

Index

Name Roll Class.....

SubjectSection Phone.....

College/School

S. No.	Program	Date	Page No.	Remarks
1.	How to create and access the class component library (.dll) and consume that file in the console application?		1-5	
2.	How to give a strong name to an assembly and install that assembly in the global assembly cache?		6-7	
3.	How to disassemble the assembly and regenerate the assembly again?		8-10	
4.	Demonstrate the usage of namespace by accessing it by a. Fully qualified name b. Using directive c. Using alias directive		11-14	
5.	Simulate DLL HELL problem and solve it with the concept of versioning.		15	
6.	Write a program in C# to print welcome Message.		16	
7.	Demonstrate two ways for writing into console.		17-19	
8.	Write a program to demonstrate boxing and unboxing.		20	
9.	Write a C# program to check whether the number is prime or not, input will be taken from the user.		21	
10.	Write a program in C# to check whether a number is even or odd using the ternary operator, input will be taken from the user.		22	
11.	Write a program to demonstrate in how many ways you can create a string object.		23	
12	Write a program in C# to demonstrate call by value, call by reference, call by output parameter, and call by params		24-27	
13	Write a program in C# to create a class, constructor, and usage of this keyword.		28-29	
14	Write a program in C# to demonstrate static and instance data members for calculating area of circle.		30-31	
15	Write a program in C# to demonstrate inheritance in C# by creating PartTime Emp, FullTime Emp and Emp Class.		32-33	
16	Write a program in C# to demonstrate method Hiding in C# and how to invoke hidden base class members.		34-35	
17	Write a program to demonstrate polymorphism that enables you to invoke derived class methods through base class reference variables at runtime.		36-37	
18	Write a program to create properties in C# having get and set accessor with the following business rule (CA6) • ID should always be non negative numbers • Name cannot be set to null • If student name is missing "No name should be returned" • Passmarks should be read only		38-39	
19	Write a Program in C# to demonstrate the automation of Unit Testing by creating 4 methods and test cases in visual studio test project.		40-41	

[illegible]

Problem Statement 13: Write a program in C# to create a class, constructor and usage of this keyword.

Objective: To learn class constructor and usages of this keyword in C#.

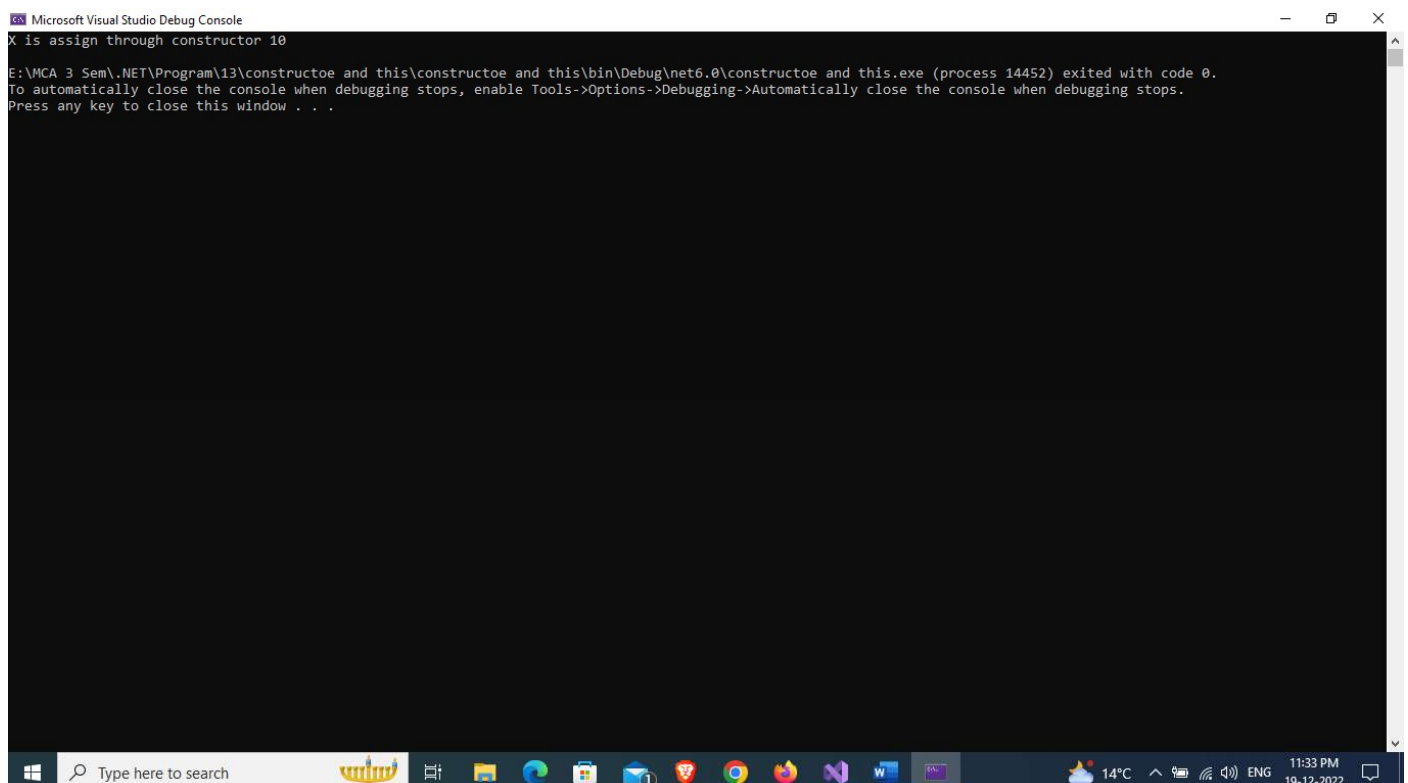
Description: A constructor is a special method that is used to initialize objects. The advantage of a constructor is that it is called when an object of a class is created. The “this” keyword in C# is used to refer to the current instance of the class. It is also used to differentiate between the method parameters and class fields if they both have the same name.

Code constructor:

```
using System;
class Constr
{
    int x;
    public Constr()
    {
        x= 10;
    }

    static void Main(string[] args)
    {
        Constr obj = new Constr();
        Console.WriteLine("X is assign through constructor "+ obj.x);
    }
}
```

Output:



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:

```
X is assign through constructor 10
E:\MCA 3 Sem\NET\Program\13\constructoe and this\constructoe and this\bin\Debug\net6.0\constructoe and this.exe (process 14452) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

The Windows taskbar is visible at the bottom, showing the search bar, task view button, and several application icons. The system tray on the right shows the temperature (14°C), time (11:33 PM), and date (19-12-2022).

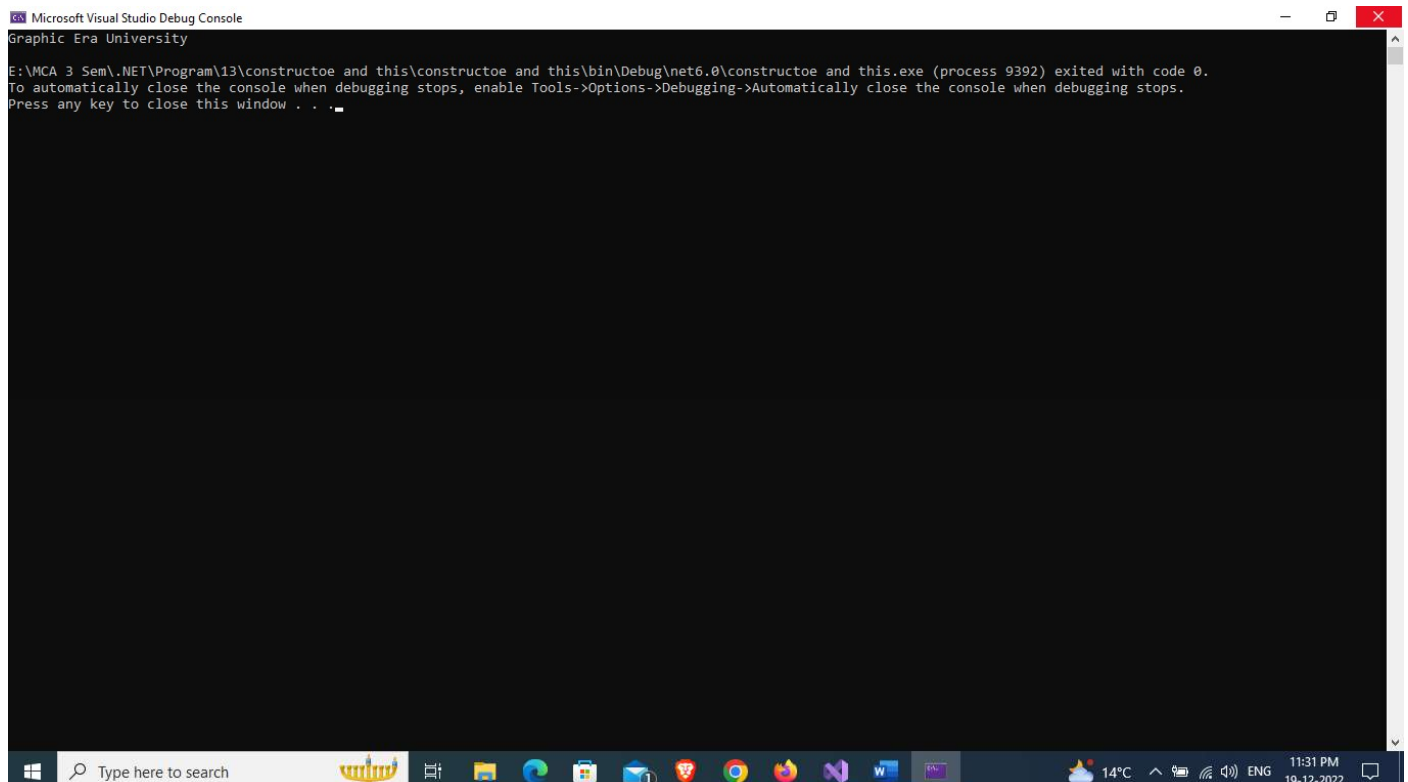
Code using this keyword:

```
using System;
class First
{
    public string Name;

