

## Set-1

a.

1. **Program to display the Addition, Subtraction, Multiplication and Division of two numbers using Console Applications.**

**Description:** This program is designed to perform four basic arithmetic operations—addition, subtraction, multiplication, and division—on two numbers entered by the user. The program first prompts the user to input two integers. It then calculates the sum, difference, and product of the numbers, displaying the results in the console. For division, the program ensures that the result is a floating-point number by casting the numbers to `double`, preventing truncation when dividing integers. It also includes a safeguard to check for division by zero, displaying a warning if the second number is zero. After performing these operations, the program prints the results to the console, providing a simple and interactive way to learn basic arithmetic operations using C#.

### **PORGRAM:**

```
using System;
```

```
namespace addition_sub
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            // Print a description of the program
```

```
            Console.WriteLine("\n\nFunction to calculate the sum, subtraction, multiplication, and division of  
two numbers:\n");
```

```
            Console.WriteLine("-----\n");
```

```
            // Prompt the user to enter a number and read the input as an integer
```

```
            Console.WriteLine("Enter a number: ");
```

```
            int n1 = Convert.ToInt32(Console.ReadLine());
```

```
            // Prompt the user to enter another number and read the input as an integer
```

```
            Console.WriteLine("Enter another number: ");
```

```
            int n2 = Convert.ToInt32(Console.ReadLine());
```

```

// Perform the operations

int sum = n1 + n2;

Console.WriteLine("\nThe sum of two numbers is: " + sum);


int difference = n1 - n2;

Console.WriteLine("\nThe subtraction of two numbers is: " + difference);


int product = n1 * n2;

Console.WriteLine("\nThe multiplication of two numbers is: " + product);


// Ensure division is done using floating-point numbers
if (n2 != 0)
{
    double division = (double)n1 / n2;

    Console.WriteLine("\nThe division of two numbers is: " + division);
}
else
{
    Console.WriteLine("\nDivision by zero is not allowed.");
}


Console.ReadLine();
}
}
}

```

### OUTPUT:

Function to calculate the sum, subtraction, multiplication, and division of two numbers:

-----

Enter a number: 10

Enter another number: 5

The sum of two numbers is: 15

The subtraction of two numbers is: 5

The multiplication of two numbers is: 50

The division of two numbers is: 2

b.

. program to find the reverse of a string

Descript on: This C# program reverses a user-input string and displays the reversed version on the console. It prompts the user to enter a string, then converts the string into a character array

using the ``ToCharArray()`` method. The ``Array.Reverse()`` method is used to reverse the characters

in the array, and the reversed array is then converted back into a string. The resul ng reversed string is displayed on the console. The program handles basic string manipula on in C# by leveraging the array reversal technique. It provides a straigh orward example of how to work with strings and arrays in C#. Lastly, it waits for the user to press a key before closing the applica on, ensuring the result can be viewed.

PORGRAM:

using System;

namespace StringReverse

{

class Program

{

static void Main(string[] args)

{

// Ask the user to input a string

Console.Write("Enter a string: ");

string input = Console.ReadLine();

```

// Reverse the string using the Reverse() method and Convert it to a new string
char[] charArray = input.ToCharArray();
Array.Reverse(charArray);
string reversedString = new string(charArray);

// Display the reversed string
Console.WriteLine("Reversed string: " + reversedString);

// Wait for the user to press a key before closing
Console.ReadKey();
}
}
}

```

OUTPUT:

Enter a string: Hello

Reversed string: olleH

c.

ASP represents Active Server Pages. ASP.Net is a piece of .Net technology, and it contains CLR as well. It is an open-source server-side innovation that empowers the developers to fabricate amazing web administrations, sites, and web applications. It is a structure created by Microsoft on which we can grow new age sites utilizing web forms (aspx), MVC, HTML, JavaScript, and CSS and so on. It is a successor of Microsoft Active Server Pages (ASP).

Exe in simple words means an executable file that is generated when we build an application. The assemblies are directly loaded when we run an Exe. Meanwhile, an Exe cannot be shared with the other applications. It is a file extension for an executable file format. An executable file can be easily run by a program in Microsoft DOS or Windows through a command or a double click. DLL stands for Dynamic Link Library. It is a library that has codes that are hidden. The codes are encapsulated inside the library. An application can consist of n number of DLLs which can be shared with the other applications as well. Other applications that need to share this DLL need not worry about the code till the time they are able to call the function on this DLL.

## Set-2

a.

### 1. Write a simple calculator program using windows applications.

**Description:** This program is a simple Windows Forms calculator that performs basic arithmetic operations such as addition, subtraction, multiplication, and division. The user can input numbers using buttons (0-9) and select an operation by clicking on the respective operation buttons (+, -, \*, /). Once the user enters two numbers and selects an operation, they can click the "=" button to calculate and display the result. The input and result are shown in a textbox. There is also a "Clear" button to reset the input and variables. The program uses event handlers for each button to perform the necessary actions, such as appending numbers or calculating the result based on the selected operation. This simple calculator demonstrates the fundamentals of user interaction and arithmetic operations in a Windows Forms application.

