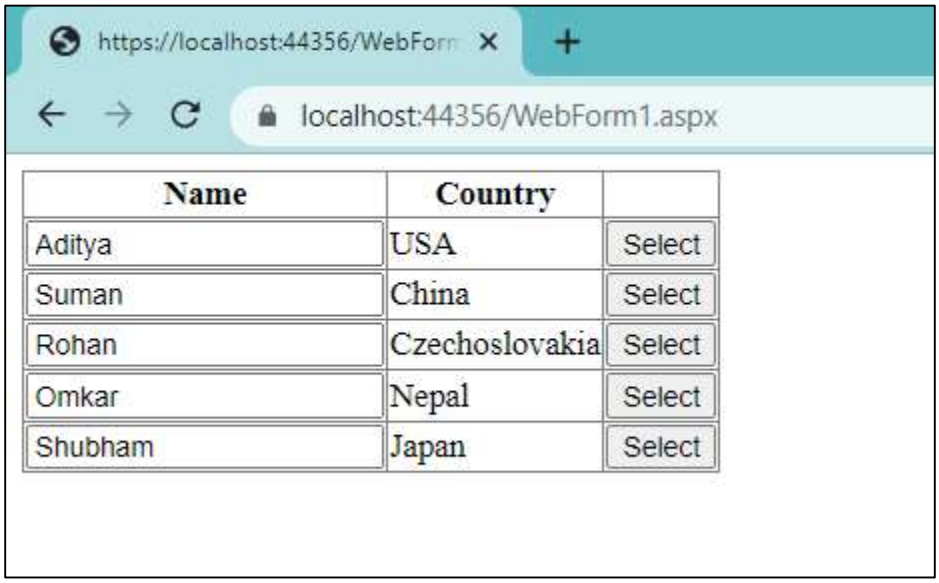
WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Grid_ViewPrax.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
OnRowCommand="GridView1_RowCommand">
                <Columns>
                    <asp:TemplateField HeaderText="Name" ItemStyle-Width="150">
                        <ItemTemplate>
                            <asp:TextBox ID="txtName" runat="server" Text='<%# Eval("Name") %>'></asp:TextBox>
                        </ItemTemplate>
                    </asp:TemplateField>
                    <asp:BoundField DataField="Country" HeaderText="Country" />
                    <asp:TemplateField>
                        <ItemTemplate>
                            <asp:Button ID="Button1" runat="server" Text="Select" CommandName="Select"
CommandArgument="<%# Container.DataItemIndex %>"/>
                        </ItemTemplate>
                    </asp:TemplateField>
                </Columns>
            </asp:GridView>
        </div>
    </form>
</body>
</html>
```

WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
namespace Grid_ViewPrax
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!this.IsPostBack) {
                DataTable dt = new DataTable();
                dt.Columns.AddRange(new DataColumn[2] { new DataColumn("Name"), new DataColumn("Country") });
                dt.Rows.Add("Aditya","USA");
                dt.Rows.Add("Suman", "China ");
                dt.Rows.Add("Rohan", "Czechoslovakia");
                dt.Rows.Add("Omkar", "Nepal");
                dt.Rows.Add("Shubham", "Japan");
                GridView1.DataSource = dt;
                GridView1.DataBind();
            }
        }
        protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
        {
            if(e.CommandName == "Select")
            {
                int rowIndex = Convert.ToInt32(e.CommandArgument);
                GridViewRow row = GridView1.Rows[rowIndex];
                string name = (row.FindControl("txtName") as TextBox).Text;
                string country = row.Cells[1].Text;
                ClientScript.RegisterStartupScript(this.GetType(),"alert","alert('Name: "+name+"\\n Country:
"+country+"')",true);
            }
        }
    }
}
```

Output:


https://localhost:44356/WebForm
localhost:44356/WebForm1.aspx
Name
Country
Aditya
USA
Select
Suman
China
Select
Rohan
Czechoslovakia
Select
Omkar
Nepal
Select
Shubham
Japan
Select


localhost:44356 says
Name: Aditya
Country: USA
OK

b) Create a web application to demonstrate GridView paging and Create own table format using GridView.

Source code:

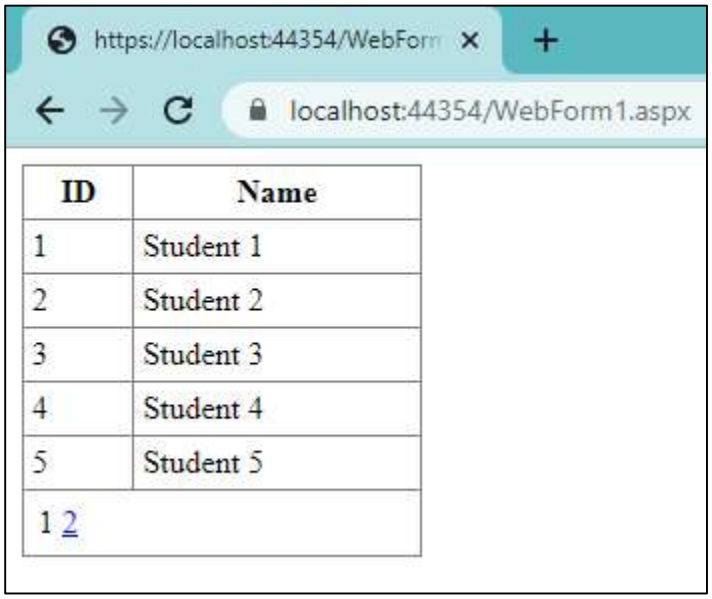
WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac9_B.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
                <asp:gridview ID="grdData" runat="server" AutoGenerateColumns="False" CellPadding="4" PageSize="5"
GridLines="both" Width="200" AllowPaging="True" OnPageIndexChanging="grdData_PageIndexChanging">
        <columns>
                <asp:boundfield DataField="ID" HeaderText="ID"></asp:boundfield>
                <asp:boundfield DataField="Name" HeaderText="Name"></asp:boundfield>
         </columns>
                </asp:gridview>
        </div>
        </form>
</body>
</html>
```

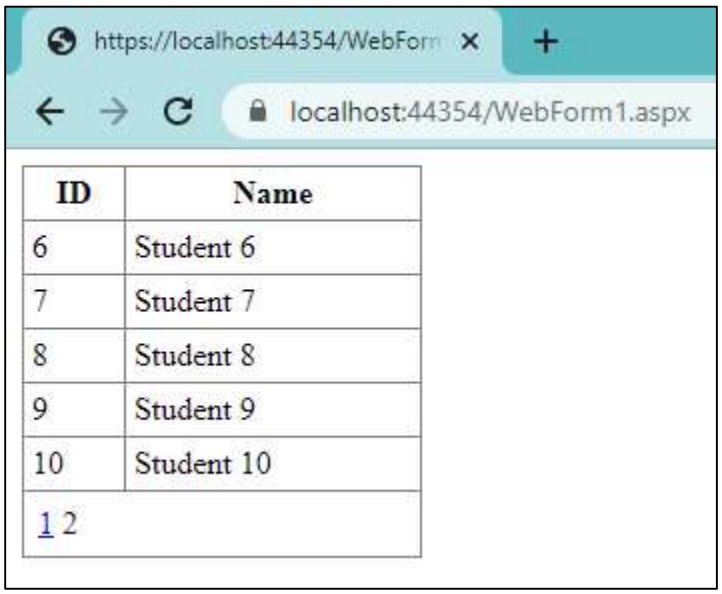
WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

namespace Prac9_B
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
                LoadGridData();
        }
        private void LoadGridData()
        {
            DataTable dt = new DataTable();
            dt.Columns.Add("Id");
            dt.Columns.Add("Name");
            for (int i = 0; i < 10; i++)
            {
                DataRow dr = dt.NewRow();
                dr["Id"] = i + 1;
                dr["Name"] = "Student " + (i + 1);
                dt.Rows.Add(dr);
            }
            grdData.DataSource = dt;
            grdData.DataBind();
        }
        protected void grdData_PageIndexChanging(object sender, GridViewPageEventArgs e)
        {
            grdData.PageIndex = e.NewPageIndex;
            LoadGridData();
        }
    }
}
```

Output:



ID	Name
1	Student 1
2	Student 2
3	Student 3
4	Student 4
5	Student 5
1 2	



ID	Name
6	Student 6
7	Student 7
8	Student 8
9	Student 9
10	Student 10
1 2	

c) Create a web application to demonstrate use of GridView control template and GridView hyperlink.

Source code:

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac_9C.WebForm1"
%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:GridView ID="GridView1" HeaderStyle-BackColor="#3AC0F2" HeaderStyle-ForeColor="White"
    RowStyle-BackColor="#A1DCF2" AlternatingRowStyle-BackColor="White" AlternatingRowStyle-
ForeColor="#000"
    runat="server" AutoGenerateColumns="false">
    <Columns>
        <asp:HyperLinkField DataTextField="Name" DataNavigateUrlFields="Id"
DataNavigateUrlFormatString="~/WebForm2.aspx?Id={0}"
            HeaderText="Name" ItemStyle-Width = "150" />
        <asp:BoundField DataField="Country" HeaderText="Country" ItemStyle-Width = "150" />
    </Columns>
</asp:GridView>
        </div>
    </form>
</body>
</html>
```

WebForm1.aspx.cs code:

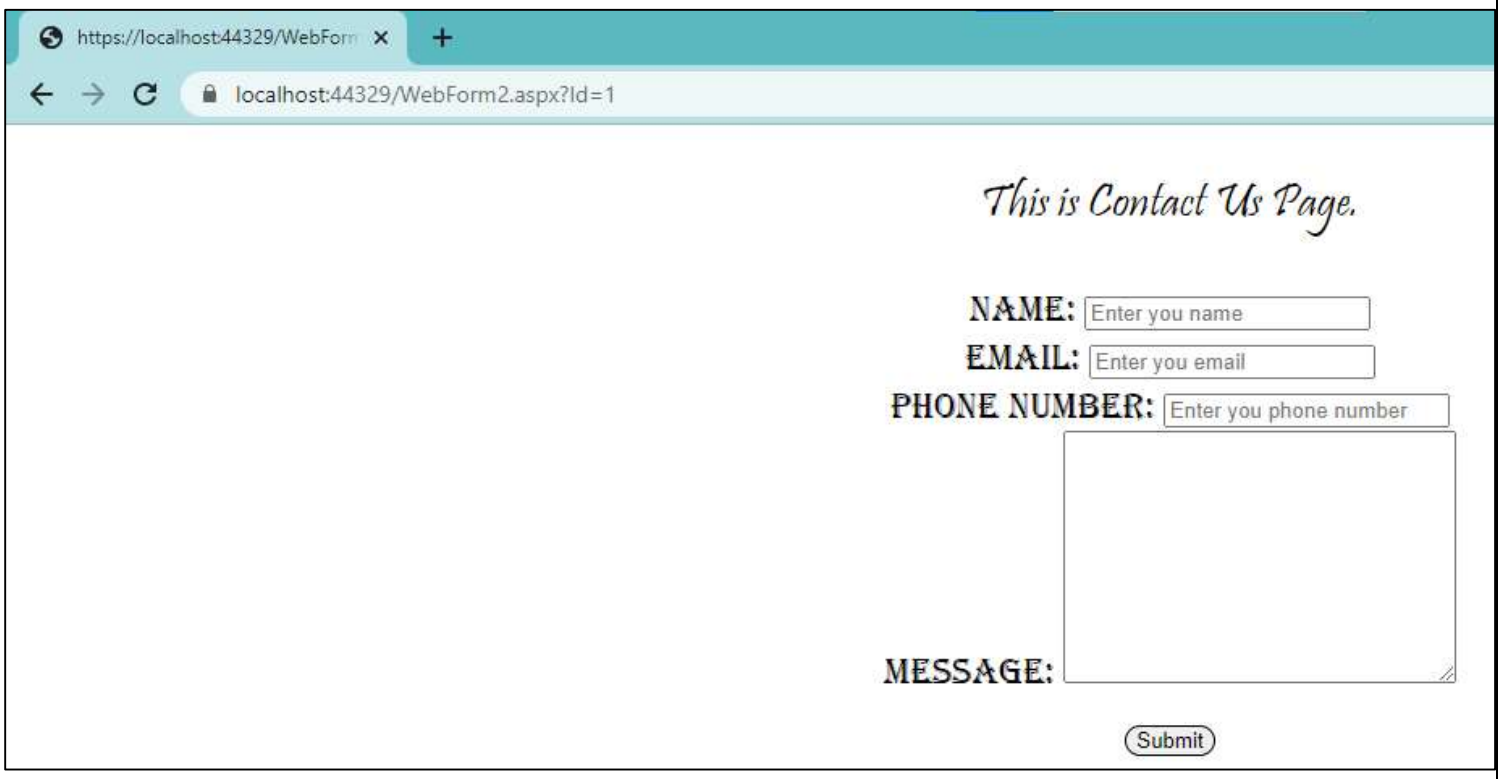
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

namespace Prac_9C
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!this.IsPostBack)
            {
                DataTable dt = new DataTable();
                dt.Columns.AddRange(new DataColumn[3] { new DataColumn("Id"), new DataColumn("Name"), new
DataColumn("Country") });
                dt.Rows.Add(1, "Aditya", "USA");
                dt.Rows.Add(2, "Suman", "China ");
                dt.Rows.Add(3, "Rohan", "Czechoslovakia");
                dt.Rows.Add(4, "Omkar", "Nepal");
                dt.Rows.Add(5, "Shubham", "Japan");
                GridView1.DataSource = dt;
                GridView1.DataBind();
            }
        }


    }
}
```

WebForm2.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs"
Inherits="Prac_9C.WebForm2" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style>
        .contact-box{
            text-align:center;
        }
        p{
            font-family:Pristina;
            font-size:30px;
        }
        .contacts label{
            font-family:Algerian
        }
        .submit input{
            border: 1px solid black;
            border-radius: 20px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div class="contact-box">
        <p>This is Contact Us Page.</p>
                <div class="contacts" style="font-size: 22px; font-family: Calibri;">
                    <label for="name">Name: </label>
                    <input type="text" name="name" id="name" placeholder="Enter you name">
                </div>
                <div class="contacts" style="font-size: 22px; font-family: Calibri;">
                    <label for="email">Email: </label>
                    <input type="email" name="email" id="email" placeholder="Enter you email">
                </div>
                <div class="contacts" style="font-size: 22px; font-family: Calibri;">
                    <label for="phone">Phone Number: </label>
                    <input type="text" name="phone" id="phone" placeholder="Enter you phone
number">
                </div>
                <div class="contacts" style="font-size: 22px; font-family: Calibri;">
                    <label for="message">Message: </label>
                    <textarea name="message" id="Message" cols="30" rows="10"></textarea>
                </div>
                <div class="content" style="font-size: 34px; font-family: Calibri;">
                    <div class="submit"><input type="submit" value="Submit"><br />
                    </div>
                </div>
            <br />