    public string GetName()
    {
        return Name;
    }

    public void SetName(string Name)
    {
        this.Name = Name;
    }
}

class program
{
    public static void Main()
    {
        First obj = new First();
        obj.SetName("Graphic Era University");
        Console.WriteLine(obj.GetName());
    }
}
```

Output:The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output displays "Graphic Era University" on the first line. Below this, a message states: "E:\MCA 3 Sem\NET\Program\13\constructoe and this\constructoe and this\bin\Debug\net6.0\constructoe and this.exe (process 9392) exited with code 0. To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops. Press any key to close this window . . .". The console window is currently empty except for these messages. The Windows taskbar is visible at the bottom, showing the search bar, task view button, and several application icons including File Explorer, Edge, Mail, and others. The system tray on the right shows the temperature as 14°C, the time as 11:31 PM, and the date as 19-12-2022.

Problem Statement 14: Write a program in C# to demonstrate static and instance data members for calculating area of circle.

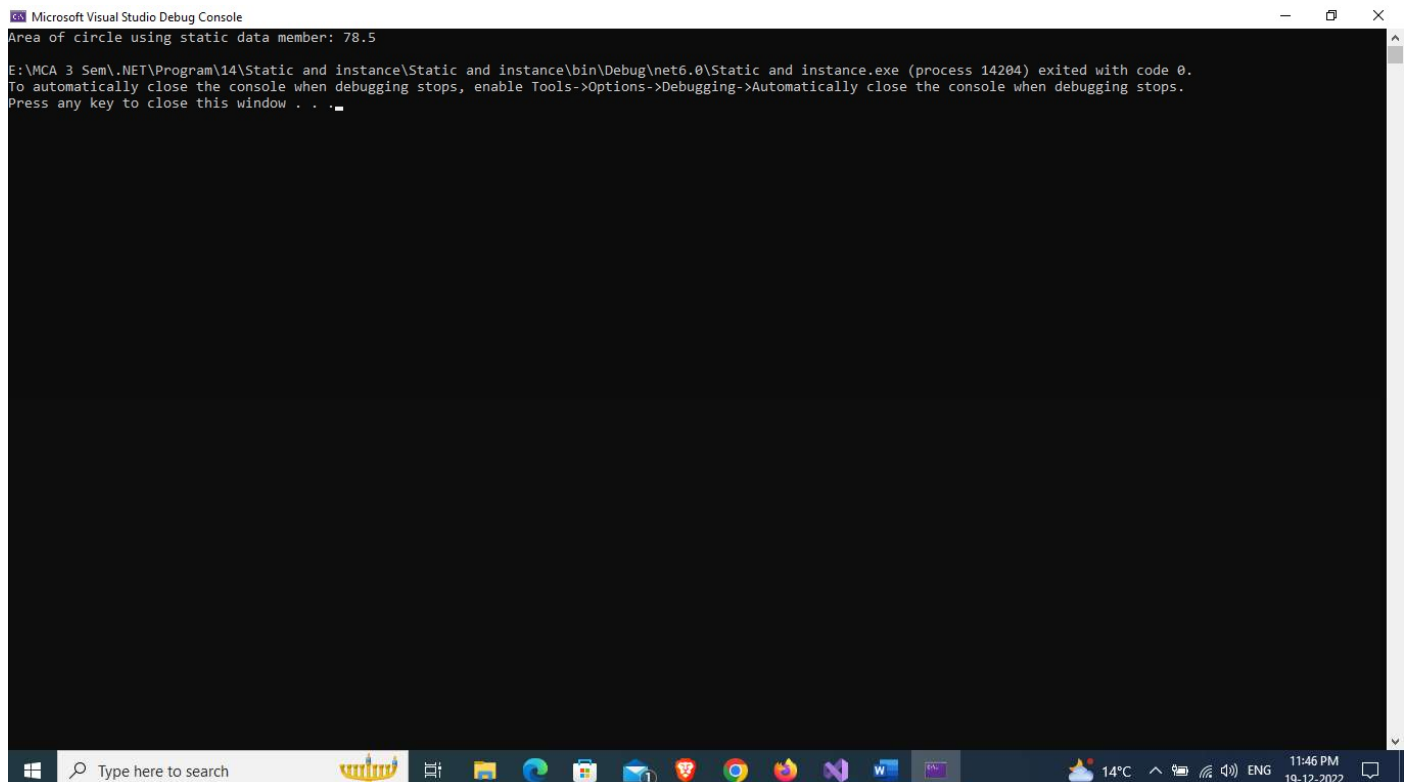
Objective: To learn static and instance data member in C#.

Description: An instance data member of a class is recreated each time when a new instance of the class is created, and it is associated with that instance only. Whereas a static data member of a class is not recreated with new instance creation, only one copy of it is shared by all the instances

Code using static data member:

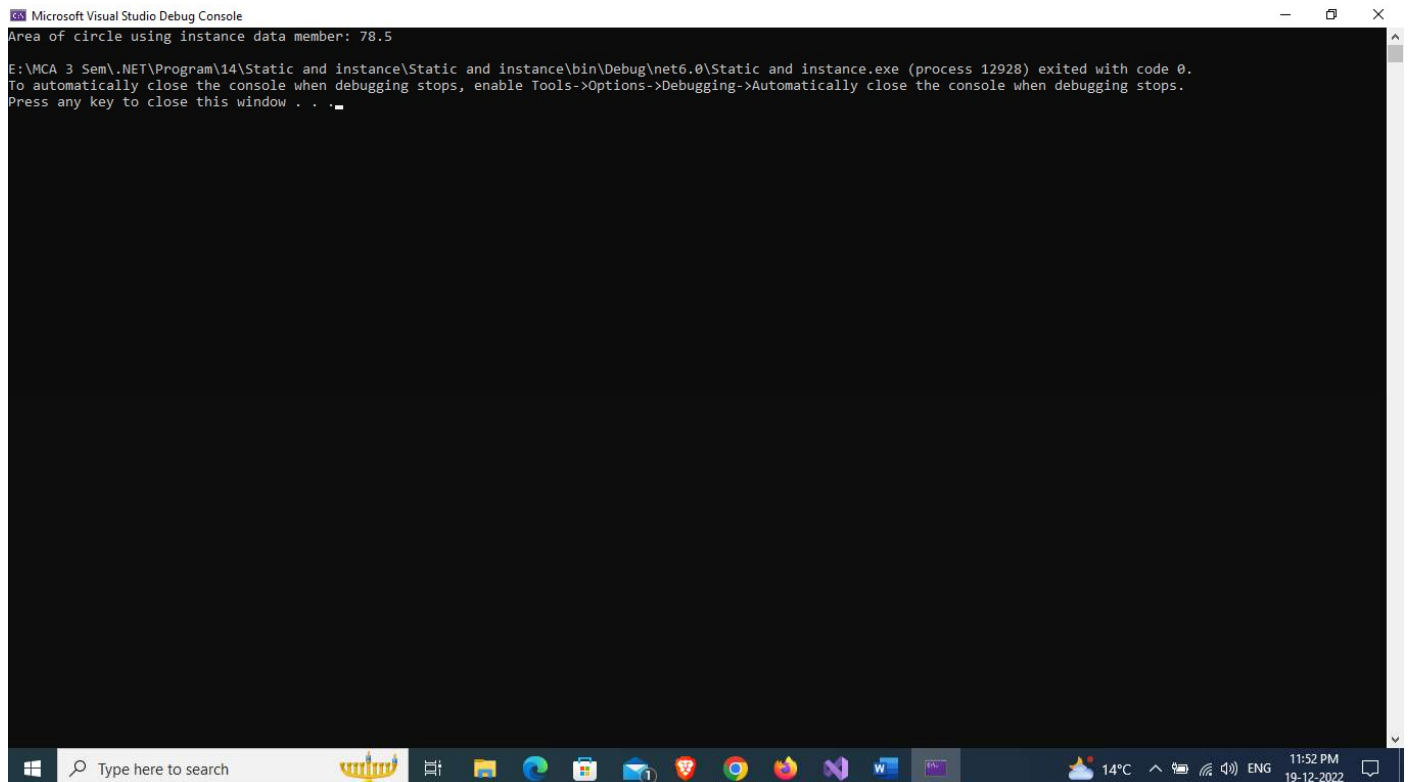
```
using System;
static class First
{
    public static int num = 5;
    public static double area = 3.14 * num * num;
}
public class Final
{
    static public void Main()
    {
        Console.WriteLine("Area of circle using static data member: {0} ", First.area);
    }
}
```

Output:

The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:
Area of circle using static data member: 78.5
E:\MCA 3 Sem\NET\Program\14\Static and instance\Static and instance\bin\Debug\net6.0\Static and instance.exe (process 14204) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
The Windows taskbar is visible at the bottom, showing the search bar, task view, and various application icons. The system tray on the right shows the temperature as 14°C, signal strength icons, and the date and time as 11:46 PM on 19-12-2022.

Code using instance data member:

```
using System;
class First
{
    public int num = 5;
    public double a;
    public void area()
    {
        a = 3.14 * num * num;
        Console.WriteLine("Area of circle using instance data member: {0} ", a);
    }
}
public class Final
{
    static public void Main()
    {
        First fst = new First();
        fst.area();
    }
}
```

Output:The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:
Area of circle using instance data member: 78.5
E:\MCA 3 Sem\NET\Program\14\Static and instance\Static and instance\bin\Debug\net6.0\Static and instance.exe (process 12928) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
The Windows taskbar is visible at the bottom, showing the search bar, task view button, and several application icons including File Explorer, Edge, Mail, and VS Code. The system tray on the right shows the temperature as 14°C, the time as 11:52 PM, and the date as 19-12-2022.

Problem Statement 15: Write a program in C# to demonstrate inheritance in C# by creating PartTime Emp, FullTime Emp and Emp Class.

Objective: To learn inheritance in C#.

Description: An instance data member of a class is recreated each time when a new instance of the class is created, and it is associated with that instance only. Whereas a static data member of a class is not recreated with new instance creation, only one copy of it is shared by all the instances

Code:

```
using System;

class PartTimeEmp
{
    public int PId = 201;
    public string Pname = "Ronaldo";
    public int Psalary = 30000;
}
class FullTimeEmp : PartTimeEmp
{
    public int FId = 101;
    public string Fname = "Messi";
    public int Fsalary = 50000;
}
class Emp
{
    static public void Main(string[] args)
    {
        FullTimeEmp fte = new FullTimeEmp();
        Console.WriteLine("Part Time Employee Details");
        Console.WriteLine("Id: " + fte.PId);
        Console.WriteLine("Name: " + fte.Pname);
        Console.WriteLine("Salary: " + fte.Psalary);
        Console.WriteLine();
        Console.WriteLine("Full Time Employee Details");
        Console.WriteLine("Id: " + fte.FId);
        Console.WriteLine("Name: " + fte.Fname);
        Console.WriteLine("Salary: " + fte.Fsalary);
    }
}
```

Output:

```
Microsoft Visual Studio Debug Console
Part Time Employee Details
Id: 201
Name: Ronaldo
Salary: 30000

Full Time Employee Details
Id: 101
Name: Messi
Salary: 50000

E:\MCA 3 Sem\NET\Program\14\Static and instance\Static and instance\bin\Debug\net6.0\Static and instance.exe (process 12472) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```


Problem Statement 16: Write a program in C# to demonstrate method Hiding in C# and how to invoke hidden base class members.

Objective: To learn method hiding concept in C#.

Description: An instance data member of a class is recreated each time when a new instance of the class is created, and it is associated with that instance only. Whereas a static data member of a class is not recreated with new instance creation, only one copy of it is shared by all the instances.

Code:

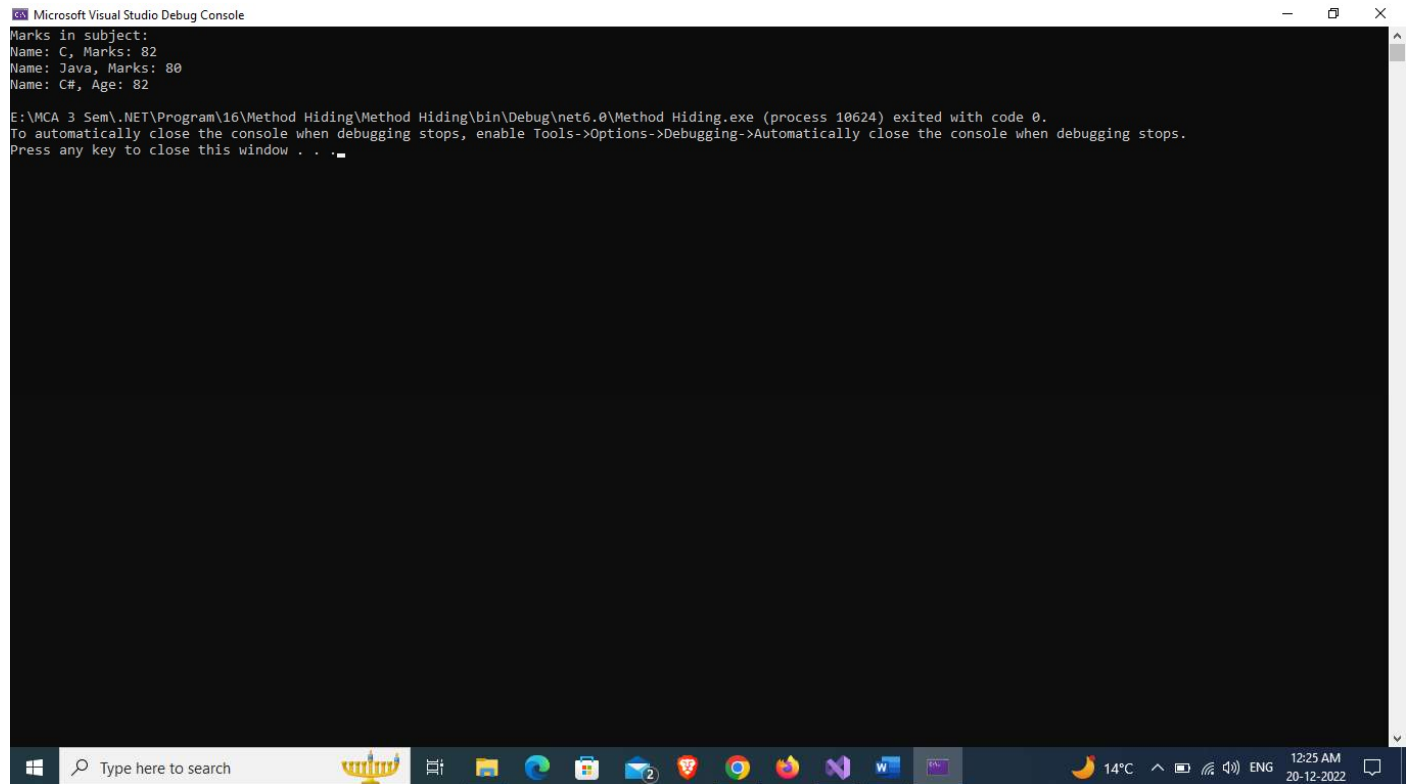
```
using System;
public class MCA
{
    public void subject()
    {
        Console.WriteLine("Marks in subject: ");
    }
}

// Derived Class
public class Marks : MCA
{
    public new void subject()
    {
        base.subject();
        Console.WriteLine("Name: C, Marks: 82 \nName: Java," +
            " Marks: 80 \nName: C#, Age: 82");
    }
}

class GFG
{
    // Main method
    static public void Main()
    {
        // Creating the object of the derived class
        Marks obj = new Marks();

        // Access the method of derived class
        obj.subject();
    }
}
```