#### **PROGRAM:**

```
using System;
using System.Windows.Forms;

namespace Calculator
{
    public partial class Calculator : Form
    {
        public Calculator()
        {
            InitializeComponent();

            string option;
            int num1, num2, result;

            // Number buttons
            private void button1_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "1"; }
            private void button2_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "2"; }
            private void button3_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "3"; }
            private void button6_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "4"; }
            private void button5_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "5"; }
            private void button13_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "6"; }
        }

        private void button8_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "7"; }
        private void button4_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "8"; }
        private void button11_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "9"; }
    }

    private void button12_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "0"; }
}

// Operation buttons
private void button7_Click(object sender, EventArgs e) { option = "*"; num1 =
int.Parse(textBox1.Text); textBox1.Clear(); }
private void button9_Click(object sender, EventArgs e) { option = "+"; num1 =
int.Parse(textBox1.Text); textBox1.Clear(); }
private void button10_Click(object sender, EventArgs e) { option = "-"; num1 =
int.Parse(textBox1.Text); textBox1.Clear(); }
```

```
private void button14_Click(object sender, EventArgs e) { option = "/"; num1 =  
int.Parse(textBox1.Text); textBox1.Clear(); }
```

```
// Result button
```

```
private void button15_Click(object sender, EventArgs e)
```

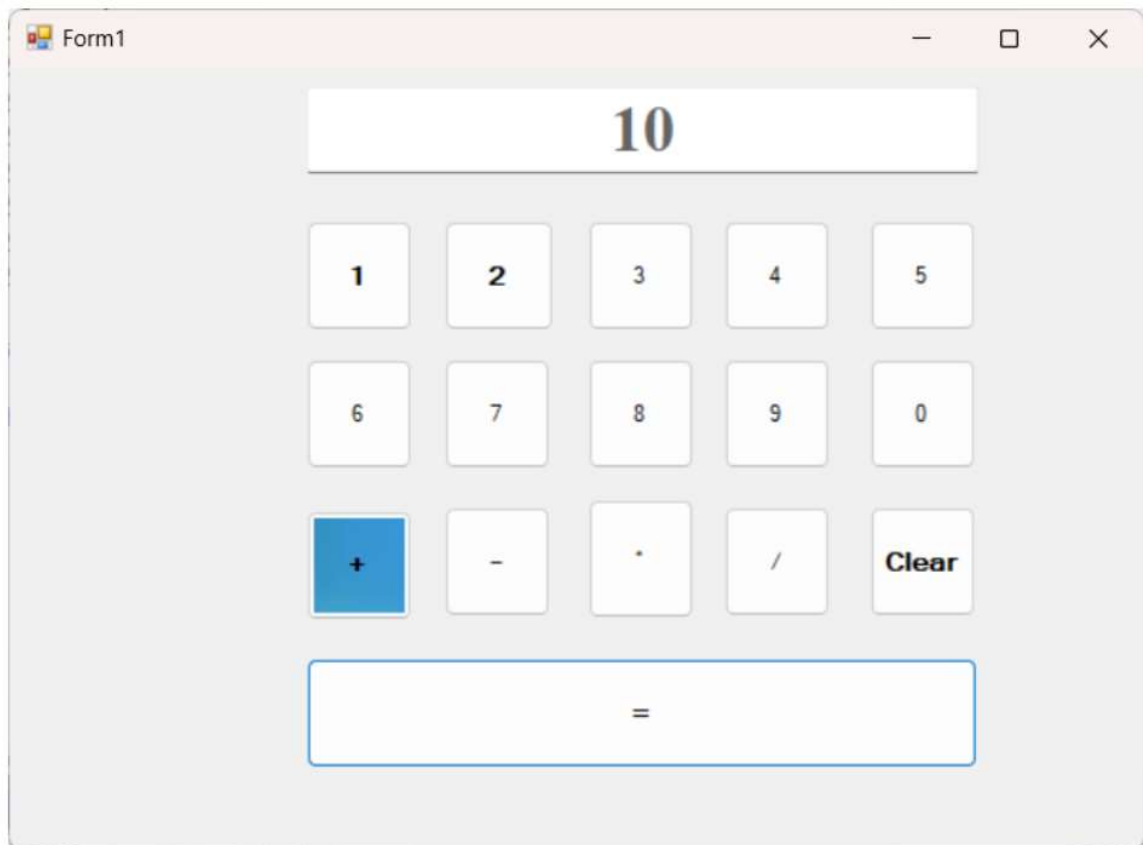
```
{  
    num2 = int.Parse(textBox1.Text);  
    switch (option)  
    {  
        case "+":  
            result = num1 + num2;  
            break;  
        case "-":  
            result = num1 - num2;  
            break;  
        case "*":  
            result = num1 * num2;  
            break;  
        case "/":  
            result = num1 / num2;  
            break;  
    }  
    textBox1.Text = result.ToString();  
}
```

```
// Clear button
```

```
private void button16_Click(object sender, EventArgs e)
```

```
{  
    textBox1.Clear();  
    result = 0;  
    num1 = 0;  
    num2 = 0;  
}  
}  
}
```

**OUTPUT:**



b.