        </div>
    </form>
</body>
</html>
```

Output:


https://localhost:44329/WebForm
localhost:44329/WebForm1.aspx
Name
Country
Aditya
USA
Suman
China
Rohan
Czechoslovakia
Omkar
Nepal
Shubham
Japan


https://localhost:44329/WebForm
localhost:44329/WebForm2.aspx?Id=1
This is Contact Us Page.
NAME:
Enter you name
EMAIL:
Enter you email
PHONE NUMBER:
Enter you phone number
MESSAGE:
Submit

Practical 10: Working with AJAX and XML

a) Create a web application to demonstrate reading and writing operation with XML.

XML stands for Extensible markup language. By using XML we can easily retrieve and store data and display the data without using databases in our applications. If we need to display dynamic data in our application it will take time to connect databases and retrieve data from databases but if we use XML to store data we can do operations with xml files directly without using databases. If we store data in a database that is incompatible to some of the computer applications but if we store data in XML format it will support all applications. It is independent for all software applications and it is accessible with all applications.

Source code:

XMLDemo.aspx code:

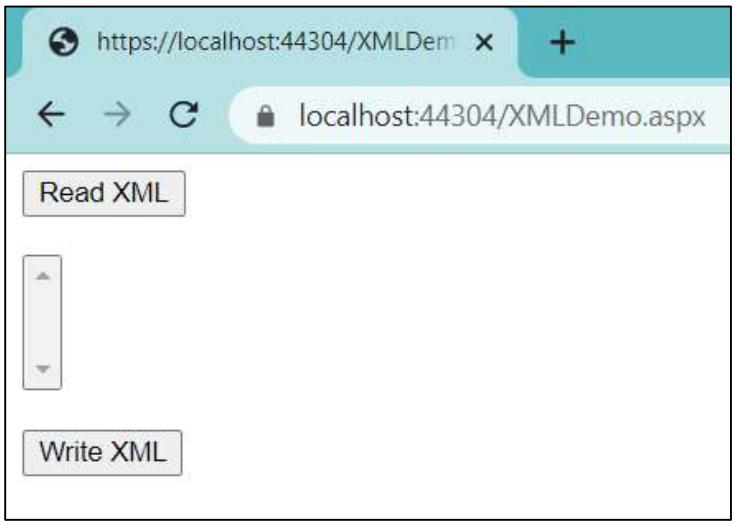
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="XMLDemo.aspx.cs" Inherits="XMLandFormView.XMLDemo" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="readBtn" runat="server" Text="Read XML" OnClick="readBtn_Click"></asp:Button><br /><br />
            <asp:ListBox ID="ListBox1" runat="server"></asp:ListBox><br /><br />
            <asp:Button ID="writeBtn" runat="server" Text="Write XML" OnClick="writeBtn_Click"></asp:Button>
        </div>
    </form>
</body>
</html>
```

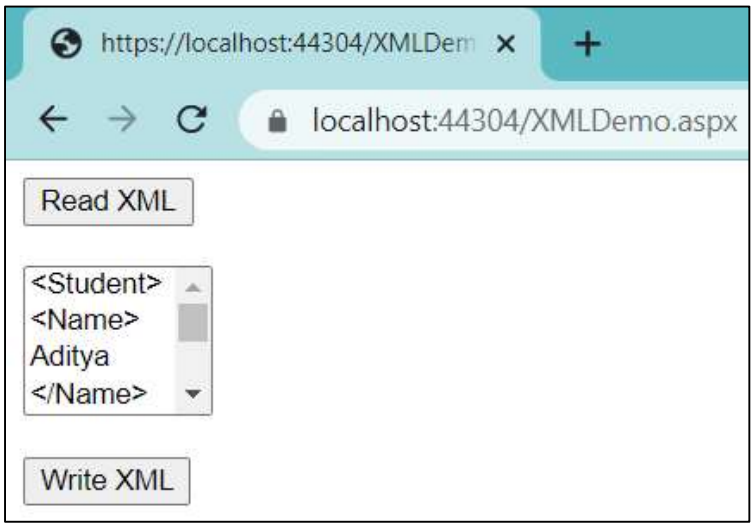
XMLDemo.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml;
namespace XMLandFormView
{
    public partial class XMLDemo : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void readBtn_Click(object sender, EventArgs e)
        {
            String xml = "C:\\Users\\Admin\\source\\repos\\XMLandFormView\\XMLandFormView\\myXmlFile.xml";
            XmlReader rd = XmlReader.Create(xml);
            while (rd.Read())
            {
                switch (rd.NodeType)
                {
                    case XmlNodeType.Element:
                        ListBox1.Items.Add("<"+rd.Name+">");
                        break;
                    case XmlNodeType.Text:
                        ListBox1.Items.Add(rd.Value);
                        break;
                    case XmlNodeType.EndElement:
                        ListBox1.Items.Add("</"+rd.Name+">");
                        break;
                }
            }
        }
        protected void writeBtn_Click(object sender, EventArgs e)
        {
            XmlTextWriter tw = new
XmlTextWriter("C:\\Users\\Admin\\source\\repos\\XMLandFormView\\XMLandFormView\\myXmlFile.xml", null);
            tw.WriteStartDocument();
            tw.WriteStartElement("Student");
            tw.WriteStartElement("Name","");
            tw.WriteString("Aditya");
            tw.WriteEndElement();
```