Output:



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:

```
Marks in subject:  
Name: C, Marks: 82  
Name: Java, Marks: 80  
Name: C#, Age: 82  
  
E:\MCA 3 Sem\NET\Program\16\Method Hiding\Method Hiding\bin\Debug\net6.0\Method Hiding.exe (process 10624) exited with code 0.  
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.  
Press any key to close this window . . .
```

Below the console window, the Windows taskbar is visible. It includes a search bar with the text "Type here to search", several application icons (including a yellow icon with a red flame), and the system tray area on the right showing the temperature as 14°C, the time as 12:25 AM, and the date as 20-12-2022.

Problem Statement 17: Write a program to demonstrate polymorphism that enables you to invoke derived class methods through base class reference variables at runtime.

Objective: To learn polymorphism concept in C#.

Description: Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

Code:

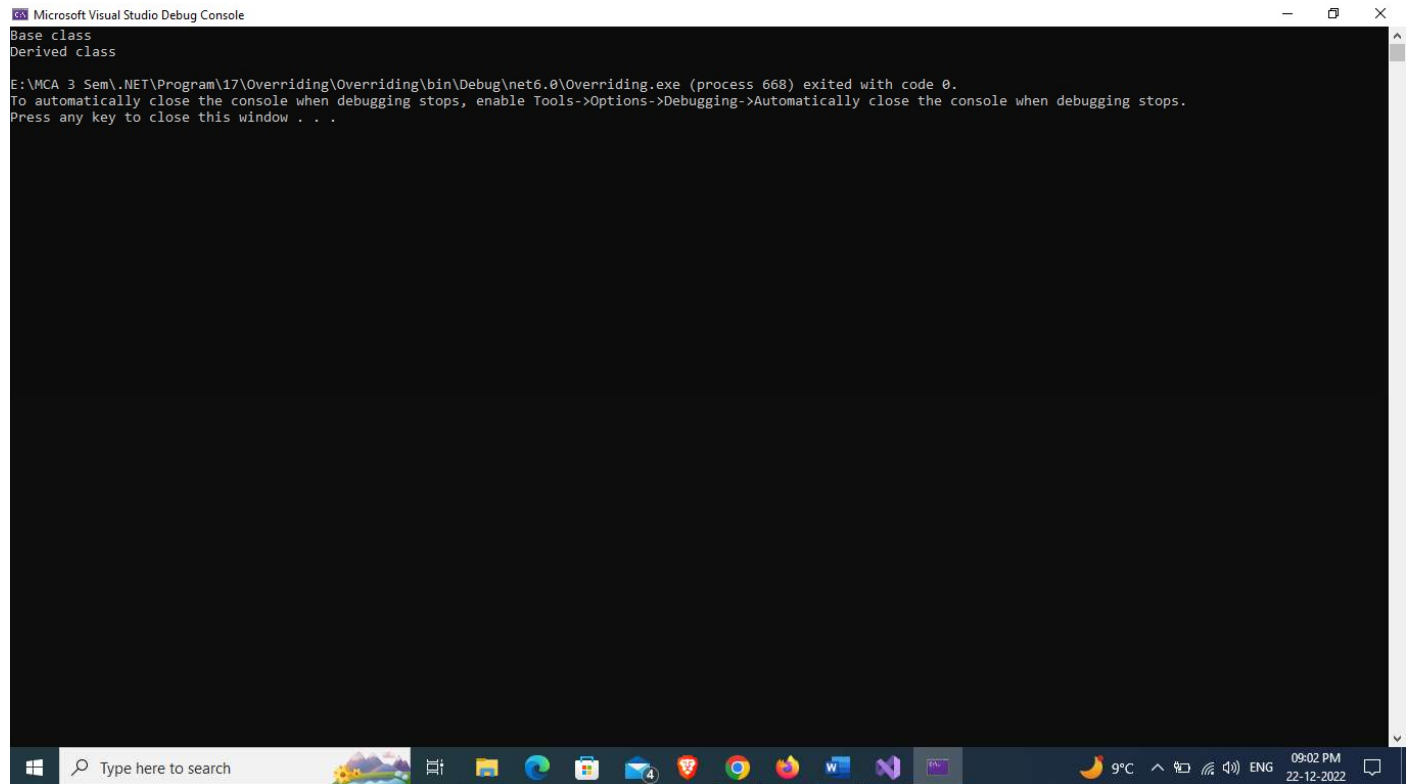
```
using System;

class baseClass
{
    public virtual void show()
    {
        Console.WriteLine("Base class");
    }
}

class derived : baseClass
{
    public override void show()
    {
        Console.WriteLine("Derived class");
    }
}

class Overriding
{
    public static void Main()
    {
        baseClass obj = new baseClass();
        obj.show();
        obj = new derived();
        obj.show();
    }
}
```

Output:



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:

```
Base class
Derived class

E:\MCA 3 Sem\NET\Program\17\Overriding\Overriding\bin\Debug\net6.0\Overriding.exe (process 668) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

The console window is positioned above the Windows taskbar. The taskbar includes a search bar with the text "Type here to search", several application icons (including File Explorer, Edge, Mail, and VS Code), and system tray icons showing a temperature of 9°C, network status, and the time 09:02 PM on 22-12-2022.

Problem Statement 18: Write a program to create properties in C# having get and set accessor with the following business rule (CA6)

- ID should always be non-negative numbers
 - Name cannot be set to null
 - If student name is missing “No name should be returned”
 - Passmarks should be read only
-

Objective: To learn properties concept in C#.

Description: You learned from the previous chapter that private variables can only be accessed within the same class (an outside class has no access to it). However, sometimes we need to access them - and it can be done with properties. A property is like a combination of a variable and a method, and it has two methods: a get and a set method:

Code:

```
public class Student
{
    // Property for the student's ID
    public int ID { get; set; }

    // Property for the student's name
    private string _name;
    public string Name
    {
        get { return _name; }
        set
        {
            if (value == null)
            {
                throw new ArgumentNullException("Name cannot be set to null");
            }
            _name = value;
        }
    }

    // Property for the student's pass marks
    private int _passMarks;
    public int PassMarks
    {
        get { return _passMarks; }
    }

    // Constructor for the Student class
    public Student(int id, string name, int passMarks)
    {
        if (id < 0)
        {
```

```

        throw new ArgumentException("ID must be a non-negative number");
    }
    ID = id;
    Name = name;
    _passMarks = passMarks;
}
static void Main(String[] args)
{
    Student student = new Student(1, null, 50);

    // Set the student's ID
    // student.ID = 2;

    // Set the student's name
    //student.Name = "Jane Doe";

    // Get the student's pass marks
    //int passMarks = student.PassMarks;

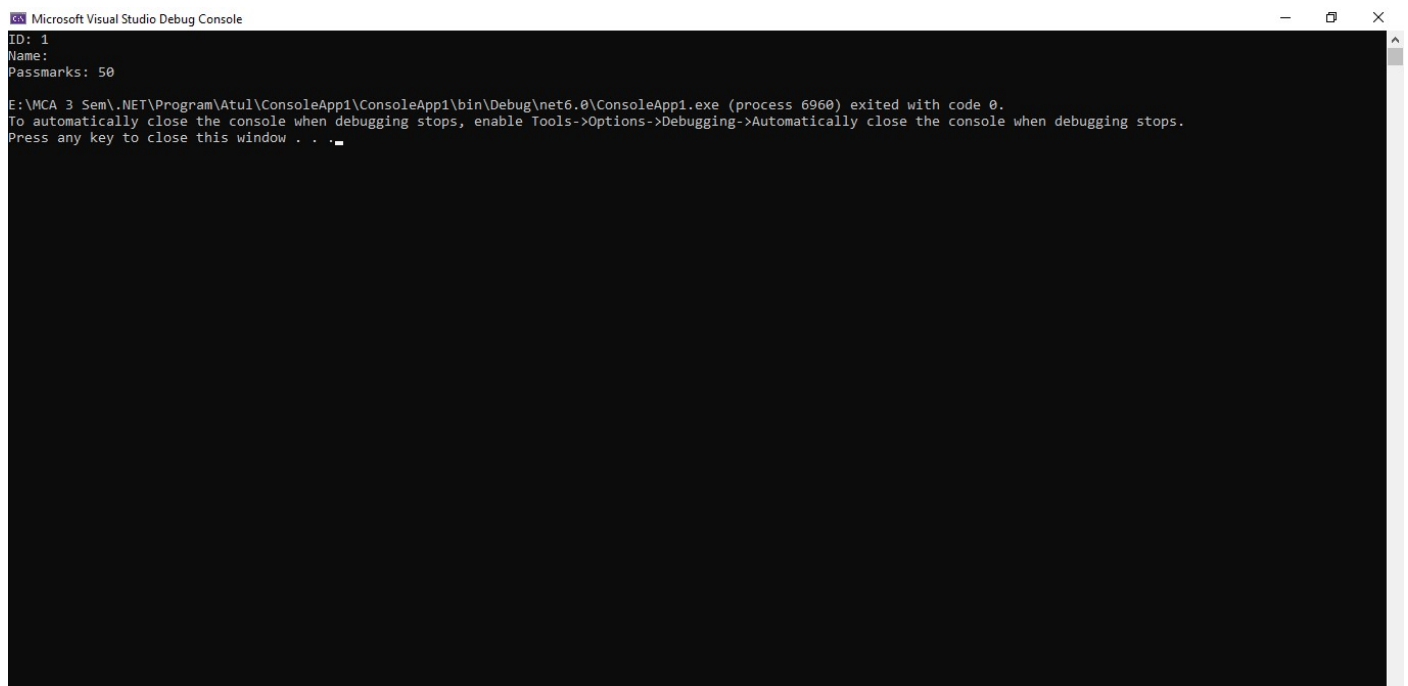
    // Print the student's pass marks
    //Console.WriteLine(passMarks);

    Console.WriteLine("ID: " + student.ID);
    Console.WriteLine("Name: ", student.Name);
    Console.WriteLine("Passmarks: " + student.PassMarks);

}
}

```

Output:



The screenshot shows the Microsoft Visual Studio Debug Console window. The output of the program is displayed as follows:

```

ID: 1
Name:
Passmarks: 50

```

Below the output, a message indicates that the application has exited with code 0 and provides instructions on how to automatically close the console when debugging stops.