AOT (Ahead-of-time) compilation converts Intermediate Language (IL) to machine code before execution, enabling optimizations by linking across modules to create a single executable. Unlike JIT (Just-in-time), AOT doesn't compile at runtime, optimizing instead for performance.

App models include workload-specific APIs like ASP.NET, Entity Framework, WPF, and Windows Forms. ASP.NET has two versions: ASP.NET 4.x, exclusive to .NET Framework, and ASP.NET Core, a cross-platform variant for .NET.

The .NET ecosystem comprises runtime software, development tools, and resources. Core .NET components include the Base Class Library (BCL) and CLR, responsible for memory management and code execution. Cross-platform capabilities allow applications to run on diverse operating systems.

Key .NET implementations include .NET Framework, .NET (formerly .NET Core), and Mono, each offering unique capabilities like small runtimes for mobile. A self-contained app compiled as Native AOT can run without the .NET runtime.

The .NET SDK includes libraries, CLI tools, and essential binaries to create and deploy applications across platforms.

## Set-3

a.

### 1. Program to display "Hello World" using console application using MS Visual Studio.

**Description:** To display "Hello World" in a console application using Microsoft Visual Studio, start by creating a new Console App project. Open Visual Studio and select Console App (.NET Core) or Console App (.NET Framework). In the default Program.cs file, replace the existing code with a simple `Console.WriteLine("Hello World");` to print the message to the console. The `Console.ReadKey();` command ensures the console window remains open until a key is pressed. Once you've made these changes, run the program by pressing `Ctrl + F5` or clicking the Start button. The program will display "Hello World" in the console window, and you can close it by pressing any key. This is a basic example of using C# in a console environment to interact with users through simple output.

#### PROGRAM:

```
using System;

namespace ConsoleApp1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
            Console.ReadKey();
        }
    }
}
```

**OUTPUT:** Hello World

b.

program to find the palindrome the given number

**Description:** This C# program checks if a given number is a palindrome. The user is prompted to enter a number, and the program stores the original number for comparison. Using a `while` loop, the number is reversed by extracting its last digit with the modulus operator and adding it

to the reversed number. The number is then reduced by dividing it by 10 in each iteration.

After

the loop, the program compares the original number with the reversed one. If they are equal, the number is a palindrome; otherwise, it is not. The program displays the result to the user and



waits for a key press before exiting. It demonstrates basic number manipulation and comparison techniques in C#.

PROGRAM:

```
using System;

public class PalindromeExample
{
    public static void Main(string[] args)
    {
        int n,r,sum=0,temp;

        Console.Write("Enter the Number: ");

        n = int.Parse(Console.ReadLine());

        temp=n;

        while(n>0)
        {
            r=n%10;

            sum=(sum*10)+r;

            n=n/10;

        }

        if(temp==sum)

            Console.Write("Number is Palindrome.");

        else

            Console.Write("Number is not Palindrome");

    }
}
```

OUTPUT:

Enter a number: 121

121 is a palindrome.

Enter a number: 123

123 is not a palindrome.

c.

JIT is abbreviated as Just in Time. It acts as a compiler that helps in converting the Intermediate Language to a Native code. The code is generally available in Native language during the execution step. Native code is nothing special but the hardware specifications that can be easily understood by the CPU. The native code even is stored so that it is accessible for subsequent calls by the end-users. In short, it converts the MSIL code within an assembly to native code that can be understood by the CPU architecture of the target machine to run a .NET application. It also cross-checks the values that are passed to parameters of any method. It helps in managing the execution of .NET programs regardless of any type of .NET programming language. The first step is a language-specific compiler converting the source code to the intermediate language. Then the intermediate language is transformed into the machine code by the JIT compiler.

#### Set-4

1. **Program to display the Addition, Subtraction, Multiplication and Division of two numbers using Console Applications.**

**Description:** This program is designed to perform four basic arithmetic operations—addition, subtraction, multiplication, and division—on two numbers entered by the user. The program first prompts the user to input two integers. It then calculates the sum, difference, and product of the numbers, displaying the results in the console. For division, the program ensures that the result is a floating-point number by casting the numbers to `double`, preventing truncation when dividing integers. It also includes a safeguard to check for division by zero, displaying a warning if the second number is zero. After performing these operations, the program prints the results to the console, providing a simple and interactive way to learn basic arithmetic operations using C#.