```
            tw.WriteStartElement("Roll_no","");
            tw.WriteString("3030");
            tw.WriteEndElement();
            tw.WriteEndElement();
            tw.WriteEndDocument();
            tw.Close();
        }
    }
}
```

Output:






```
<?xml version="1.0"?><Student><Name>Aditya</Name><Roll_no>3030</Roll_no></Student>
```

b) Create a web application to demonstrate use of various Ajax controls.

AJAX stands for Asynchronous JavaScript and XML. This is a cross platform technology which speeds up response time. The AJAX server controls add script to the page which is executed and processed by the browser.

However,like other ASP.NET server controls, these AJAX server controls also can have methods and event handlers associated with them, which are processed on the server side.

i) Update Panel:

Update Panel: UpdatePanel controls are a central part of AJAX functionality in ASP.NET. They are used with the ScriptManager control to enable partial-page rendering. Partial-page rendering reduces the need for synchronous postbacks and complete page updates when only part of the page has to be updated.

Source code:

WebForm1.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Prac10_i.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <asp:UpdatePanel runat="server" id="UpdatePanel" updatemode="Conditional">
        <Triggers>
                <asp:AsyncPostBackTrigger controlid="UpdateButton2" eventname="Click" />
        </Triggers>
                <ContentTemplate>
                <asp:Label runat="server" id="DateTimeLabel1" />
                <asp:Button runat="server" id="UpdateButton1" onclick="UpdateButton_Click" text="Update" />
                </ContentTemplate>
        </asp:UpdatePanel>
        <asp:UpdatePanel runat="server" id="UpdatePanel1" updatemode="Conditional">
                <ContentTemplate>
                <asp:Label runat="server" id="DateTimeLabel2" />
                <asp:Button runat="server" id="UpdateButton2" onclick="UpdateButton_Click" text="Update" />
                </ContentTemplate>
        </asp:UpdatePanel>
        </div>
    </form>
</body>
</html>
```

WebForm1.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Prac10_i
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void UpdateButton_Click(object sender, EventArgs e)
        {
            DateTimeLabel1.Text = DateTime.Now.ToString();
            DateTimeLabel2.Text = DateTime.Now.ToString();
        }
    }
}
```

Output:

ii) Timer Control:

Timer Control: The ASP.NET AJAX Timer control performs postbacks at defined intervals. If you use the Timer control with an UpdatePanel control, you can enable partial-page updates at a defined interval. You can also use the Timer control to post the whole page.

Source code:

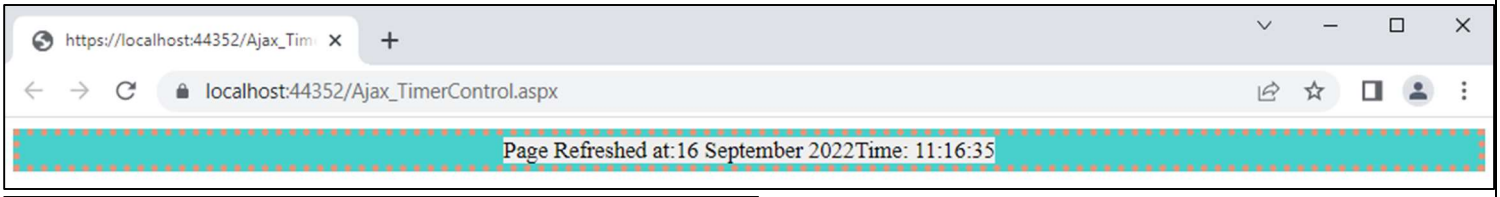
Ajax_TimerControl.aspx code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Ajax_TimerControl.aspx.cs"
Inherits="Ajax_DisconnectedDataAccess.Ajax_TimerControl" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>

            <asp:UpdatePanel ID="UpdatePanel1" runat="server">
                <ContentTemplate>
                    <asp:Panel ID="Panel1" runat="server" BackColor="MediumTurquoise"
BorderStyle="Dotted" BorderColor="DarkSalmon" BorderWidth="5">
                        <center>
                            <asp:Label ID="label2" runat="server" BackColor="ButtonShadow"
Text="Label"></asp:Label>
                            <asp:Timer ID="timer1" runat="server" Interval="1000"></asp:Timer>
                        </center>
                    </asp:Panel>
                </ContentTemplate>
            </asp:UpdatePanel>
        </div>
    </form>
</body>
</html>
```