```

E:\MCA 3 Sem\NET\Program\Atul\ConsoleApp1\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe (process 6960) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

Problem Statement 19: Write a Program in C# to demonstrate the automation of Unit Testing by creating 4 methods and test cases in visual studio test project.

Objective: To learn how to demonstrate unit testing in C#.

Description: Unit testing breaks the program down into the smallest bit of code, usually function-level, and ensures that the function returns the value one expects. By using a unit testing framework, the unit tests become a separate entity which can then run automated tests on the program as it is being built.

1. Create a project and under this project create a class SumMethodClass. Write a function named Add in that class.

Code:

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

namespace Sum
{
    public class SumMethodClass
    {
        public int Add(int a, int b)
        {
            return a + b;
        }
    }
}
```

2. Create another project and under this project create a unit test file to test above function.

Code:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Sum;
using System;

namespace TestForAdd
{
    [TestClass]
    public class UnitTest1
    {
        SumMethodClass obj = new SumMethodClass();
        [TestMethod]
        public void TestMethod1()
        {
            int a = 5;
            int b = 10;
            int sum = a + b;
            int fail = a + b + 1;
            int ans = obj.Add(a, b);
            Assert.AreEqual(sum, ans);
        }
    }
}
```

```
Assert.AreEqual(fail, ans);
```

```
    }  
}  
}
```

Output:

The screenshot shows the Test Explorer window with a successful test run. The test run finished with 1 test passed, 0 failed, and 0 skipped, taking 857 ms. The test is named 'TestForAdd (1)' and has a duration of 17 ms. The test is expanded, showing a sub-test 'TestForAdd (1)' with a duration of 17 ms, and a sub-test 'UnitTest1 (1)' with a duration of 17 ms. The sub-test 'UnitTest1 (1)' is further expanded, showing a sub-test 'TestMethod1' with a duration of 17 ms. The Group Summary on the right shows 'TestForAdd' with 'Tests in group: 1' and 'Total Duration: 17 ms'. The Outcomes section shows '1 Passed'.

Test	Duration	Traits	Error Message
TestForAdd (1)	17 ms		
TestForAdd (1)	17 ms		
UnitTest1 (1)	17 ms		
TestMethod1	17 ms		

Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 857 ms

Group Summary

TestForAdd

Tests in group: 1

Total Duration: 17 ms

Outcomes

1 Passed

The screenshot shows the Test Explorer window with a failed test run. The test run finished with 1 test failed, 0 passed, and 0 skipped, taking 1.9 sec. The test is named 'TestForAdd (1)' and has a duration of 870 ms. The test is expanded, showing a sub-test 'TestForAdd (1)' with a duration of 870 ms, and a sub-test 'UnitTest1 (1)' with a duration of 870 ms. The sub-test 'UnitTest1 (1)' is further expanded, showing a sub-test 'TestMethod1' with a duration of 870 ms. The error message for 'TestMethod1' is 'Assert.AreEqual failed. Expected:<16>. Actual:<15>'. The Group Summary on the right shows 'TestForAdd' with 'Tests in group: 1' and 'Total Duration: 870 ms'. The Outcomes section shows '1 Failed'.

Test	Duration	Traits	Error Message
TestForAdd (1)	870 ms		
TestForAdd (1)	870 ms		
UnitTest1 (1)	870 ms		
TestMethod1	870 ms		Assert.AreEqual failed. Expected:<16>. Actual:<15>.

Test run finished: 1 Tests (0 Passed, 1 Failed, 0 Skipped) run in 1.9 sec

Group Summary

TestForAdd

Tests in group: 1

Total Duration: 870 ms

Outcomes

1 Failed

Problem Statement 20: Write a Program in C# to demonstrate exception handling in C# by creating two operands and one operator for performing basic mathematical operations using switch case having two conditions:

A) if operator is / and operand2 is 0 then throw divide by zero exception.

B) if selection of operator is wrong then throw an exception of bad operation or invalid operator selected (nesting of exceptions).

Objective: To learn the concept of exception handling in C#.

Description: An exception is a problem that arises during the execution of a program. A C# exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero. Exceptions provide a way to transfer control from one part of a program to another. C# exception handling is built upon four keywords: try, catch, finally, and throw

Code: if operator is / and operand2 is 0 then throw divide by zero exception

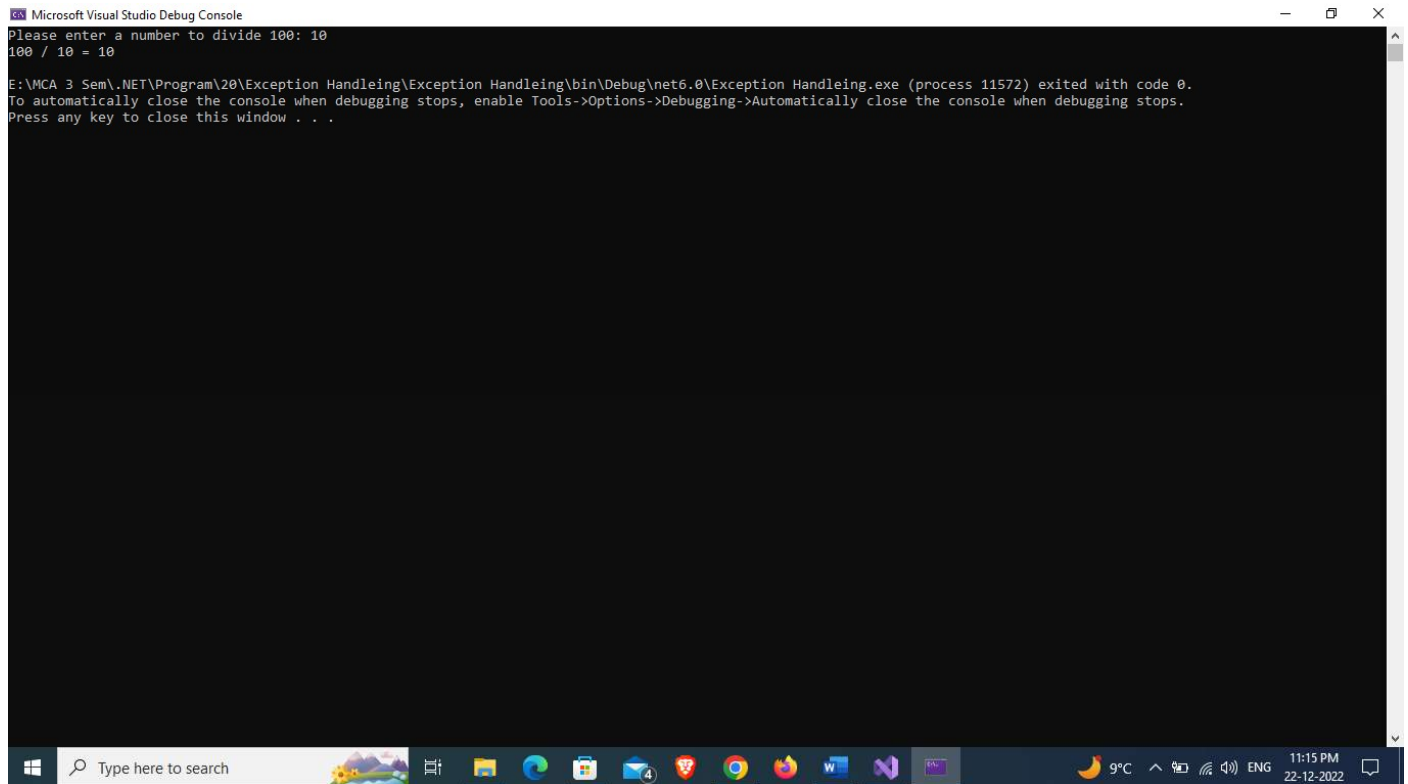
```
using System;
class ExceptionHandling
{
    static void Main(string[] args)
    {
        Console.Write("Please enter a number to divide 100: ");

        try
        {
            int num = int.Parse(Console.ReadLine());

            int result = 100 / num;

            Console.WriteLine("100 / {0} = {1}", num, result);
        }
        catch (DivideByZeroException ex)
        {
            Console.Write("Cannot divide by zero. Please try again." + ex);
        }
    }
}
```

Output:

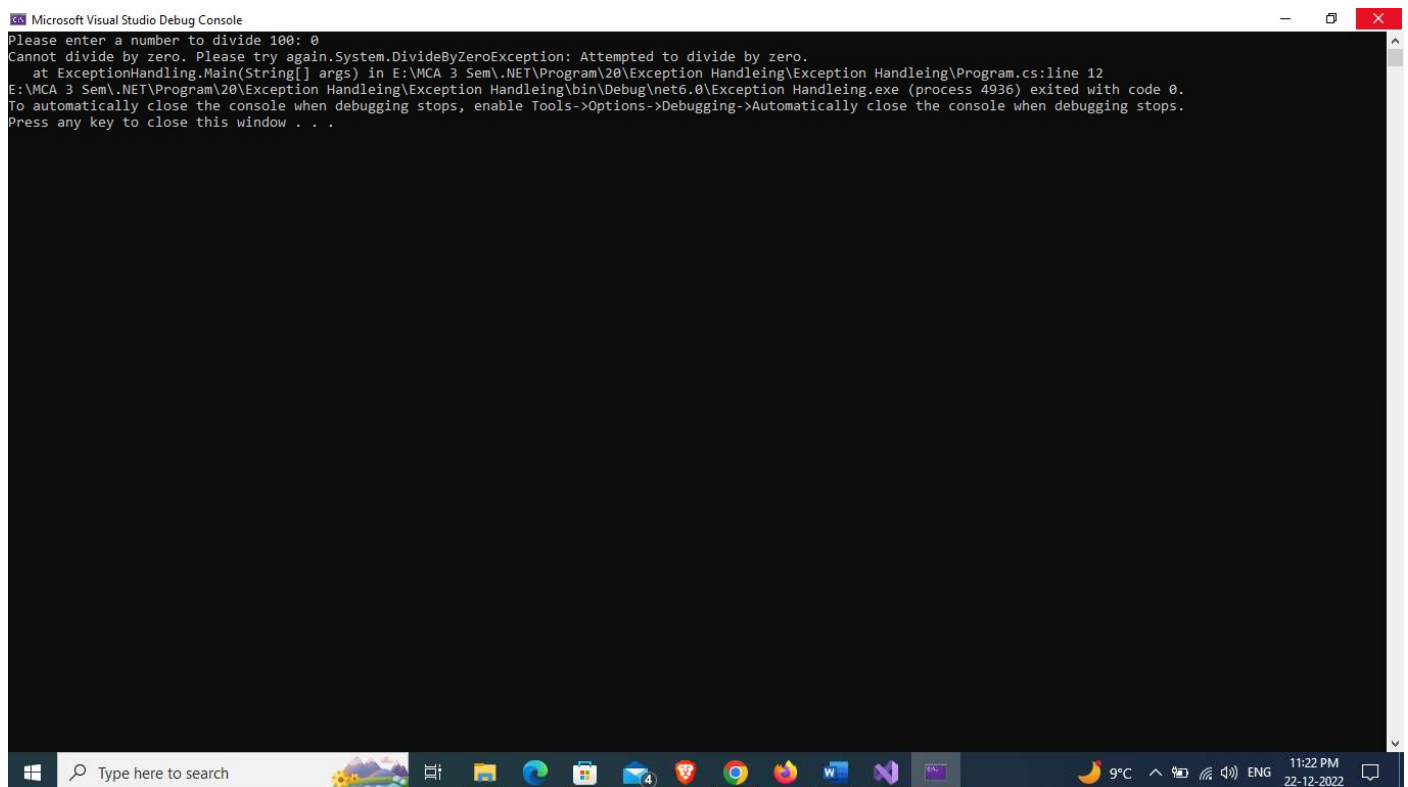


Microsoft Visual Studio Debug Console

```
Please enter a number to divide 100: 10
100 / 10 = 10

E:\MCA 3 Sem\NET\Program\20\Exception Handling\Exception Handling\bin\Debug\net6.0\Exception Handling.exe (process 11572) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

The screenshot shows the Visual Studio Debug Console window. The user has entered '10' as the divisor for 100. The output shows '100 / 10 = 10'. The application has exited with code 0. The Windows taskbar at the bottom shows the time as 11:15 PM on 22-12-2022.



Microsoft Visual Studio Debug Console

```
Please enter a number to divide 100: 0
Cannot divide by zero. Please try again.System.DivideByZeroException: Attempted to divide by zero.
   at ExceptionHandling.Main(String[] args) in E:\MCA 3 Sem\NET\Program\20\Exception Handling\Exception Handling\Program.cs:line 12
E:\MCA 3 Sem\NET\Program\20\Exception Handling\Exception Handling\bin\Debug\net6.0\Exception Handling.exe (process 4936) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

The screenshot shows the Visual Studio Debug Console window. The user has entered '0' as the divisor for 100. The output shows an error: 'Cannot divide by zero. Please try again.System.DivideByZeroException: Attempted to divide by zero.' The application has exited with code 0. The Windows taskbar at the bottom shows the time as 11:22 PM on 22-12-2022.