#### **PORGRAM:**

```
using System;
```

```
namespace addition_sub
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            // Print a description of the program
```

```
            Console.WriteLine("\n\nFunction to calculate the sum, subtraction, multiplication, and division of two numbers:\n");
```

```
            Console.WriteLine("-----\n");
```

```
            // Prompt the user to enter a number and read the input as an integer
```

```
            Console.WriteLine("Enter a number: ");
```

```
            int n1 = Convert.ToInt32(Console.ReadLine());
```

```
            // Prompt the user to enter another number and read the input as an integer
```

```
            Console.WriteLine("Enter another number: ");
```

```
            int n2 = Convert.ToInt32(Console.ReadLine());
```

```

// Perform the operations

int sum = n1 + n2;

Console.WriteLine("\nThe sum of two numbers is: " + sum);


int difference = n1 - n2;

Console.WriteLine("\nThe subtraction of two numbers is: " + difference);


int product = n1 * n2;

Console.WriteLine("\nThe multiplication of two numbers is: " + product);


// Ensure division is done using floating-point numbers
if (n2 != 0)
{
    double division = (double)n1 / n2;

    Console.WriteLine("\nThe division of two numbers is: " + division);
}
else
{
    Console.WriteLine("\nDivision by zero is not allowed.");
}


Console.ReadLine();
}
}
}

```

### OUTPUT:

Function to calculate the sum, subtraction, multiplication, and division of two numbers:

-----

Enter a number: 10

Enter another number: 5

The sum of two numbers is: 15

The subtraction of two numbers is: 5

The multiplication of two numbers is: 50

The division of two numbers is: 2

b.

. program to find the reverse of a string

Descript on: This C# program reverses a user-input string and displays the reversed version on the console. It prompts the user to enter a string, then converts the string into a character array

using the ``ToCharArray()`` method. The ``Array.Reverse()`` method is used to reverse the characters

in the array, and the reversed array is then converted back into a string. The resul ng reversed string is displayed on the console. The program handles basic string manipula on in C# by leveraging the array reversal technique. It provides a straigh orward example of how to work with strings and arrays in C#. Lastly, it waits for the user to press a key before closing the applica on, ensuring the result can be viewed.

PORGRAM:

using System;

namespace StringReverse

```
{  
    class Program  
    {  
        sta c void Main(string[] args)  
        {  
            // Ask the user to input a string  
            Console.Write("Enter a string: ");  
            string input = Console.ReadLine();
```

```

// Reverse the string using the Reverse() method and Convert it to a new string
char[] charArray = input.ToCharArray();
Array.Reverse(charArray);
string reversedString = new string(charArray);

// Display the reversed string
Console.WriteLine("Reversed string: " + reversedString);

// Wait for the user to press a key before closing
Console.ReadKey();
}
}
}

```

OUTPUT:

Enter a string: Hello

Reversed string: olleH

c.

MVC is a framework used to make web applications. It is a design model for building the .Net applications. The web application base expands on Model-View-Controller design which isolates the application rationale from UI, and input by the user will be constrained by the Controller.

- The Model The model is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself.
- The View It implies the introduction of information in a specific format, activated by a controller's choice to present the information. It uses the data prepared by the Controller to generate a final presentable response. They are script-based templating frameworks like JSP, ASP, PHP and simple to incorporate with AJAX innovation.
- The Controller The controller is in charge of reacting to the client input and perform interaction on the model objects. The controller gets the input, it approves the input and perform the business task that changes the state of data objects.

## Set-5

1. a. **Program to display the Addition, Subtraction, Multiplication and Division of two numbers using Console Applications.**

**Description:** This program is designed to perform four basic arithmetic operations—addition, subtraction, multiplication, and division—on two numbers entered by the user. The program first prompts the user to input two integers. It then calculates the sum, difference, and product of the numbers, displaying the results in the console. For division, the program ensures that the result is a floating-point number by casting the numbers to `double`, preventing truncation when dividing integers. It also includes a safeguard to check for division by zero, displaying a warning if the second number is zero. After performing these operations, the program prints the results to the console, providing a simple and interactive way to learn basic arithmetic operations using C#.

### **PORGRAM:**

```
using System;
```

```
namespace addition_sub
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            // Print a description of the program
```

```
            Console.WriteLine("\n\nFunction to calculate the sum, subtraction, multiplication, and division of two numbers:\n");
```

```
            Console.WriteLine("-----\n");
```

```
            // Prompt the user to enter a number and read the input as an integer
```

```
            Console.WriteLine("Enter a number: ");
```

```
            int n1 = Convert.ToInt32(Console.ReadLine());
```

```
            // Prompt the user to enter another number and read the input as an integer
```

```
            Console.WriteLine("Enter another number: ");
```

```
            int n2 = Convert.ToInt32(Console.ReadLine());
```

```

// Perform the operations

int sum = n1 + n2;

Console.WriteLine("\nThe sum of two numbers is: " + sum);


int difference = n1 - n2;

Console.WriteLine("\nThe subtraction of two numbers is: " + difference);


int product = n1 * n2;

Console.WriteLine("\nThe multiplication of two numbers is: " + product);


// Ensure division is done using floating-point numbers
if (n2 != 0)
{
    double division = (double)n1 / n2;

    Console.WriteLine("\nThe division of two numbers is: " + division);
}
else
{
    Console.WriteLine("\nDivision by zero is not allowed.");
}


Console.ReadLine();
}
}
}

```

### OUTPUT:

Function to calculate the sum, subtraction, multiplication, and division of two numbers:

-----

Enter a number: 10

Enter another number: 5



The sum of two numbers is: 15

The subtraction of two numbers is: 5

The multiplication of two numbers is: 50

The division of two numbers is: 2

b.

