.NET PRACTICAL

Hepi Chaniyara

160470107008

Table of Contents

PRACTICAL:1

AIM: INTRODUCTION TO C#

Variables:

Initialization

Scope

Constant

Predefined Data Types

Value Types

Reference TYpes

Flow Control

Conditional Statements(if, switch)

Loop(for, while, dowhile, foreach)

Jump(goto, break, continue, return)

Eumerations

Passing Arguments

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

namespace aim
{
    class Program
    {
            static int newint=100;
            public enum TimeOfDay
            {
            Morning = 0,
            Afternoon = 1,
            Evening = 2
```

```
            }
        public static void Main(string[] args)
        {
            Console.WriteLine("\n integer types");
            sbyte sb = 10;
            short s = 33;
            int i = 10;
            long l = 33L;
            byte b = 22;
            ushort us = 33;
            uint ul = 33u;
            ulong ulo = 33ul;
            Console.WriteLine("{0},{1},{2},{3},{4},{5},{6},{7}", sb, s, i, l, b, us,
ul, ulo);
            float f = 1.122345656767f;
            double d = 12.1234455657878797;
            Console.Write("\nFloat and Double:\n");
            Console.WriteLine("{0} and \n{1}", f, d);
                    decimal dec=111.66666666666666666666M;
                    Console.WriteLine("decimal:\n{0} ",dec);
                    Console.WriteLine("\nBoolean:");
                    bool boolean =true;
                    Console.WriteLine("Status: " + boolean);
          /     Console.ReadLine();
                    char character ='d';
                    Console.WriteLine(character);
                    character = '\0';
                    Console.WriteLine("Now null: " + character);
                    object o1 = "Hi, I am ALICE";
                    object o2 = 15.3454365;
                    string strObj = o1 as string;
                    Console.WriteLine(strObj);
                    Console.WriteLine(o1.GetHashCode() + " " + o1.GetType());
                    Console.WriteLine(o2.GetHashCode() + " " + o2.GetType());
```

```
Console.WriteLine(o1.Equals(o2));
string s1, s2;
s1 = "this is string";
s2 = s1;
Console.WriteLine("S1 is: {0} and s2 is {1}", s1, s2); s2 =
"other string";
Console.WriteLine("S1 is: {0} and s2 is {1}", s1, s2); s1 =
"c:C:\\Users\\Dell\\source\\repos\\aim";
Console.WriteLine(s1);
s1 = @"c:C:\Users\Dell\source\repos\aim\aim";
Console.WriteLine(s1);
s1 = @"We can also write
like this";
Console.WriteLine(s1);
bool isZero;
Console.WriteLine("\nFlow Control: (if)\ni is " + i);
if (i == 10)
{
isZero = true;
Console.WriteLine("i is Zero {0}",isZero);
}
else
{
isZero = false;
Console.WriteLine("i is Non - zero");
}
int integerA = 1;
Console.WriteLine("\nSwitch:");
switch (integerA)
{
case 1:
Console.WriteLine("integerA = 1");
break;
case 2:
```

```
                    Console.WriteLine("integerA = 2");
                    //goto case 3;
                    break;
                    case 3:
                    Console.WriteLine("integerA = 3");
                    break;
                    default:
                    Console.WriteLine("integerA is not 1, 2, or 3");
                    break;}
                    WriteGreeting(TimeOfDay.Morning);
                    Console.WriteLine("Argument is: {0}",args[1]);

                     void WriteGreeting(TimeOfDay timeOfDay)
                    {
                    switch (timeOfDay)
                    {
                    case TimeOfDay.Morning:
                    Console.WriteLine("Good morning!");
                    break;
                    case TimeOfDay.Afternoon:
                    Console.WriteLine("Good afternoon!");
                    break;
                    case TimeOfDay.Evening:
                    Console.WriteLine("Good evening!");
                    break;
                    default:
                    Console.WriteLine("Hello!");
                    break;
        }       }
                    Console.WriteLine("Scope of Variables.\n1:"); int
            newint=0;
                    int j;
            for (/*int*/ j = 0; j < 2; j++) //removing comment from for loop will
raise error
```