Code: if selection of operator is wrong then throw an exception of bad operation or invalid operator selected (nesting of exceptions).

```
using System;
namespace CustomExceptionHandling
{
    class CustomException
    {
        static void Main(string[] args)
        {
            try
            {
                string s = OperatCheck();
                Console.WriteLine(s);
                Console.ReadLine();
            }
            catch (Operat ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        static string OperatCheck()
        {
            Console.Write("Enter Operator: ");
            String s = Console.ReadLine();
            if (s != "+" || s != "-" || s != "*" || s != "/" || s != "%")
            {
                throw new Exception("Operator not valid for operate two integer");
            }
            return s;
        }
    }
    public class Operat : Exception
    {
        public Operat(String s):base(s)
        {
        }
    }
}
```

Output:

Enter Operator: 6

Unhandled Exception:

System.Exception: Operator not valid for operate two integer

at CustomExceptionHandler.CustomException.OperatCheck () [0x00059] in
<c46a10cf33bf4e3f83469dea6b521260>:0

at CustomExceptionHandler.CustomException.Main (System.String[] args) [0x00002] in
<c46a10cf33bf4e3f83469dea6b521260>:0

[ERROR] FATAL UNHANDLED EXCEPTION: System.Exception: Operator not valid for operate two
integer

at CustomExceptionHandler.CustomException.OperatCheck () [0x00059] in
<c46a10cf33bf4e3f83469dea6b521260>:0

at CustomExceptionHandler.CustomException.Main (System.String[] args) [0x00002] in
<c46a10cf33bf4e3f83469dea6b521260>:0

Problem Statement 21 (A): Create Master Page and Design pages with ASP.NET controls, Styling sites with ASP.NET themes:

Objective: To learn the single master page concept in ASP.NET.

Description: A master page is an ASP.NET file with the extension. master (for example, MySite. master) with a predefined layout that can include static text, HTML elements, and server controls. The master page is identified by a special @ Master directive that replaces the @ Page directive that is used for ordinary .aspx pages.

Code: Master Page

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Graphic Era Home Page</title>
    <link rel="stylesheet" type="text/css" href="css/my.css"/>
</head>
<body>
<header id="header">
<h1>Graphic Era Deemed to be University</h1>
</header>
<nav id="nav">
    <ul>
        <li><a href="home.aspx">Home</a></li>
        <li><a href="About.aspx">About</a></li>
        <li><a href="Article/Article.aspx">Article</a></li>
        <li><a href="#">Contact</a></li>
    </ul>
</nav>
<aside id="side">
    <h1>news</h1>
    <a href="#"><p>how to create website</p></a>
    <a href="#"><p>how to create master pages</p></a>
    <a href="#">walkthrough ASP.NET</a>
</aside>

<div id="con">
    <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
    </asp:ContentPlaceHolder>
</div>

<footer id="footer">
```

```

        copyright @Graphic Era Deemed to be University</footer>
</body>
</html>
    <form id="form1" runat="server">

    </form>

```

Code: Home Page

```

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

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <h1>Home page</h1>
</asp:Content>

```

Code: About us

```

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

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <h1>About - Graphic Era (Deemed to be University), Dehradun</h1>
    <p>

```

In 1993 a young man with just twenty nine thousand and loads of determination embarked on a mission to transform the higher education landscape of the Doon valley. Graphic Era Deemed to be University is the culmination of the vision of its founder, Prof (Dr) Kamal Ghanshala, who had the dream to change the destiny of thousands of youth, through quality and holistic education and his vision took a concrete shape in the form of Graphic Era Institute in 1996.

In 2008, the Institute was accorded the status of Deemed University under Section 3 of the UGC Act, 1956 vide Notification F.9-48/2007-U.3 (A) dated August 14, 2008 and approved by Ministry of Human Resource Development, Government of India. In 2015 Graphic Era University was accredited by NAAC with grade 'A'.

University has acquired transnational dimensions through student exchange and knowledge sharing programs with many foreign universities and has been acclaimed and honored at international forums for its brilliance in upholding the highest standards of education. With recognition from prestigious institutions, Graphic Era University is setting new benchmarks in education. It has taken big initiative in engineering programs by getting into Partnerships with Tata Technologies and IBM to create next age Engineering Professionals through Industry Collaborations. Graphic Era hosts Technology Business Incubator that provides support for technology-based entrepreneurship. At present, Graphic Era has innumerable students on its rolls, pursuing education in different disciplines, ranging from engineering, science, business, management, commerce, hospitality to humanities and social sciences. The alumni of Graphic Era can be encountered worldwide in marquee brands like Apple, Google, Microsoft, HSBC, to name a few and in the service of the nation in all wings of the Armed forces.

Graphic Era (Deemed to be University) has been conferred All-India NIRF Rank 89 in the Engineering Category, thus being accorded as the highest-ranked Engineering University in Uttarakhand, after IIT Roorkee, in the MHRD NIRF (National Institutional Ranking Framework) Rankings in June 2020. The University has also been awarded All India Rank 97 in the University Category, establishing GEU as the highest-ranked university in Uttarakhand amongst all Government and Private universities in the state.

</p>
</asp:Content>

Code: CSS File

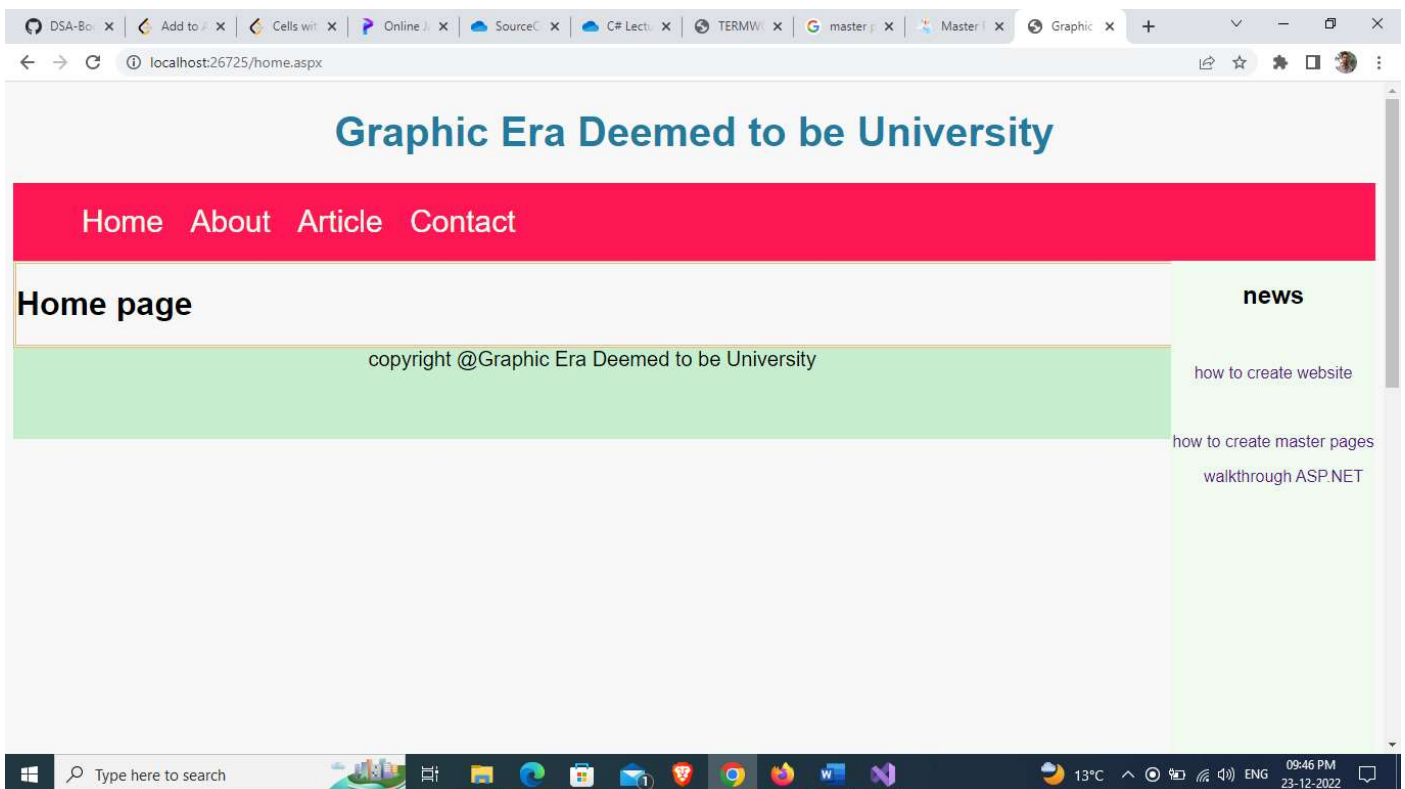
```
#header{
    color: #247BA0;
    text-align: center;
    font-size: 20px;
}
#nav{
    background-color:#FF1654;
    padding: 5px;
}
ul{
    list-style-type: none;
}
li a {
    color: #F1FAEE;
    font-size: 30px;
    column-width: 5%;
}
li
{
    display: inline;
    padding-left: 2px;
    column-width: 20px;
}
a{
text-decoration: none;
margin-left:20px
}
li a:hover{
    background-color: #F3FFBD;
    color: #FF1654;
    padding:1%;
}
#side{
    text-align: center;
    float: right;
    width: 15%;
    padding-bottom: 79%;
    background-color: #F1FAEE;
```

```

}
#article{
    background-color: #EEF5DB;
    padding: 10px;
    padding-bottom: 75%;
}
#footer{
    background-color: #C7EFCF;
    text-align:center;
    padding-bottom: 5%;
    font-size: 20px;
}
#con{
    border:double;
    border-color:burlywood;
}

```

Output:



Problem Statement 21 (B): Create Master Page and Design pages with ASP.NET controls, Styling sites with ASP.NET themes:

Objective: To learn the multiple master page concept in ASP.NET.

Description: A master page is an ASP.NET file with the extension. master (for example, MySite. master) with a predefined layout that can include static text, HTML elements, and server controls. The master page is identified by a special @ Master directive that replaces the @ Page directive that is used for ordinary .aspx pages.

Code: First Master Page

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

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
```

```
  <link href="style.css" type="text/css" rel="stylesheet" />
```

```
</head>
```

```
<body>
```

```
  <form id="form1" runat="server">
```

```
    <div id="header">
```

```
      <div id="heimg"></div>
```

```
    </div>
```

```
    <div id="menu">
```

```
      <ul>
```

```
        <li><a href="Home.aspx">Home</a></li>
```

```
        <li><a href="technology.aspx">Technology</a></li>
```

```
        <li><a href="#">Recent Posts</a></li>
```

```
        <li><a href="#">Abouts</a></li>
```

```
        <li><a href="#">Login</a></li>
```

```
      </ul>
```

```
    </div>
```

```
  <div id="center">
```

```
    <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
```

```
  </asp:ContentPlaceHolder>
```

```
</div>
```

```
  <div id="footer">
```

```

        <h5>copyrights all reserved Graphic Era 2020</h5>
    </div>

</form>
</body>
</html>