. program to find the reverse of a string

Descript on: This C# program reverses a user-input string and displays the reversed version on the console. It prompts the user to enter a string, then converts the string into a character array

using the ``ToCharArray()`` method. The ``Array.Reverse()`` method is used to reverse the characters

in the array, and the reversed array is then converted back into a string. The resul ng reversed string is displayed on the console. The program handles basic string manipula on in C# by leveraging the array reversal technique. It provides a straigh orward example of how to work with strings and arrays in C#. Lastly, it waits for the user to press a key before closing the applica on, ensuring the result can be viewed.

PORGRAM:

```
using System;
```

```
namespace StringReverse
```

```
{
```

```
    class Program
```

```
    {
```

```
        sta c void Main(string[] args)
```

```
        {
```

```
            // Ask the user to input a string
```

```
            Console.Write("Enter a string: ");
```

```
            string input = Console.ReadLine();
```

```

// Reverse the string using the Reverse() method and Convert it to a new string
char[] charArray = input.ToCharArray();
Array.Reverse(charArray);
string reversedString = new string(charArray);

// Display the reversed string
Console.WriteLine("Reversed string: " + reversedString);

// Wait for the user to press a key before closing
Console.ReadKey();
}
}
}

```

OUTPUT:

Enter a string: Hello

Reversed string: olleH

c.

The .NET Framework, developed by Microsoft, supports software development and is currently used in version 4.7.1. It enables creating both form-based and web-based applications and supports multiple programming languages, including C# and Visual Basic.

#### **.NET Framework Architecture:**

1. **Language:**
  - **WinForms** - For desktop applications (e.g., Notepad).
  - **ASP.NET** - For web applications on browsers.
  - **ADO.NET** - For database-interacting applications (e.g., SQL Server).
2. **Class Library:** Includes libraries with methods to perform file operations, making it easier to handle tasks like reading text from files.
3. **Common Language Runtime (CLR):**
  - **Exception Handling** - Manages runtime errors.
  - **Garbage Collection** - Frees unused resources, like closing database connections after use.

## Set-6

a.

### 1. Write a program to display the first 10 numbers and their sum using console application.

**Description:** This program is designed to display the first 10 numbers and calculate their sum using a simple loop. It begins by initializing a `sum` variable to zero, which will hold the cumulative total of the numbers. A `for` loop runs from 1 to 10, displaying each number and adding it to the `sum` variable during each iteration. After the loop completes, the program outputs the sum of the numbers, which in this case will be 55. The program is efficient as it avoids user input and automatically calculates the sum of the first 10 integers. It offers a simple demonstration of how loops can be used for both iteration and accumulation. This basic example is useful for understanding control structures in C# and performing arithmetic operations.

#### PROGRAM:

```
using System;

namespace add
{
    class Program
    {
        static void Main(string[] args)
        {
            int sum = 0;

            // Loop to display the first 10 numbers and calculate the sum
            for (int number = 1; number <= 10; number++)
            {
                Console.WriteLine("Number: " + number);
                sum += number;
            }

            // Display the sum of the numbers
            Console.WriteLine("Sum of the first 10 numbers: " + sum);
            Console.ReadLine();
        }
    }
}
```

#### OUTPUT:

```
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Number: 6
Number: 7
Number: 8
Number: 9
Number: 10
Sum of the first 10 numbers: 55
```

b.

program to find the palindrome the given number

Descript on: This C# program checks if a given number is a palindrome. The user is prompted to enter a number, and the program stores the original number for comparison. Using a `while` loop, the number is reversed by extracting its last digit with the modulus operator and adding it

to the reversed number. The number is then reduced by dividing it by 10 in each iteration.

After

the loop, the program compares the original number with the reversed one. If they are equal, the number is a palindrome; otherwise, it is not. The program displays the result to the user and

waits for a key press before exiting. It demonstrates basic number manipulation and comparison

techniques in C#.

PROGRAM:

using System;

```
public class PalindromeExample
{
    public static void Main(string[] args)
    {
        int n,r,sum=0,temp;
        Console.Write("Enter the Number: ");
        n = int.Parse(Console.ReadLine());
        temp=n;
        while(n>0)
        {
            r=n%10;
            sum=(sum*10)+r;
            n=n/10;
```

```

    }
    if(temp==sum)
        Console.WriteLine("Number is Palindrome.");
    else
        Console.WriteLine("Number is not Palindrome");
    }
}

```

OUTPUT:

Enter a number: 121

121 is a palindrome.

Enter a number: 123

123 is not a palindrome.

c.

Each .NET Framework data provider has a Connection object inheriting from DbConnection with a provider-specific ConnectionString property that defines the connection syntax. Four core data providers in .NET Framework are:

1. **System.Data.SqlClient** - Accesses Microsoft SQL Server.
2. **System.Data.OleDb** - Accesses OLE DB data sources.
3. **System.Data.Odbc** - Accesses ODBC data sources.
4. **System.Data.OracleClient** - Accesses Oracle (version 8.1.7+).