```
{
    //int j;
    //uncomment above line to error "A local variable named 'j' cannot be declared in this
    //scope because it would give a different meaning to 'j', which is already
    //used in a 'parent or current' scope to denote something else"
    Console.Write("{0} {1}\n", newint, Program.newint);
}
        Console.WriteLine("2:");
for (int k = 0; k < 3; k++)
{
    Console.Write("{0} ", k);
}//Scope of k ends here
Console.Write("\n");
//Console.Write(k);
//uncomment above line to see error "The name 'k' does not exist in the current context"
for (int k = 3; k > 0; k--)
{
    Console.Write("{0} ", k);
}//scope of k ends here again

Console.WriteLine("Constants");
        const int valConst = 100; // This value cannot be changed.
Console.WriteLine("{0} is constant value", valConst); //valConst = 45;
//uncomment above line to see error "The left-hand side of an assignment must be a variable, property or indexer"

//const only allow constant variables into the expression const
int valConst2 = valConst + 9 /* + j*/;
//remove comments from the above line to see error "The expression being assigned to 'valConst2' must be constant"
Console.WriteLine("Another Constant: {0}", valConst2);
```

```
            Console.WriteLine("\nPredefined Data Types\n\nValue Types and
Reference Types");

            //Value Types

            int vali = 2, valj = vali;

            Console.WriteLine("vali is: {0} and valj is: {1}", vali, valj);

            valj = 90;

            Console.WriteLine("vali is: {0} and valj is: {1}", vali, valj);

            //Referece Types

            Vector x, y;

            x = new Vector();

            x.value = 3;

            y = x;

            Console.WriteLine("x is: {0} and y is:{1}", x.value, y.value);

            y.value = 234;

            Console.WriteLine("x is: {0} and y is:{1}", x.value, y.value);

            //If a variable is a reference, it is possible to indicate that it does not
refer to any object by setting its value to null:

            y = null;

            //Console.Write("Value for y is: " + y.value);

            //uncomment above line to see runtime exception
"System.NullReferenceException: Object reference not set to an instance of an
object."

//CTS                    }

                         public class Vector

                         {

                         public int value;

                         }

  }

}
```

```
 integer types
10,33,10,33,22,33,33,33

Float and Double:
1.122346 and
12.1234455657879
decimal:
111.66666666666666666666

Boolean:
Status: True
d
Now null:
Hi, I am ALICE
1487365606 System.String
1302462624 System.Double
False
S1 is: this is string and s2 is this is string
S1 is: this is string and s2 is other string
c:C:\Users\Dell\source\repos\aim
c:C:\Users\Dell\source\repos\aim\aim
We can also write
                        like this

Flow Control: (if)
i is 10
i is Zero True

Switch:
integerA = 1
Good morning!
Scope of Variables.
1:
0 100
0 100
2:
0 1 2
3 2 1 Constants
100 is constant value
Another Constant: 109

Predefined Data Types

Value Types and Reference Types
vali is: 2 and valj is: 2
vali is: 2 and valj is: 90
x is: 3 and y is:3
x is: 234 and y is:234
```

PRACTICAL:2

Program 1

Write console based program in code behind language VB or C# to print following pattern.

@ @ @ @ @

@ @ @ @

@ @ @

@ @

@

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace practical2
{
    class Program
    {
    static void Main(string[] args)
        {
            for(int i=5;i>0;i--)
            {
                for (int j = i; j > 0; j--)
                {
                    Console.Write("@");
                }
                Console.WriteLine(" ");
            }
            Console.ReadKey();
        }
    }
}
```

Program 2

Write console based program in code behind language VB or C# to print following pattern.

1

1 2

1 2 3

1 2 3 4

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

namespace practical2._1
{
    class Program
    {
        static void Main(string[] args)
        {
            for(int i=1;i<=5;i++)
```

```
            {
                for(int j=i;j>0;j--)
                {
                    Console.Write("{0}",i);
                }
                Console.WriteLine("");
            }
            Console.ReadKey();
        }
    }
}
```

Program 3

Write C# code to prompt a user to input his/her name and country name and then the output will be shown as an example below:

Hello Ram from country India

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace practical2._2
{
```