```

Code: Second Master Page

```

<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Secondmasterpage.master.cs"
Inherits="MasterDemo.Secondmasterpage" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="style.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <form id="form1" runat="server">

        <div id="first">

            <h1>ASP.NET</h1>
        </div>
        <div id="menuone">
            <ul>
                <li><a href="Home.aspx">Home</a></li>
                <li><a href="#">Technology</a></li>
                <li><a href="#">Article</a></li>
                <li><a href="#">Blogs</a></li>
                <li><a href="#">Abouts</a></li>
                <li><a href="#">Login</a></li>
            </ul>
        </div>
        <div id="second">
            <h3>Welcome to ASP.NET page!!!!!!!!!!</h3>

            <div id="imageset"></div>
            <div id="data">
                <h3>C-sharp</h3>
                <h3>ASP.NET</h3>
                <h3>XAMARIN</h3>
                <h3>Android</h3>
                <h3>VisualStudio 2019</h3>

            </div>
        </div>
    </form>

```

```

<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

</asp:ContentPlaceHolder>
</div>

</div>
<div id="footer">
  <h5>copyrights all reserved Graphic Era 2020</h5>
</div>

</form>
</body>
</html>

```

Code: Home Page

```

<%@ Page Title="" Language="C#" MasterPageFile="~/Firstmasterpage.Master"
AutoEventWireup="true" CodeBehind="Home.aspx.cs" Inherits="MasterDemo.Home" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <h1>Home</h1>
  <p>Your homepage is an essential tool for your business and often serves as the first impression
to potential customers. There are many important components of effective web design, like white
space, font selection, color schemes, and layout, but the core of a website is its content, not its
design.</p>
</asp:Content>

```

Code: Technology.aspx

```

<%@ Page Title="" Language="C#" MasterPageFile="~/Secondmasterpage.Master"
AutoEventWireup="true" CodeBehind="Technology.aspx.cs" Inherits="MasterDemo.Technology"
%>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <h1>Technology</h1>
  <p>A version of the image is everywhere: a family is gathered together—at the kitchen table, in
the living room, in the car—and each face is aglow in the light of a screen. There is, of course, a name
for it. Technoference refers to the interruptions in interpersonal communication caused by attention
paid to personal technological devices. In other words, it's that thing where you're looking at your
phone or tablet and don't hear the question your kid or your friend or your mom or your boyfriend
asked you. Even though you were totally listening.</p>
</asp:Content>

```

Code: Style.css

```
body {
    background-color: cadetblue;
}

#header {
    background-color: chartreuse;
    border-image-width: 50px;
    text-align: center;
    width: 1000px;
    height: 70px;
    margin-left: auto;
    margin-right: auto;
}

#heimg {
    background-image: url("photo/csssss.jpg");
    width: 510px;
    height: 50px;
    margin-left: auto;
    margin-right: auto;
    background-repeat: no-repeat;
}

#heimgone {
    background-image: url("photo/csssss.jpg");
    width: 510px;
    height: 50px;
    margin-left: auto;
    margin-right: auto;
}

#menu {
    background-color: yellow;
    width: 1000px;
    height: 50px;
    margin-left: auto;
    margin-right: auto;
}

ul {
    list-style-type: none;
}

li a {
    background-color: coral;
    font-size: 40px;
}
```

```
li {
    display: inline;
    padding-left: 3px;
    column-width: 15px;
}

a {
    text-decoration: none;
    margin-left: auto;
}

li a:hover {
    background-color: crimson;
    padding: 1px;
}

#center {
    background-color: white;
    text-align: center;
    width: 1000px;
    height: 700px;
    margin-left: auto;
    margin-right: auto;
}

#img {
    background-image: url("photo/csssss.jpg");
    width: 700px;
    height: 50px;
    margin-left: auto;
    margin-right: auto;
}

#footer {
    background-color: brown;
    width: 1000px;
    height: 30px;
    margin-left: auto;
    margin-right: auto;
}

#footer {
    text-align: center;
}

#first {
    background-color: burlywood;
    text-align: center;
    width: 1000px;
```

```
height: 70px;
margin-left: auto;
margin-right: auto;
}

#menuone {
background-color: burlywood;
width: 1000px;
height: 40px;
margin-left: auto;
margin-right: auto;
}

#second {
background-color: antiquewhite;
text-align: center;
width: 1000px;
height: 700px;
margin-left: auto;
margin-right: auto;
}

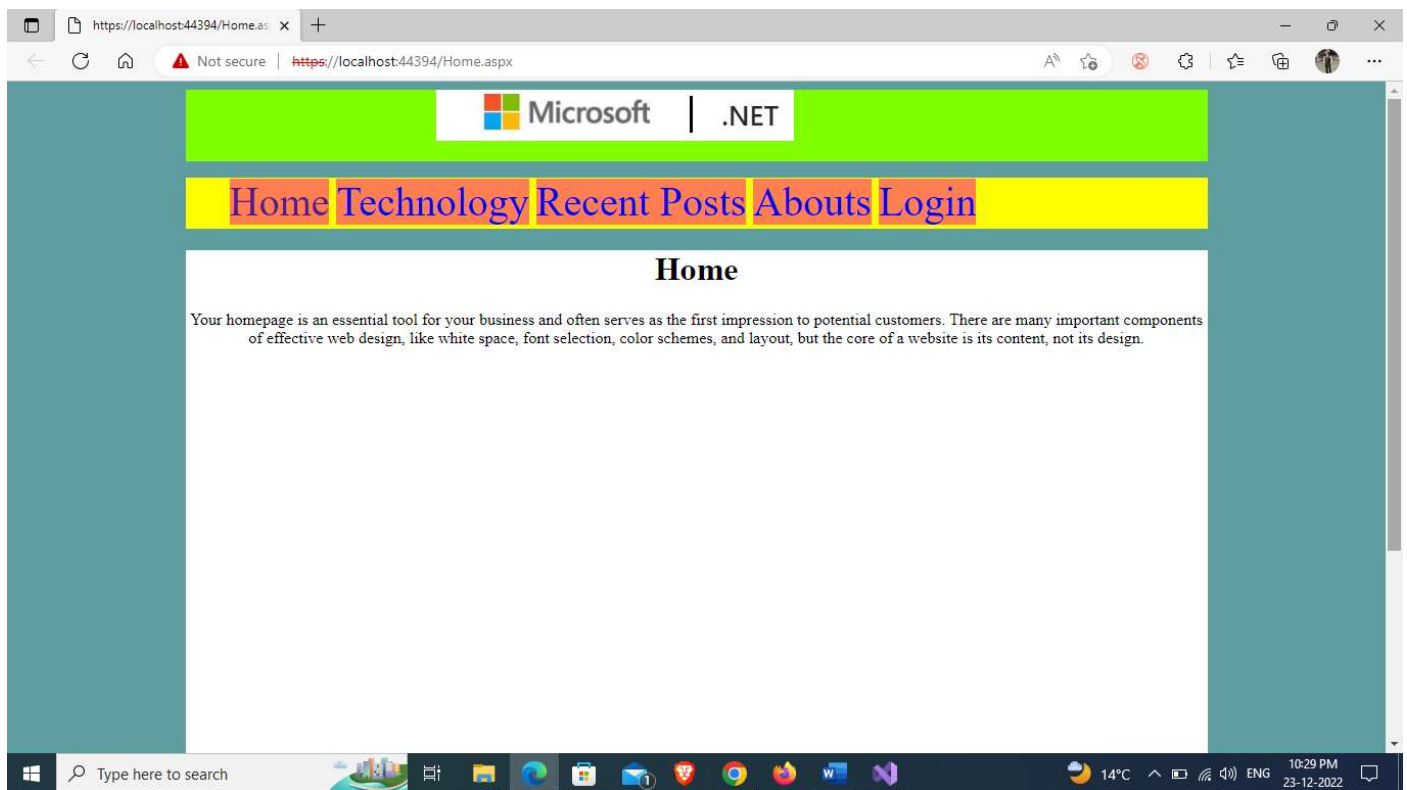
#imageset {
background-image: url("photo/unnamed.png");
width: 300px;
height: 300px;
margin-left: auto;
margin-right: auto;
background-repeat: no-repeat;
}

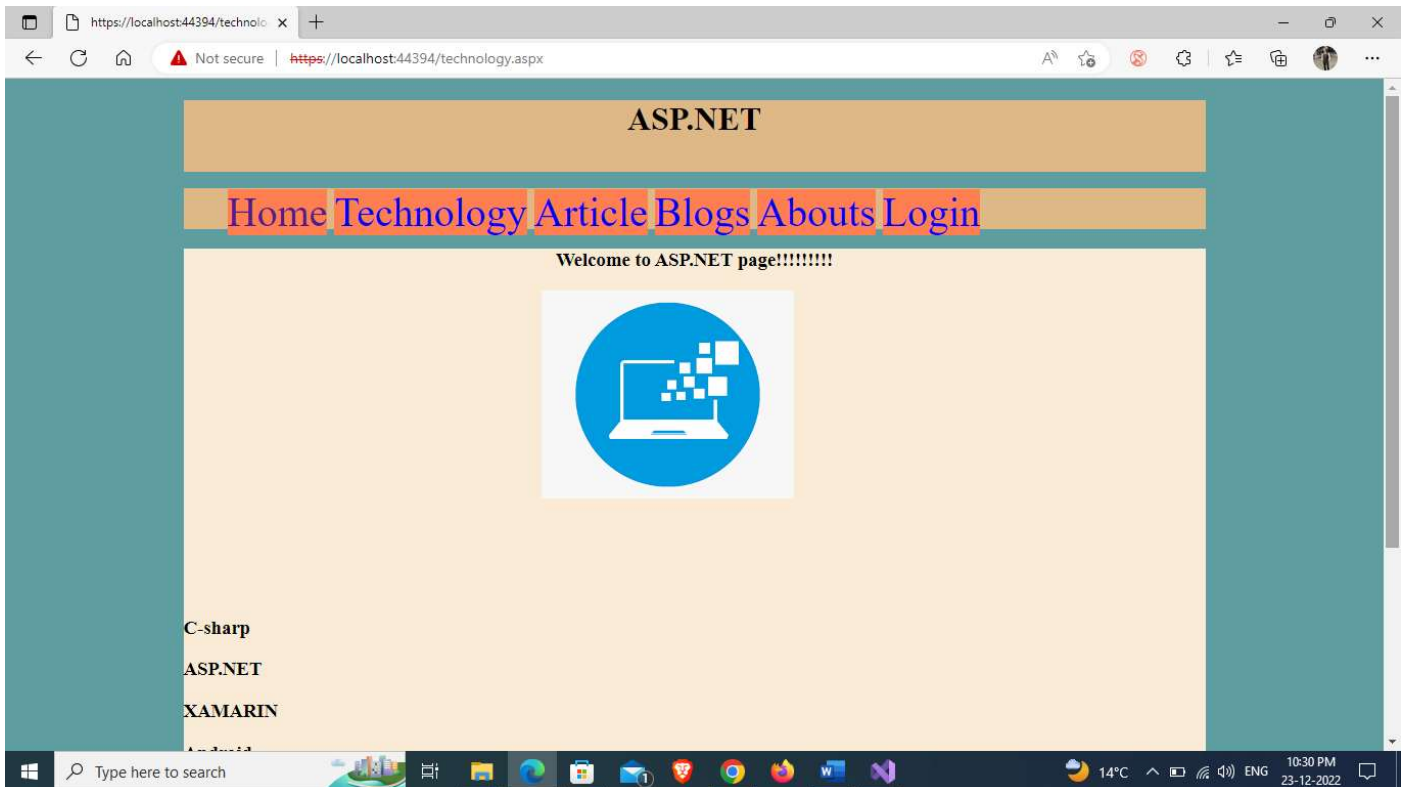
#last {
background-color: burlywood;
text-align: center;
width: 1000px;
height: 30px;
margin-left: auto;
margin-right: auto;
}

#imgone {
background-image: url("photo/unnamed.png");
width: 300px;
height: 300px;
margin-left: auto;
margin-right: auto;
background-repeat: no-repeat;
}
```

```
#ceimg {  
    background-image: url("photo/unnamed.png");  
    width: 300px;  
    height: 300px;  
    margin-left: auto;  
    margin-right: auto;  
    background-repeat: no-repeat;  
}  
  
#data {  
    text-align: left;  
}
```

Output:





Problem Statement 22: Create a registration form and show the working of all the validation controls used in it.