Ajax_TimerControl.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Ajax_DisconnectedDataAccess
{
    public partial class Ajax_TimerControl : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            label2.Text = "Page Refreshed at:" + DateTime.Today.ToLongDateString() + "Time: " +
DateTime.Now.ToLongTimeString();
        }
    }
}
```

Output:



ii) Update Progress Timer Control:

Update Progress Control: The UpdateProgress control provides status information about partial-page updates in UpdatePanel controls. You can customise the default content and the layout of the UpdateProgress control. To prevent flashing when a partial-page update is very fast, you can specify a delay before the UpdateProgress control is displayed

Source code:

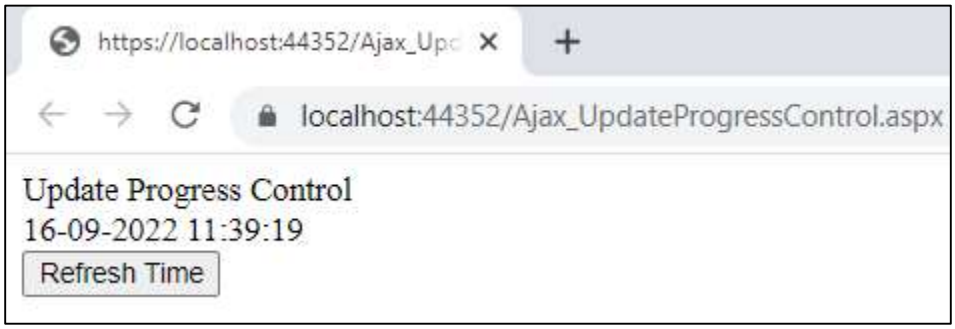
Ajax_UpdateProgressControl.aspx code:

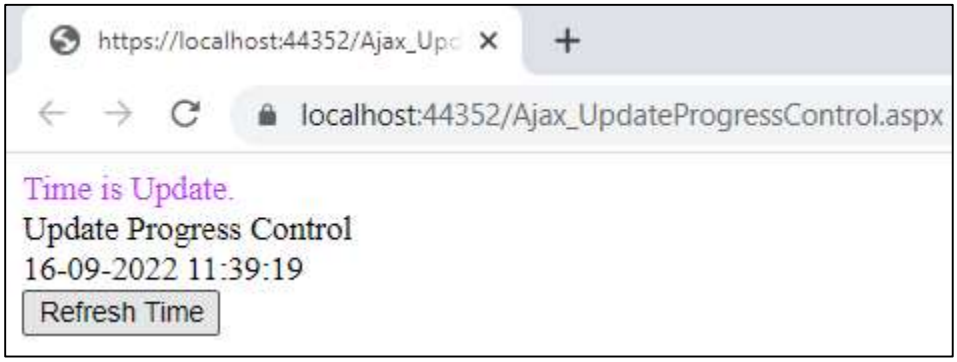
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Ajax_UpdateProgressControl.aspx.cs"
Inherits="Ajax_DisconnectedDataAccess.Ajax_UpdateProgressControl" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>

            <asp:UpdateProgress ID="UpdateProgress1" runat="server">
                <ProgressTemplate>
                    <asp:Label ID="Label2" runat="server" Text="Time is Update."
ForeColor="#CC33FF"></asp:Label>
                </ProgressTemplate>
            </asp:UpdateProgress>

            <asp:UpdatePanel ID="UpdatePanel1" runat="server">
                <ContentTemplate>
                    Update Progress Control
                    <br />
                    <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
                    <br />
                    <asp:Button ID="Button1" runat="server" Text="Refresh Time" />
                </ContentTemplate>
            </asp:UpdatePanel>


        </div>
    </form>
</body>
</html>
```

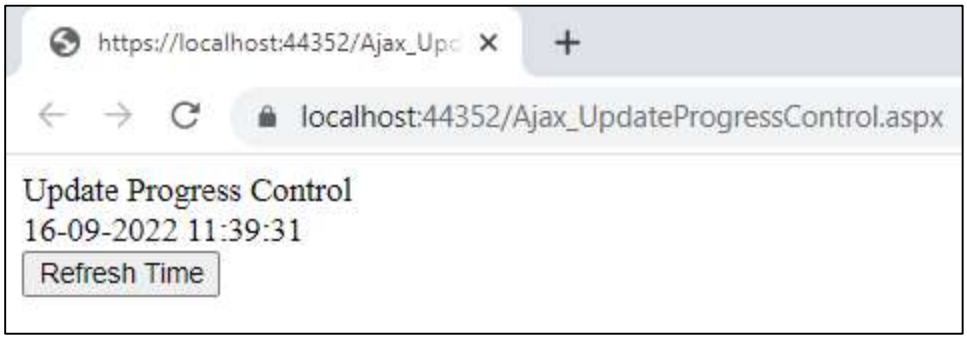
Ajax_UpdateProgressControl.aspx.cs code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Ajax_DisconnectedDataAccess
{
    public partial class Ajax_UpdateProgressControl : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            System.Threading.Thread.Sleep(2000);
            Label1.Text = DateTime.Now.ToString();
        }
    }
}
```

Output:


https://localhost:44352/Ajax_Up
localhost:44352/Ajax_UpdateProgressControl.aspx
Update Progress Control
16-09-2022 11:39:19
Refresh Time


https://localhost:44352/Ajax_Up
localhost:44352/Ajax_UpdateProgressControl.aspx
Time is Update.
Update Progress Control
16-09-2022 11:39:19
Refresh Time


https://localhost:44352/Ajax_Up
localhost:44352/Ajax_UpdateProgressControl.aspx
Update Progress Control
16-09-2022 11:39:31
Refresh Time

Practical 11: Programs to create and use DLL:

A Dynamic Link library (DLL) is a library that contains functions and codes that can be used by more than one program at a time. Once we have created a DLL file, we can use it in many applications. The only thing we need to do is to add the reference/import the DLL File. Both DLL and .exe files are executable program modules but the difference is that we cannot execute DLL files directly.

To create a dll file, you need to open a

A] Calculator:

Source code:

Create a .dll file by opening a new project and selecting class library(.NET framework) as a project template, then add .cs file named as "**Functions**", then write the below code into it.

Functions.cs:

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

namespace functions
{
    public static class Function
    {
        public static double add(double a, double b)
        {
            return a + b;
        }
        public static double sub(double a, double b)
        {
            return a - b;
        }
        public static double multi(double a, double b)
        {
            return a * b;
        }
        public static double div(double a, double b)
        {
            return a / b;
        }
    }
}
```