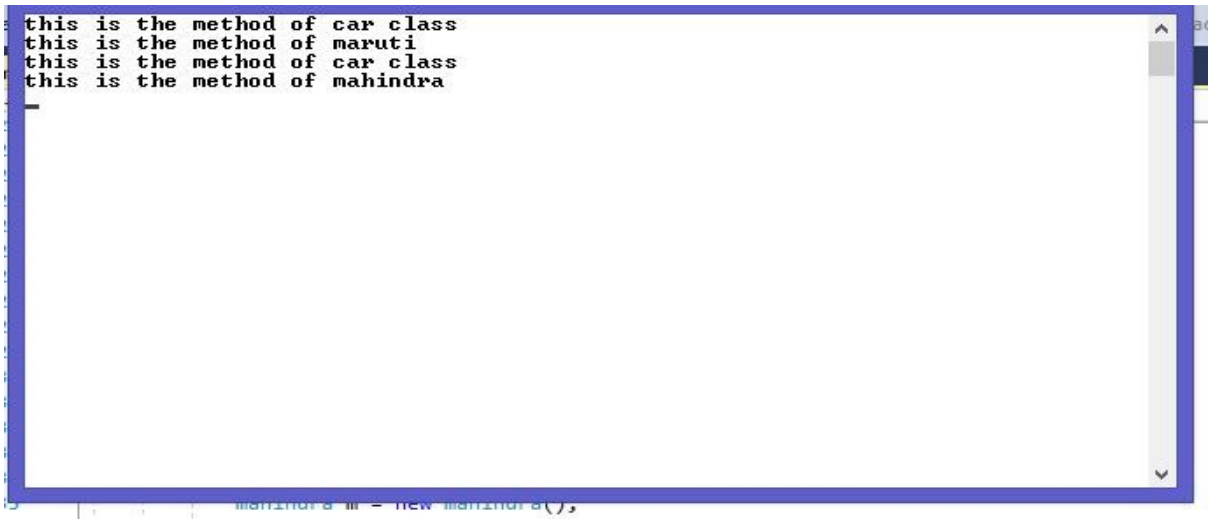
```
    class Program
    {
        static void Main(string[] args)
        {
            string name;
            string country;
            Console.WriteLine("enter your name:");
            name=Console.ReadLine();
            Console.WriteLine("enter your country:");
            country = Console.ReadLine();
            Console.WriteLine("hello {0} from country {1}",name,country);
            Console.ReadKey();
        }
    }
}
```

Program 4

What is inheritance? Create C# console application to define Car class and derive Maruti and Mahindra from it to demonstrate inheritance.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace practical2._3
{
    class car
    {
        public void Method1()
        {
            Console.WriteLine("this is the method of car class");
        }
    }
    class maruti:car
    {
        public void method2()
        {
            Console.WriteLine("this is the method of maruti");
            Console.ReadKey();
        }
    }
    class mahindra:car
    {
        public void method3()
        {
```

```
            Console.WriteLine("this is the method of mahindra");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            mahindra m = new mahindra();
            maruti m1 = new maruti();
            m.Method1();
            m1.Method1();
            Console.ReadKey();
        }
    }
}
```



PRACTICAL:3

AIM: Method & constructor overloading

Program 1

Write a c# program to add two integers, two vectors and two metric using method overloading.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace p3
{
    public class Add
    {
        public void add()
        {
            int[,] m1 = new int[50, 50]; int[,] m2 = new
            int[50, 50]; int[,] m3 = new int[50, 50];
            Console.WriteLine("enter size of array:");
            int size = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("enter first array:"); for (int i = 0; i
            < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    m1[i, j] = Convert.ToInt32(Console.ReadLine());
                }
            }
            Console.WriteLine("enter second array:");
            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
```

```
                {
                    m2[i, j] = Convert.ToInt32(Console.ReadLine());
                }
            }

            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    m3[i, j] = m1[i, j] + m2[i, j];
                }
            }
            Console.WriteLine("addition array:");
            for (int i = 0; i < size; i++)
            {
                Console.Write("\n");
                for (int j = 0; j < size; j++)
                {
                    Console.Write("{0}\t", m3[i, j]);
                }
                Console.Write("\n");
            }
        }
        public int add(int a, int b)
        {
            return (a + b);
        }
    }
        public class Vector
        {
            public void add()
            {
                Console.WriteLine("enter first vector"); int x =
                Convert.ToInt32(Console.ReadLine());
```

```
                int y = Convert.ToInt32(Console.ReadLine());
                int z = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("enter second vector");
                int x1 = Convert.ToInt32(Console.ReadLine());
                int y1 = Convert.ToInt32(Console.ReadLine());
                int z1 = Convert.ToInt32(Console.ReadLine());
                int x2 = x + x1;
                int y2 = y + y1;
                int z2 = z + z1;
                Console.WriteLine("<" + x2 + "," + y2 + "," + z2 + ">");

            }
        }
    class Program
      {
        static void Main(string[] args)
        {

            Add a1 = new Add();
            Vector v1 = new Vector();
            v1.add();
            a1.add();
            int res=a1.add(1, 2);
            Console.Write("method overloading for addtion{0}",res);
            Console.ReadLine();

        }
      }
}
```

```
enter first vector
1
2
3
enter second vector
1
2
3
<2,4,6>
enter size of array:
2
enter first array:
1
2
3
4
enter second array:
1
2
3
4
addition array:

2       4

6       8
method overloading for addtion3
```

Program 2

Write a c# program that create student object. Overload constror to create new instant with following details.