ADO.NET 2.0 introduced connection string builders (e.g., SqlConnectionStringBuilder) to construct valid connection strings at runtime, removing the need for manual concatenation.

**Windows Authentication** (or integrated security) is supported by providers with syntax as follows:

- **SqlClient**: Integrated Security=true; or Integrated Security=SSPI;
- **OleDb**: Integrated Security=SSPI;
- **Odbc**: Trusted\_Connection=yes;
- **OracleClient**: Integrated Security=yes;

## Set-7

a.

### 1. Program to display addition of two numbers using windows application.

**Description:** This program is a simple Windows Forms application that performs the addition of two numbers. It features a form with three textboxes and a button. The user enters two numbers in the first two textboxes, and when the button is clicked, the program retrieves these numbers, adds them, and displays the sum in the third textbox. The numbers entered by the user are converted to integers using `Convert.ToInt32()`, and the sum is calculated. The result is then displayed in the third textbox, formatted as a string. This program demonstrates how to handle user input and perform basic arithmetic operations in a Windows Forms environment. It's a useful example for beginners learning to create interactive desktop applications.

#### PROGRAM:

```
using System;  
using System.Windows.Forms;
```

```
namespace addition  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
        }  
  
        private void button1_Click(object sender, EventArgs e)  
        {  
            // Retrieve the input values from textboxes and convert them to integers  
            int num1 = Convert.ToInt32(textBox1.Text);  
            int num2 = Convert.ToInt32(textBox2.Text);  
  
            // Calculate the sum of the two numbers  
            int sum = num1 + num2;  
  
            // Display the sum in the third textbox (textBox3)  
            textBox3.Text = "Sum: " + sum.ToString();  
        }  
    }  
}
```

#### OUTPUT:

Form1

Program to display additon of two numbers

Number 1: 10

Number 2: 20

Addition Sum: 30

b.

program to find the palindrome the given number

Descrip on: This C# program checks if a given number is a palindrome. The user is prompted to enter a number, and the program stores the original number for comparison. Using a `while` loop, the number is reversed by extrac ng its last digit with the modulus operator and adding it

to the reversed number. The number is then reduced by dividing it by 10 in each itera on.

A er

the loop, the program compares the original number with the reversed one. If they are equal, the number is a palindrome; otherwise, it is not. The program displays the result to the user and

waits for a key press before exi ng. It demonstrates basic number manipula on and comparison

techniques in C#.

PROGRAM:

using System;

public class PalindromeExample

{

public static void Main(string[] args)

```

{
    int n,r,sum=0,temp;

    Console.Write("Enter the Number: ");

    n = int.Parse(Console.ReadLine());

    temp=n;

    while(n>0)
    {
        r=n%10;

        sum=(sum*10)+r;

        n=n/10;
    }

    if(temp==sum)

        Console.Write("Number is Palindrome.");

    else

        Console.Write("Number is not Palindrome");

}
}

```

OUTPUT:

Enter a number: 121

121 is a palindrome.

Enter a number: 123

123 is not a palindrome.

c.

AOT (Ahead-of-time) compilation converts Intermediate Language (IL) to machine code before execution, enabling optimizations by linking across modules to create a single executable. Unlike JIT (Just-in-time), AOT doesn't compile at runtime, optimizing instead for performance.

App models include workload-specific APIs like ASP.NET, Entity Framework, WPF, and Windows Forms. ASP.NET has two versions: ASP.NET 4.x, exclusive to .NET Framework, and ASP.NET Core, a cross-platform variant for .NET.

The .NET ecosystem comprises runtime software, development tools, and resources. Core .NET components include the Base Class Library (BCL) and CLR, responsible for memory management and code execution. Cross-platform capabilities allow applications to run on diverse operating systems.



Key .NET implementations include .NET Framework, .NET (formerly .NET Core), and Mono, each offering unique capabilities like small runtimes for mobile. A self-contained app compiled as Native AOT can run without the .NET runtime.

The .NET SDK includes libraries, CLI tools, and essential binaries to create and deploy applications across platforms.

## Set-8

a.

### 1. Write a C# program to convert input string to Upper Case and vice versa.

**Description:** This program is a Windows Forms application that allows users to convert input strings between uppercase and lowercase. It features two textboxes, where the user enters text in the first textbox. Clicking button1 converts the text to lowercase and displays the result in both a MessageBox and the second textbox. Similarly, clicking button2 converts the text to uppercase and shows the result in the second textbox and a MessageBox. Additionally, button3 clears the contents of both textboxes. The program demonstrates string manipulation in C# and provides an interactive way to convert text case through a graphical interface. It is a practical example for learning how to handle user input and perform basic operations in a Windows Forms environment.