Objective: To learn the implementation of form validation concept in ASP.NET.

Description: An important aspect of creating ASP.NET Web pages for user input is to be able to check that the information users enter is valid. ASP.NET provides a set of validation controls that provide an easy-to-use but powerful way to check for errors and, if necessary, display messages to the user.

Code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Register.aspx.cs"
Inherits="ValidationControlDemo1.Register" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <table style="width:100%;">
                <tr>
                    <td>Name</td>
                    <td>
                        <asp:TextBox ID="txtname" runat="server"></asp:TextBox>
                    </td>
                    <td>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="txtname" ForeColor="Red" ErrorMessage="Please Enter User
Name"></asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td>Password</td>
                    <td>
                        <asp:TextBox ID="txtpass" runat="server" TextMode="Password"></asp:TextBox>
                    </td>
                    <td>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ForeColor="Red" ControlToValidate="txtpass" ErrorMessage="password is
required"></asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td>ReEnter Password</td>
                    <td>
```

```

        <asp:TextBox ID="txtrepass" TextMode="Password" runat="server"></asp:TextBox>
    </td>
    <td>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
        ForeColor="Red" ControlToValidate="txtrepass" ErrorMessage="re enter password is
        required"></asp:RequiredFieldValidator>
        <asp:CompareValidator ID="CompareValidator1" runat="server" ForeColor="Yellow"
        ControlToCompare="txtpass" ControlToValidate="txtrepass" ErrorMessage="password and re enter
        password doesnt match"></asp:CompareValidator>
    </td>
</tr>
<tr>
    <td>Email</td>
    <td>
        <asp:TextBox ID="txtemail" runat="server"></asp:TextBox>
    </td>
    <td>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
        ControlToValidate="txtemail" ForeColor="Red" ErrorMessage="email is
        required"></asp:RequiredFieldValidator>
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
        ForeColor="Green" ControlToValidate="txtemail" ValidationExpression="\w+([-.\']\w+)*@\w+([-
        .]\w+)*\.\w+([-.\']\w+)*" ErrorMessage="email format is
        incorrect"></asp:RegularExpressionValidator>
    </td>
</tr>
<tr>
    <td>Age</td>
    <td>
        <asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
    </td>
    <td>
        <asp:RangeValidator ID="RangeValidator1" Type="Integer" runat="server"
        ForeColor="Pink" ControlToValidate="txtAge" MinimumValue="1" MaximumValue="120"
        ErrorMessage="enter valid age"></asp:RangeValidator>
    </td>
</tr>
<tr>
    <td>Mobile No</td>
    <td>
        <asp:TextBox ID="txtPhoneNumber" runat="server"></asp:TextBox>
    </td>
    <td>
        <asp:CustomValidator ID="CustomValidator1" Display="Dynamic" runat="server"
        ErrorMessage="CustomValidator"
        OnServerValidate="CustomValidator1_ServerValidate"></asp:CustomValidator>
    </td>
</tr>
<tr>
    <td>

```

```

        <asp:ValidationSummary ID="ValidationSummary2" ShowMessageBox="true"
runat="server" />
    </td>
    <td>
        <asp:Button ID="btnsubmit" CausesValidation="true" runat="server" Text="Submit" />
    </td>
    <td>
        <asp:ValidationSummary ID="ValidationSummary1" runat="server" />
    </td>
</tr>
</table>
</div>
</form>
</body>
</html>

```

Output:

The screenshot shows a web browser window with the URL `https://localhost:44351/Register.aspx`. The page contains a registration form with the following fields: Name, Password, ReEnter Password, Email, Age, and Mobile No. Below the form fields is a 'Submit' button. A modal dialog box is displayed in the center of the screen, titled 'localhost:44351 says'. The dialog box contains the following error messages:

- Please Enter User Name
- password is required
- re enter password is required
- email is required

The 'OK' button is visible in the bottom right corner of the dialog box.

Problem Statement 23: Demonstrate how to render user control in web form.

Objective: To learn the implementation of form validation concept in ASP.NET.

Description: C# user control is defined as an implementation in programming language of C# to provide an empty control and this control can be leveraged to create other controls. This implementation provides additional flexibility to re-use controls in a large-scale web project.

Code: UserControl.ascx

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

<%@ Register src="NameUserControl.ascx" tagname="NameUserControl" tagprefix="uc1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <uc1:NameUserControl ID="NameUserControl1" runat="server" />
        </div>
    </form>
</body>
</html>
```

Code: Home.aspx

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

<%@ Register src="NameUserControl.ascx" tagname="NameUserControl" tagprefix="uc1" %>

<!DOCTYPE html>

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

```
        <uc1:NameUserControl ID="NameUserControl1" runat="server" />
    </div>
</form>
</body>
</html>
```

Code: Test.aspx

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

<%@ Register src="NameUserControl.ascx" tagname="NameUserControl" tagprefix="uc1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <uc1:NameUserControl ID="NameUserControl1" runat="server" />
        </div>
    </form>
</body>
</html>
```

Output:

First Name

Last Name

Problem Statement 24: Use ADO.NET to

- a. Establish connection and create a table
 - b. Insert Data into the Table
 - c. Retrieve Record
 - d. Deleting Record
-

Objective: To learn the implementation of ADO.NET in C#.

Description: ADO.NET is a data access technology from the Microsoft .NET Framework that provides communication between relational and non-relational systems through a common set of components. ADO.NET is a set of computer software components that programmers can use to access data and data services from a database.

Code:

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

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            dataGridView1.Visible = false;
        }
        private SqlConnection getConnection()
        {
            string constr = @"Data Source=(localdb)\mssqllocaldb;Initial Catalog=TestDB;Integrated
Security=True";
            SqlConnection con = null;
            try
            {
                con = new SqlConnection(constr);
                con.Open();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```

```

    return con;
}

private void button1_Click(object sender, EventArgs e)
{
    SqlConnection con = getConnection();
    string query = @"CREATE TABLE dbo.UserTable
    (
        ID int IDENTITY(1,1) NOT NULL,
        Name nvarchar(50) NULL,
        Email nvarchar(100) NULL,
        CONSTRAINT pk_id PRIMARY KEY (ID)
    );";
    SqlCommand cmd = new SqlCommand(query, con);
    try
    {
        cmd.ExecuteNonQuery();
        label3.Text="Table Created.";
    }
    catch(Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally
    {
        con.Close();
    }
}

private void button2_Click(object sender, EventArgs e)
{
    string name = textBox1.Text;
    string email = textBox2.Text;
    SqlConnection con = getConnection();
    string query = @"INSERT INTO UserTable (Name,Email) VALUES (@Name,@Email)";
    SqlCommand cmd = new SqlCommand(query, con);

    //cmd.Parameters.AddWithValue("@ID", id);
    cmd.Parameters.AddWithValue("@Name", name);
    cmd.Parameters.AddWithValue("@Email", email);
    try
    {
        cmd.ExecuteNonQuery();
        label3.Text = "Inserted Into Table Successfully.";
    }catch(Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

```

```

        finally
        {
            con.Close();
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        SqlConnection con = getConnection();

        string query = "SELECT * FROM UserTable";

        SqlCommand cmd = new SqlCommand(query, con);
        try
        {
            SqlDataAdapter sdt = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            sdt.Fill(dt);
            dataGridView1.DataSource = dt;
            dataGridView1.Visible = true;
        }
        catch(Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        finally
        {
            con.Close();
        }
    }

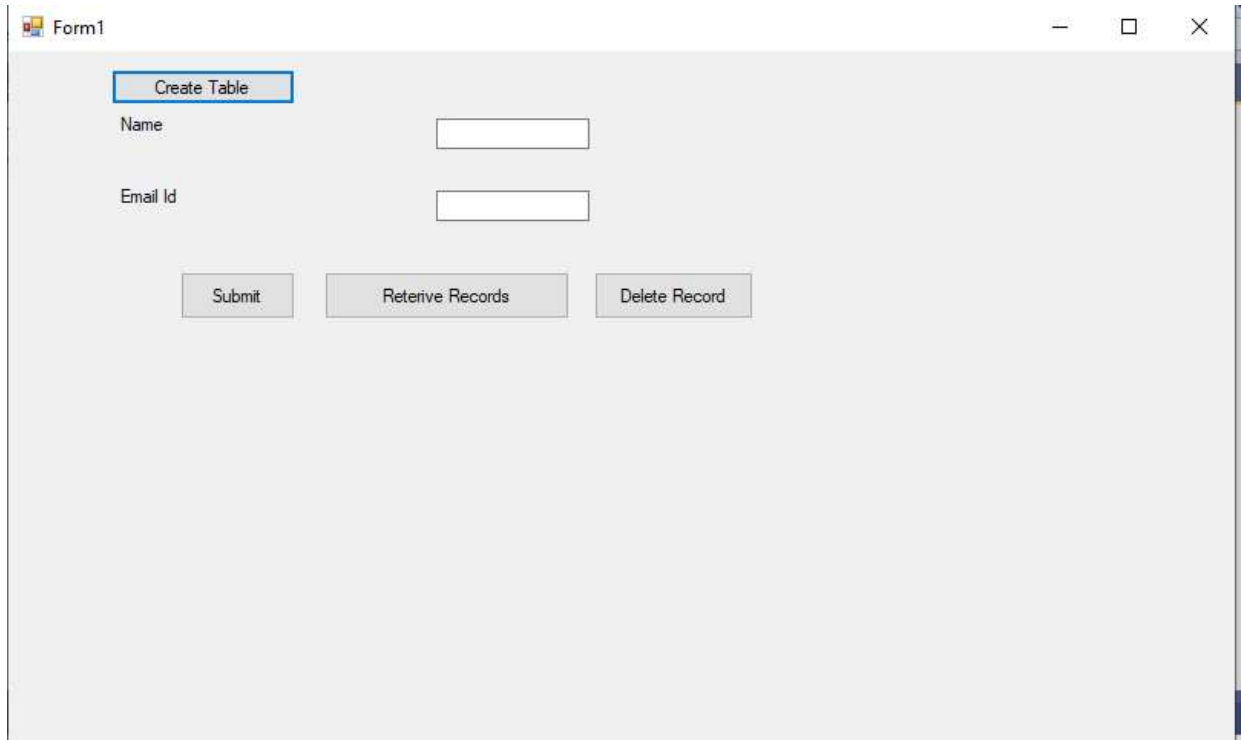
    private void button4_Click(object sender, EventArgs e)
    {
        SqlConnection con = getConnection();
        string query = "DELETE FROM UserTable";
        SqlCommand cmd = new SqlCommand(query, con);
        try
        {
            cmd.ExecuteNonQuery();
            label3.Text = "Record deleted.";
            dataGridView1.Visible = false;
        } catch(Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        finally
        {
            con.Close();
        }
    }

```

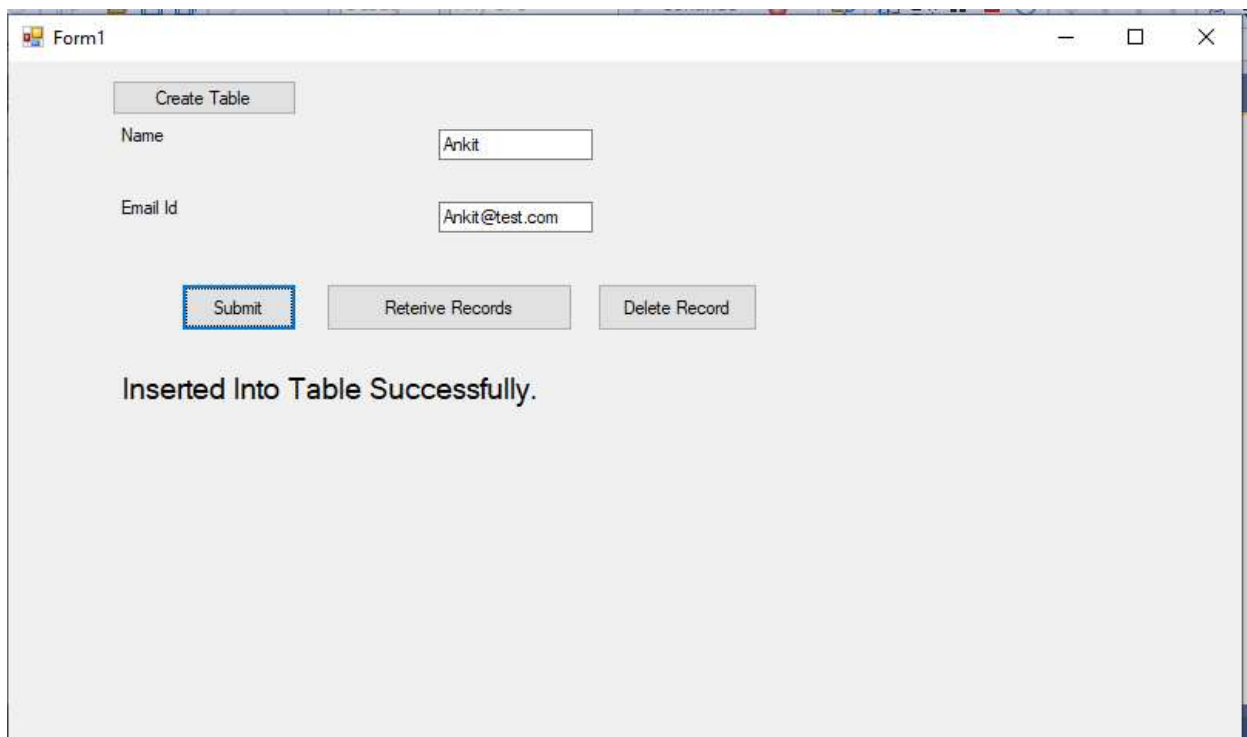