1. Name

2. Name, Enrollment

3. Name, Enrollment, Branch

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
namespace p3a1
{
    class Program
    {
        public int ID { get; set; }
```

```
        public string Name { get; set; }
        String name, branch;
        int enrol;
        public Program(String name)
        {
            this.name = name;
            Console.WriteLine("constructor 1:" + name);
        }
        public Program(String name, int enrol)
        {
            this.name = name;
            this.enrol = enrol;
            Console.WriteLine("constructor 2:" + name + " " + enrol);

        }
        public Program(String name, int enrol, String branch)
        {
            this.name = name;
            this.enrol = enrol;
            this.branch = branch;
            Console.WriteLine("constructor 3:" + name + " " + enrol + " " +
branch);

        }
        static void Main(string[] args)
        {
  Program p1 = new Program("bob");
            Program p2 = new Program("bob", 1);
            Program p3 = new Program("bob", 1, "computer");
              Console.ReadLine();

        }
    }
}
```

```
constructor 1:bob
constructor 2:bob 1
constructor 3:bob 1 computer
```

PRACTICAL:4

Create a c# program to find Methods, Properties and Constructors from class of running program.(Use Class from previous practical)

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Reflection;
using System;


namespace p2
{
    class P4
    {
        public static void Main() {
            Type T = Type.GetType("p2.Example");
            MethodInfo[] methods = T.GetMethods();
            foreach (MethodInfo method in methods)
            {
                Console.WriteLine(method.ReturnType + " " + method.Name);
            }


            PropertyInfo[] properties = T.GetProperties();


            Console.WriteLine("\nProperties");
            foreach (PropertyInfo property in properties)
            {
                Console.WriteLine(property.PropertyType + " " + property.Name);
            }


            Console.WriteLine("\nConstructors"); ConstructorInfo[]
            constructors = T.GetConstructors(); foreach
            (ConstructorInfo constructor in constructors) {
                Console.WriteLine(constructor.ToString());
            }
```

```
            }
        }
        class Example {
            public string name { get; set; }
            public int enrollment { get; set; }
            public string branch { get; set; }


            public Example() { }
            public Example(int enrollment, string name)
                { this.enrollment = enrollment; this.name =
                name;
            }
            public Example(int enrollment, string name,string branch)
            {
                this.enrollment = enrollment;
                this.name = name;
                this.branch = branch;
            }
            public void displayName()
                { Console.WriteLine("Name={0}",this.name);
            }
            public void displayEnroll()
                { Console.WriteLine("Enrollment={0}",this.enrollment);
            }
            public void displayBranch()
            {
                Console.WriteLine("Branch={0}", this.branch);
            }


    }
}
```

OUTPUT:

```
System.Int32 get_ID System.Void set_ID
System.String get_Name System.Void set_Name
System.Void printID
System.Void printName System.String ToString
System.Boolean Equals System.Int32 GetHashCode System.Type GetType

Properties
```

System.Int32 ID System.String Name

Constructors

Void .ctor(Int32, System.String) Void .ctor()

PRACTICAL:5

AIM : File Handling

Program 1:

Write a C# program to copy data from one file to another using StreamReader and StreamWriter class.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace p2
{
    class P4_1
    {
        public static void Main(){
            string f1 = @"f1.txt";
            string f2 = @"f2.txt";
            using (StreamReader reader = new StreamReader(f1))
            using (StreamWriter writer = new StreamWriter(f2))
                writer.Write(reader.ReadToEnd());
        }
    }
}
```

Program 2:

Write a C# Program to Read Lines from a File until the End of File is reached.