#### **PROGRAM:**

```
using System;
```

```
using System.Windows.Forms;
```

```
namespace StringConversion
```

```
{
```

```
    public partial class Form1 : Form
```

```
    {
```

```
        public Form1()
```

```
        {
```

```
            InitializeComponent();
```

```
        }
```

```
        // Convert to lowercase and display in MessageBox and textbox
```

```
        private void button1_Click(object sender, EventArgs e)
```

```
        {
```

```
            string s1 = textBox1.Text; // Get the input text from textbox1
```

```
            textBox2.Text = s1.ToLower(); // Convert to lowercase and display in textbox2
```

```
            MessageBox.Show("Lowercase String: " + s1.ToLower()); // Show MessageBox with
```

```
lowercase string
```

```
        }
```

```
        // Convert to uppercase and display in MessageBox and textbox
```

```
        private void button2_Click(object sender, EventArgs e)
```

```
        {
```

```
            string s1 = textBox1.Text; // Get the input text from textbox1
```

```
            textBox2.Text = s1.ToUpper(); // Convert to uppercase and display in textbox2
```

```
            MessageBox.Show("Uppercase String: " + s1.ToUpper()); // Show MessageBox with
```

```
uppercase string
```

```
        }
```

```
        // Clear the textboxes when button3 is clicked
```

```
        private void button3_Click(object sender, EventArgs e)
```

```
        {
```

```
            textBox1.Clear(); // Clear the first textbox
```

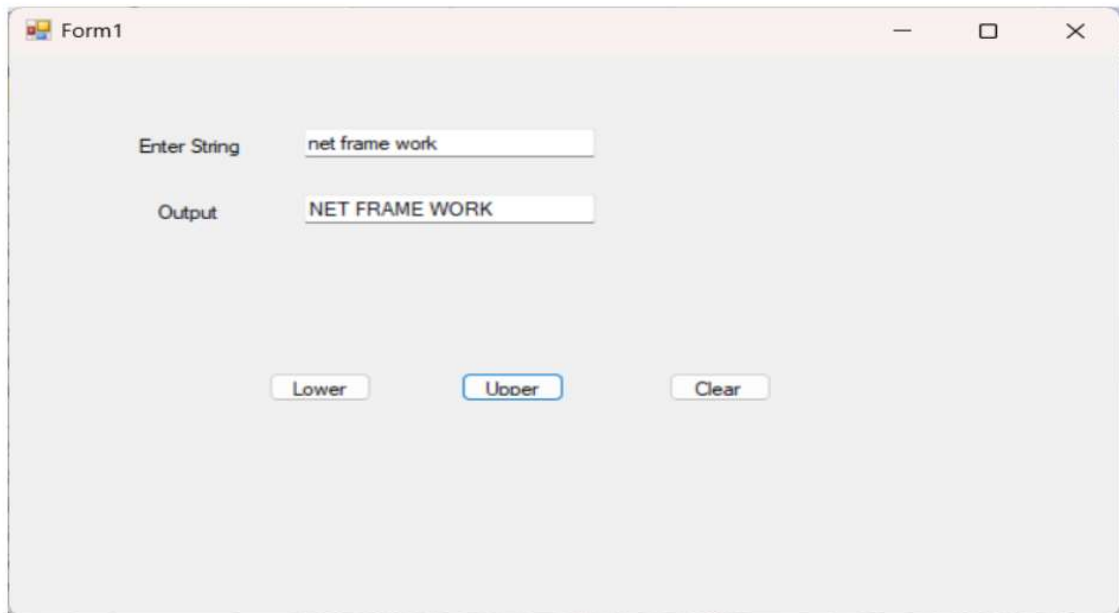
```
            textBox2.Clear(); // Clear the second textbox
```

```
        }
```

```
    }
```

}

**OUTPUT:**



The screenshot shows a Windows application window titled "Form1". Inside the window, there are two text input fields. The first field, labeled "Enter String", contains the text "net frame work". The second field, labeled "Output", contains the text "NET FRAME WORK". Below these fields are three buttons: "Lower", "Upper", and "Clear". The "Upper" button is currently selected, indicated by a blue border.

**b.**

**SQL Server Management Studio (SSMS)** is a versatile tool for managing SQL infrastructure, including SQL Server, Azure SQL Database, and Azure Synapse Analytics. SSMS integrates various graphical tools and script editors, providing developers and administrators with a unified environment to access, configure, manage, and develop SQL components.

**\*\*SSMS Components\*\*:**

1. **\*\*Object Explorer\*\***: Manages and views all objects in SQL Server instances.
2. **\*\*Template Explorer\*\***: Provides boilerplate text for query and script development.
3. **\*\*Solution Explorer\*\*** (deprecated): Manages projects like scripts and queries.
4. **\*\*Visual Database Tools\*\***: Design and build queries, tables, and database diagrams.
5. **\*\*Query and Text Editor\*\***: Builds and debugs queries interactively.

## Set-9

a.

### 1. Write a simple calculator program using windows applications.

**Description:** This program is a simple Windows Forms calculator that performs basic arithmetic operations such as addition, subtraction, multiplication, and division. The user can input numbers using buttons (0-9) and select an operation by clicking on the respective operation buttons ('+', '-', '\*', '/'). Once the user enters two numbers and selects an operation, they can click the "=" button to calculate and display the result. The input and result are shown in a textbox. There is also a "Clear" button to reset the input and variables. The program uses event handlers for each button to perform the necessary actions, such as appending numbers or calculating the result based on the selected operation. This simple calculator demonstrates the fundamentals of user interaction and arithmetic operations in a Windows Forms application.