After creating the .dll file, open a new project by Web Application (.Net Framework) library and add the .dll file in the reference section by browsing it in the folder where you have created it.

WebForm1.aspx code:

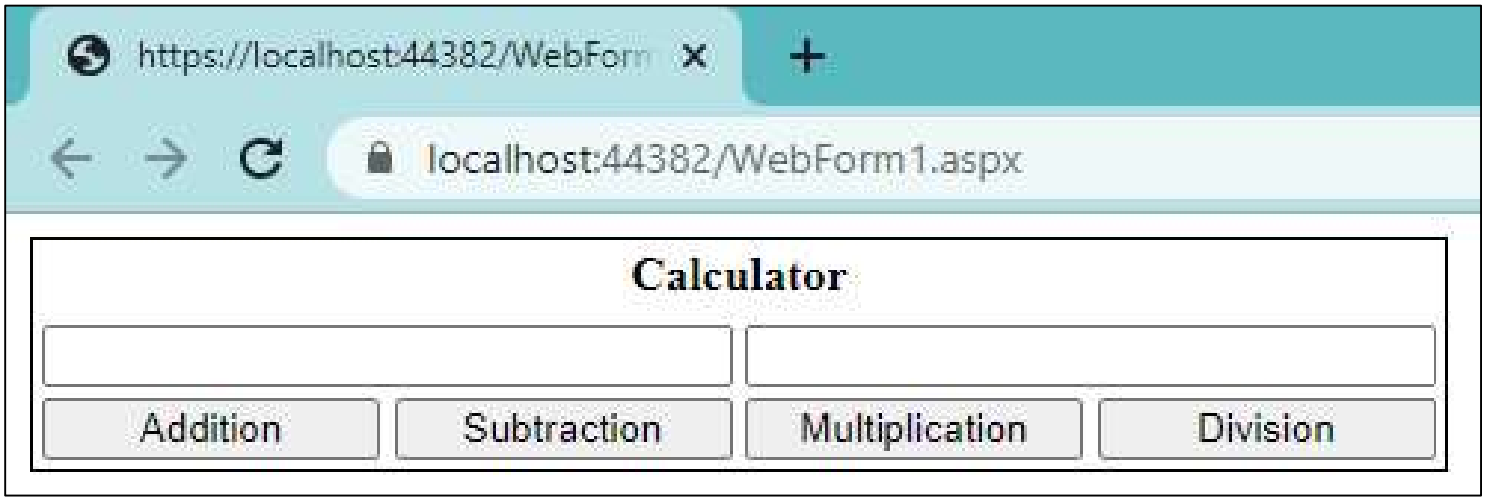
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Calculator.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style>
        table{
            border: 1px solid black;
        }
    </style>
    </head>
<body>
    <form id="form1" runat="server">
        <div>
            <table>
                <tr>
                    <th colspan="4">Calculator</th>
                </tr>
                <tr>
                    <td colspan="4"><asp:Label ID="Output" runat="server" Text="" Font-Size="20px"></asp:Label></td>
                </tr>
                <tr>
                   <td colspan="2"><asp:TextBox ID="Value1" runat="server" Width="225px"></asp:TextBox></td>
                    <td colspan="2"><asp:TextBox ID="Value2" runat="server" Width="225px"></asp:TextBox></td>
```

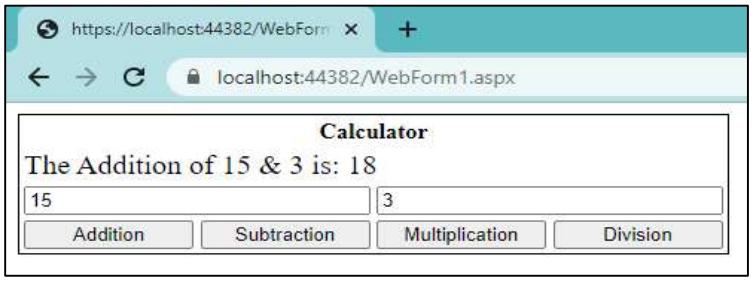
```
                </tr>
                <tr>
                    <td><asp:Button ID="Add" runat="server" Text="Addition" OnClick="Add_Click1" Width="114px" /></td>
                    <td><asp:Button ID="Sub" runat="server" Text="Subtraction" OnClick="Sub_Click" Width="114px"
/></td>
                    <td><asp:Button ID="Multi" runat="server" Text="Multiplication" OnClick="Multi_Click" Width="114px"
/></td>
                    <td><asp:Button ID="Div" runat="server" Text="Division" OnClick="Div_Click" Width="114px" /></td>
                </tr>
            </table>
           <br />
        </div>
    </form>
</body>
</html>
```

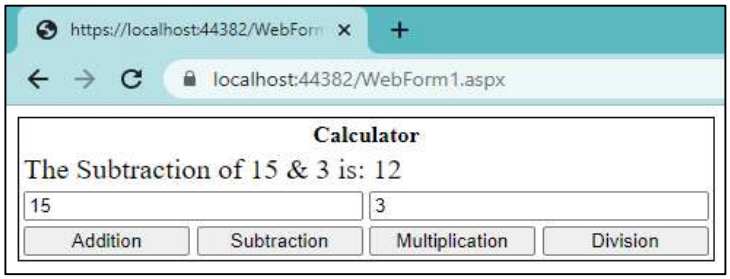
WebForm1.aspx.cs code:

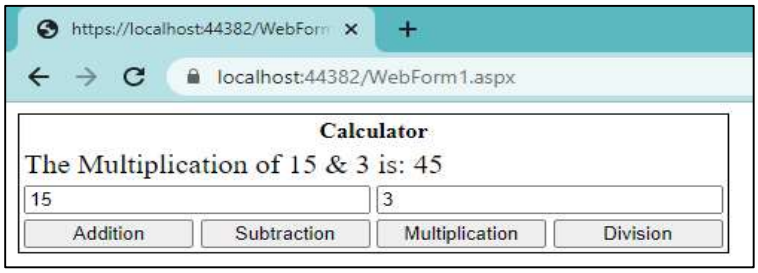
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using functions;

namespace Calculator
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Add_Click1(object sender, EventArgs e)
        {
            double a, b;
            a = Convert.ToDouble(Value1.Text);
            b = Convert.ToDouble(Value2.Text);
            Output.Text = "The Addition of " + a + " & " + b + " is: " + Convert.ToString(Function.add(a, b));
        }
        protected void Sub_Click(object sender, EventArgs e)
        {
            double a, b;
            a = Convert.ToDouble(Value1.Text);
            b = Convert.ToDouble(Value2.Text);
            Output.Text = "The Subtraction of " + a + " & " + b + " is: " + Convert.ToString(Function.sub(a, b));
        }
        protected void Multi_Click(object sender, EventArgs e)
        {
            double a, b;
            a = Convert.ToDouble(Value1.Text);
            b = Convert.ToDouble(Value2.Text);
            Output.Text = "The Multiplication of " + a + " & " + b + " is: " + Convert.ToString(Function.multi(a, b));
        }
        protected void Div_Click(object sender, EventArgs e)
        {
            double a, b;
            a = Convert.ToDouble(Value1.Text);
            b = Convert.ToDouble(Value2.Text);
            Output.Text = "The Division of " + a + " & " + b + " is: " + Convert.ToString(Function.div(a, b));
        }
    }
}
```