```
using System;
using
System.Collections.Generic;
 using System.Linq;
 using System.Text;
 using System.IO;
 namespace P2

{
    public class CopyFile
    {
        public void copyFile(string f1, string f2)
        {
            using (StreamReader reader = new StreamReader(f1))
            using (StreamWriter writer = new StreamWriter(f2))
            {
                string line = null;
                while ((line = reader.ReadLine()) != null)
                    writer.WriteLine(line);
            }
        }
    }
    public class mmain{
        public static void
            Main(){
            CopyFile cp = new CopyFile();
            string f1 = @"E:\Sem-6\VS\p2\p2\f1.txt";
            string f2 = @"E:\Sem-6\VS\p2\p2\f2.txt";
            cp.copyFile(f1,f2);

        }
    }
}
```

Program 3:

Write a C# Program to List Files in a Directory.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace p2
{
    class ListFile
    {
        public static void
            Main() {
        string[] Directories = Directory.GetDirectories(@"E:\Sem-6\VS");
        foreach (string dir in Directories)
            Console.WriteLine(dir);
        string[] files = Directory.GetFiles(@"E:\Sem-6\VS");
        foreach (string file in
            files)
            Console.WriteLine(fil
            e);


        Console.ReadKey();


    }
    }
}
```

PRACTICAL:6

Program 1:

Create Windows Form Application for Student Registration and store student Details in DataBase.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO;

namespace StudentForm
{
    public partial class Form1 : Form
    {
        string imgPath;
        public Form1()
        {
            InitializeComponent();
        }


        private void btnsave_Click(object sender, EventArgs e)
        {
            string gen = null;
            string subject = null;
            if (genMale.Checked == true) {
                gen = "m";
            }
            if (genFemale.Checked == true) {
                gen = "f";
            }
            if (ck1.Checked == true) {
                subject = subject + " s1";
            }
            if (ck2.Checked == true) {
                subject = subject + " s2";
            }
            string source = @"Data Source=Mishil-Patel\SQLExpress;Initial
Catalog=DemoDb;Integrated Security=True;Pooling=False";
```

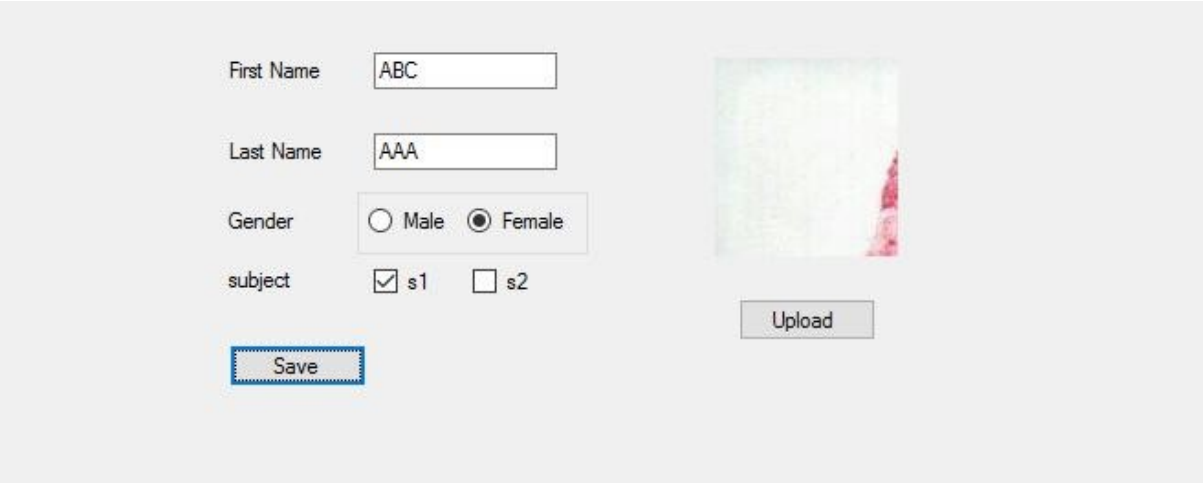
```
                string insert = "insert into tblstudent
(fname,lname,gender,subject,imgStudent) values ('" + txtfname.Text + "','" +
txtlname.Text + "','" + gen + "','" + subject + "','" + (imgPath == null ? "" :
imgPath) + "')";
                //MessageBox.Show(insert);
                //string insert = "insert into tblstudent(fname) values ('jhgjh')";
                SqlConnection conn = new SqlConnection(source);

                SqlCommand cmd = new
                SqlCommand(insert,conn); conn.Open();
                int i = cmd.ExecuteNonQuery();
                conn.Close();