```
}  
}
```

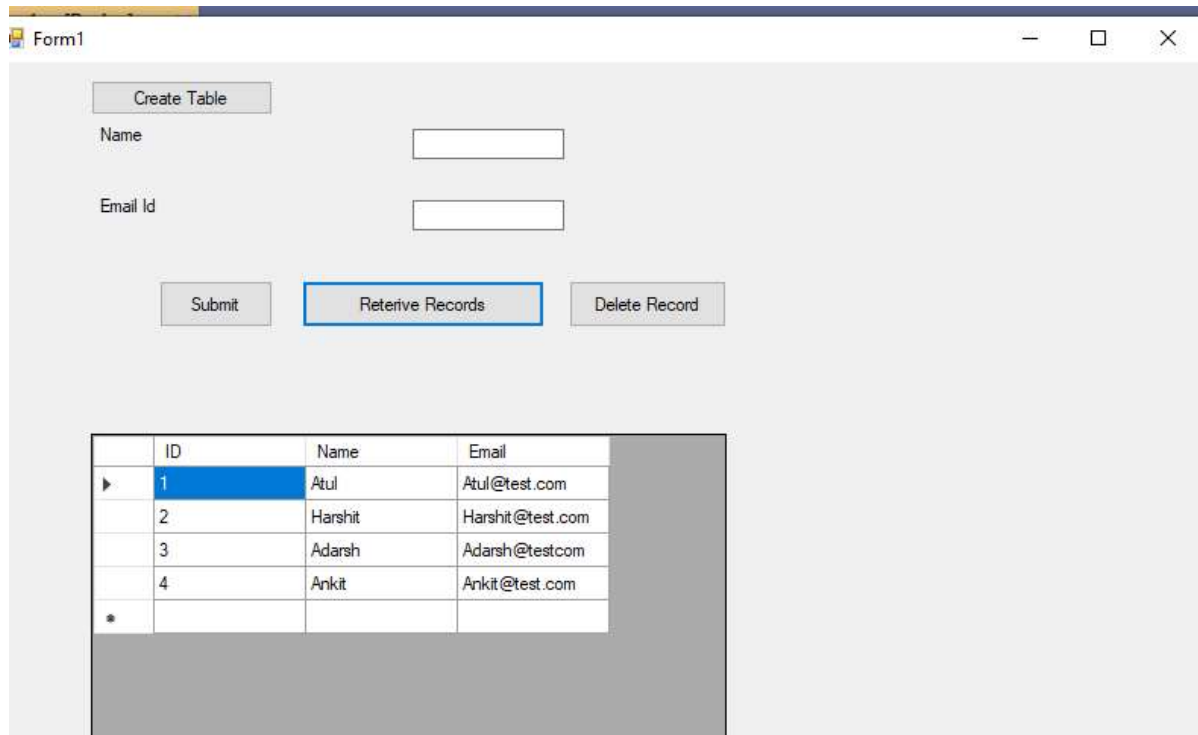
Output:



A screenshot of a Windows application window titled "Form1". The window contains a "Create Table" button at the top left. Below it are two text input fields: "Name" and "Email Id". The "Name" field is empty, and the "Email Id" field is also empty. At the bottom of the form, there are three buttons: "Submit", "Reterive Records" (note the typo), and "Delete Record".

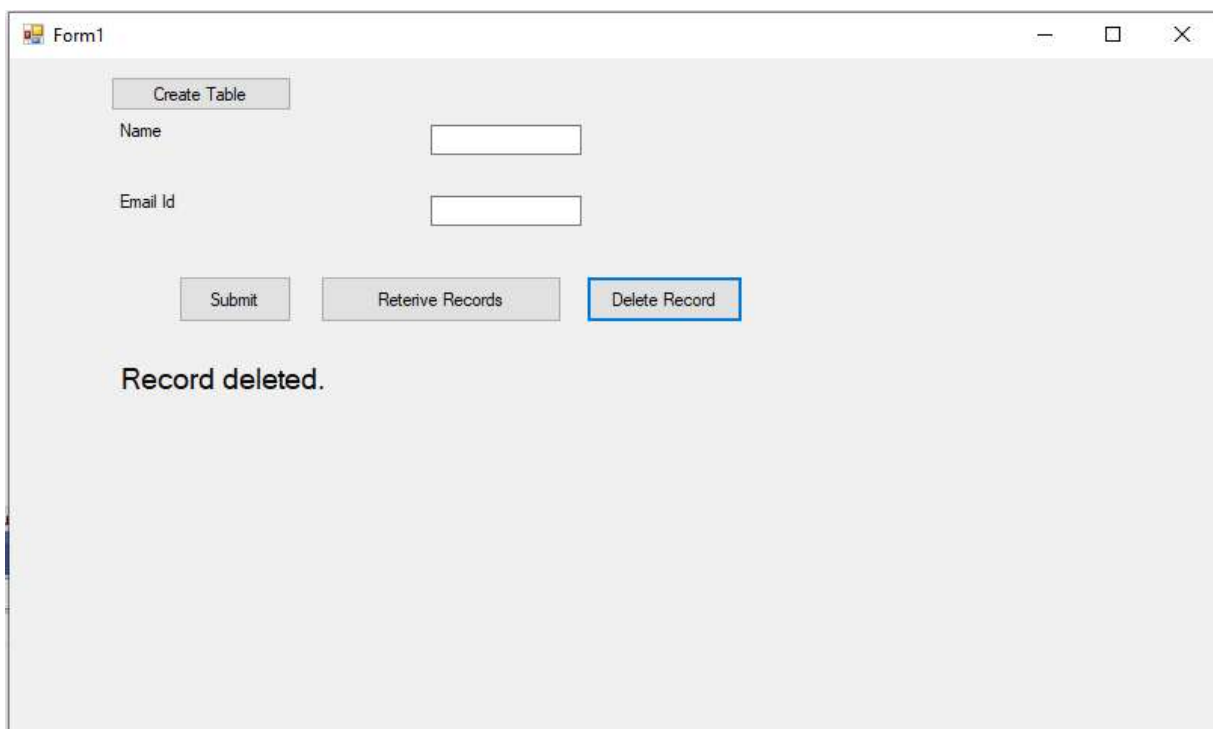


A screenshot of the same "Form1" window after data has been entered. The "Name" field now contains the text "Ankit" and the "Email Id" field contains "Ankit@test.com". The "Submit" button is highlighted with a blue border, indicating it is the active control. Below the input fields, the text "Inserted Into Table Successfully." is displayed. The other buttons, "Reterive Records" and "Delete Record", remain unchanged.



The screenshot shows a web form titled "Form1" with a light gray background. At the top left, there is a button labeled "Create Table". Below it are two input fields: "Name" and "Email Id", each followed by a text box. Further down are three buttons: "Submit", "Reterive Records" (highlighted with a blue border), and "Delete Record". Below the buttons is a table with four columns: "ID", "Name", and "Email". The table contains four rows of data, with the first row highlighted in blue. A small asterisk icon is visible to the left of the last row.

ID	Name	Email
1	Atul	Atul@test.com
2	Harshit	Harshit@test.com
3	Adarsh	Adarsh@testcom
4	Ankit	Ankit@test.com



The screenshot shows the same web form titled "Form1" after a record has been deleted. The "Create Table" button is still present. The "Name" and "Email Id" input fields are empty. The "Submit", "Reterive Records", and "Delete Record" buttons are still present, with "Delete Record" highlighted. Below the buttons, the text "Record deleted." is displayed.

Record deleted.

Problem Statement 25: Retrieve Data in GridView Control in ASP.Net

Objective: To learn the implementation of Grid View in ASP.NET.

Description: GridView is a control used to display data in tables on a web page. It displays data in both rows and columns, where each column represents a field, and each row represents a record. GridView helps to perform key activities like Insert, Delete, Sorting, and Paging.

Code: home.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
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)
        {
            string constr = @"Data Source=(localdb)\mssqllocaldb;Initial Catalog=TestDB;Integrated
Security=True";
            SqlConnection con = new SqlConnection(constr);
            try
            {
                con.Open();
                string query = "SELECT * FROM UserTable";

                SqlCommand cmd = new SqlCommand(query, con);
                SqlDataAdapter sdt = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                sdt.Fill(dt);
                gv1.DataSource = dt;
                gv1.DataBind();
            } catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            finally
            {
                con.Close();
            }
        }
    }
}
```

Code: home.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="home.aspx.cs"
Inherits="WebApplication1.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 runat="server" ID="gv1" AutoGenerateColumns="false">
```

```
<Columns>
```

```
<asp:BoundField DataField="Name" HeaderText="Student Name" />
```

```
<asp:BoundField DataField="Email" HeaderText="Student Email" />
```

```
</Columns>
```

```
</asp:GridView>
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

Output:

Student Name	Student Email
Atul	Atul@test.com
Harshit	Harshit@test.com
Adarsh	Adarsh@testcom
Ankit	Ankit@test.com