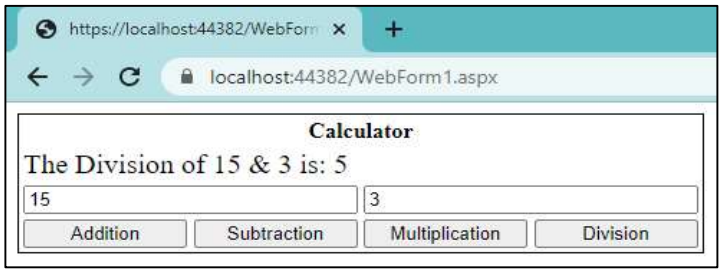
Output:


https://localhost:44382/WebForm
localhost:44382/WebForm1.aspx
Calculator
Addition
Subtraction
Multiplication
Division


https://localhost:44382/WebForm
localhost:44382/WebForm1.aspx
Calculator
The Addition of 15 & 3 is: 18
15
3
Addition
Subtraction
Multiplication
Division


https://localhost:44382/WebForm
localhost:44382/WebForm1.aspx
Calculator
The Subtraction of 15 & 3 is: 12
15
3
Addition
Subtraction
Multiplication
Division


https://localhost:44382/WebForm
localhost:44382/WebForm1.aspx
Calculator
The Multiplication of 15 & 3 is: 45
15
3
Addition
Subtraction
Multiplication
Division


https://localhost:44382/WebForm
localhost:44382/WebForm1.aspx
Calculator
The Division of 15 & 3 is: 5
15
3
Addition
Subtraction
Multiplication
Division