            }

            private void Form1_Load(object sender, EventArgs e)
            {

            }

            private void btnimg_Click(object sender, EventArgs e)
            {
                openFileDialog1.Filter = "Jpg|*.jpg";
                if (openFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    imgPath =  openFileDialog1.SafeFileName;
                    pictureBox.Image = Image.FromFile(openFileDialog1.FileName);
                    //MessageBox.Show(imgPath);
                }
            }


        }
}
```

Program.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace StudentForm
{
    static class Program
    {
        /   <summary>
        /   The main entry point for the application.
```

```
        /     </summary>
        [STAThread]
       static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

OUTPUT:



PRACTICAL:7

Program 1:

ASP.NET Validation Control

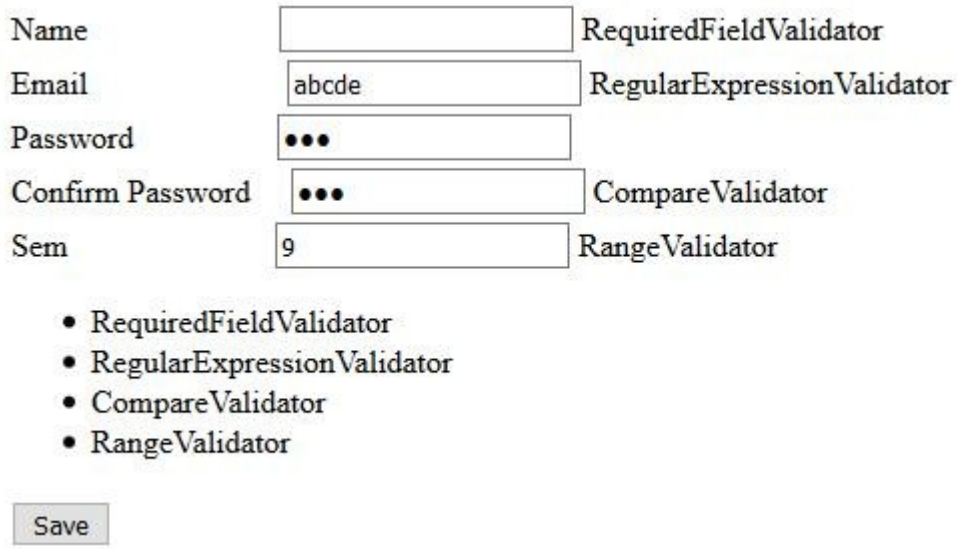
RequiredFieldValidator, CompareValidator, RegularExpressionValidator, CustomValidator, RangeValidator ,ValidationSummary

```
<%@   Page Title="Home Page" Language="C#" AutoEventWireup="true"
     CodeBehind="Default.aspx.cs" Inherits="WebApplication2._Default" %>

<form id="form1" runat="server">
     <div>
          <table>
               <tr>
                    <td>
                         <asp:Label runat="server" Text="Name"></asp:Label>


            &nbs
p;&nbs p;        

                         <asp:TextBox ID="txtname" runat="server" ></asp:TextBox>
                         <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server"
                           ControlToValidate="txtname"
ErrorMessage="RequiredFieldValidator"></asp:RequiredFieldValidator>
                         <br />
                    </td>
               </tr>
               <tr>
                    <td>
                         <asp:Label ID="Email" runat="server" Text="Email"></asp:Label>


            &nbs
p;&nbs p;         

                         <asp:TextBox ID="txtemail" runat="server"></asp:TextBox>
                         <asp:RegularExpressionValidator
ID="RegularExpressionValidator1" runat="server"
                           ErrorMessage="RegularExpressionValidator"
                         ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-
.]\w+)*"
                         ControlToValidate="txtemail"></asp:RegularExpressionValidator>
```

```
                    <br />
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Label ID="Label3" runat="server"
Text="Password"></asp:Label>


            &nbs
p;&nbs p;  

                    <asp:TextBox ID="txtpass" runat="server"
TextMode="Password"></asp:TextBox>
                    <br />
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Label ID="Label4" runat="server" Text="Confirm
Password"></asp:Label>

                       

                    <asp:TextBox ID="txtcpass" runat="server"
TextMode="Password"></asp:TextBox>
                    <asp:CompareValidator ID="CompareValidator1" runat="server"
                     ControlToCompare="txtcpass" ControlToValidate="txtpass"
                     ErrorMessage="CompareValidator"></asp:CompareValidator>
                    <br />
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Label ID="Label5" runat="server" Text="Sem"></asp:Label>


             &
nbs p;         

                    <asp:TextBox ID="txtsem" runat="server"></asp:TextBox>
                    <asp:RangeValidator ID="RangeValidator1" runat="server"
                       ControlToValidate="txtsem" ErrorMessage="RangeValidator"
MaximumValue="8"
                       MinimumValue="1"></asp:RangeValidator>
                    <br />
                    <asp:ValidationSummary ID="ValidationSummary1" runat="server"
/>
```

```
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Button ID="Button1" runat="server" Text="Save" />
                </td>
            </tr>
        </table>
    </div>

</form>
```

OUTPUT:



PRACTICAL:8

AIM: Introduction to Master Pages

Program 1:

Site1.Master:

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

namespace WebApplication1
{
    public partial class Site1 : System.Web.UI.MasterPage {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        public Label LblHeader {
            get {
                return lblheader;
            }
        }
        public Button BtnSearch {
            get {
                return btnsearch;
            }
        }
        public TextBox TxtSearch {
            get {
                return txtsearch;
            }
        }
    }
}
```

WebForm1.aspx:

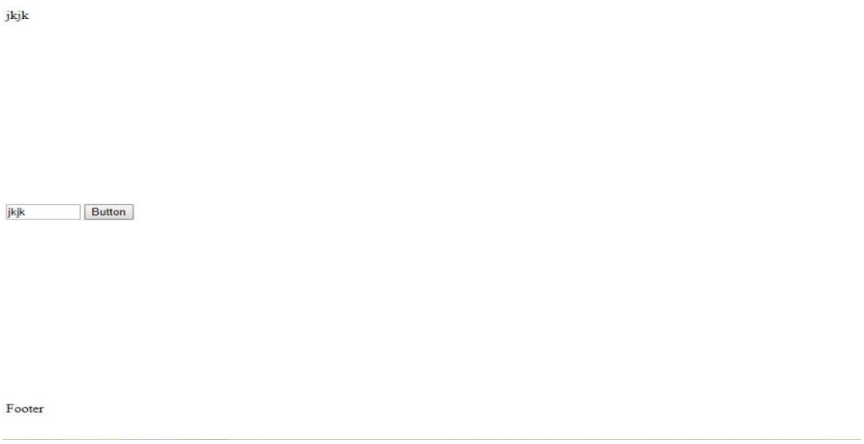
```
<%@ Page Title="" Language="C#"
MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
```

```
<asp:Content ID="Content1"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <asp:TextBox ID="txtname" runat="server" ></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Set Header" onclick="Button1_Click"
/>
</asp:Content>
```

WebForm.aspx.cs:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            ((Site1)Master).LblHeader.Text = txtname.Text;
        }

    }
}
```

OUTPUT:



Program 2:

WebForm2.aspx:

```
<%@ Page Title="" Language="C#"
MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="WebForm2.aspx.cs"
Inherits="WebApplication1.WebForm2" %>
<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <asp:GridView ID="grdstudent" runat="server">
</asp:GridView>
</asp:Content>
```

WebForm2.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
namespace WebApplication1
{
    public partial class WebForm2 : System.Web.UI.Page {
        protected void Page_Init(object sender, EventArgs e)
        {
            ((Site1)Master).BtnSearch.Click += new EventHandler(BtnSearch_Click);
        }

        void BtnSearch_Click(object sender, EventArgs e)
            { getData();
        }
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        void getData() {
            string s= ((Site1)Master).TxtSearch.Text;
            Console.WriteLine(s);
            string source = @"Data Source=computer\SQLExpress;Initial
Catalog=DemoDb;Integrated Security=True;Pooling=False";
            string select = "select * from tblstudent where fname like '%"+
((Site1)Master).TxtSearch.Text + "%'";
            SqlConnection con = new SqlConnection(source);
            SqlCommand cmd = new SqlCommand(select, con);
```

```
                con.Open();
                SqlDataReader rdr = cmd.ExecuteReader();
                grdstudent.DataSource = rdr;
                grdstudent.DataBind();
                con.Close();
            }
        }
}
```

OUTPUT:

Header

search

A

pkstudent	fname	lname	gender	subject	imgStudent
22	ABC	AAA	f	s1	IMG-20170326-WA0009.jpg

Footer