#### **PROGRAM:**

```
using System;
```

```
using System.Windows.Forms;
```

```
namespace Calculator
```

```
{  
    public partial class Calculator : Form  
    {  
        public Calculator()  
        {  
            InitializeComponent();  
        }  
    }  
}
```

```
    string option;  
    int num1, num2, result;  
  
    // Number buttons  
    private void button1_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "1"; }  
    private void button2_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "2"; }  
    private void button3_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "3"; }  
    private void button6_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "4"; }  
    private void button5_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "5"; }  
    private void button13_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "6"; }  
}  
  
    private void button8_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "7"; }  
    private void button4_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "8"; }  
    private void button11_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "9"; }  
}  
  
    private void button12_Click(object sender, EventArgs e) { textBox1.Text = textBox1.Text + "0"; }  
}
```

```
    // Operation buttons  
    private void button7_Click(object sender, EventArgs e) { option = "*"; num1 =  
int.Parse(textBox1.Text); textBox1.Clear(); }  
    private void button9_Click(object sender, EventArgs e) { option = "+"; num1 =  
int.Parse(textBox1.Text); textBox1.Clear(); }  
    private void button10_Click(object sender, EventArgs e) { option = "-"; num1 =  
int.Parse(textBox1.Text); textBox1.Clear(); }
```

```
private void button14_Click(object sender, EventArgs e) { option = "/"; num1 =  
int.Parse(textBox1.Text); textBox1.Clear(); }
```

```
// Result button
```

```
private void button15_Click(object sender, EventArgs e)
```

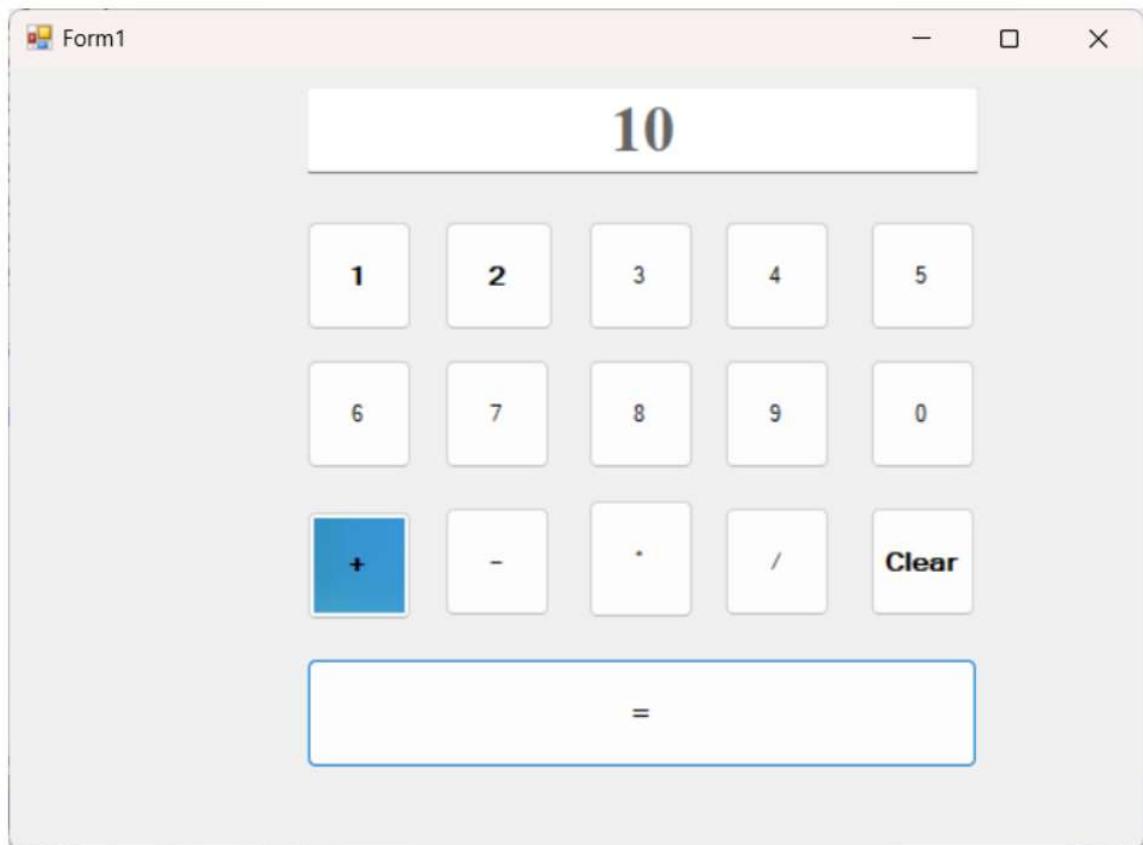
```
{  
    num2 = int.Parse(textBox1.Text);  
    switch (option)  
    {  
        case "+":  
            result = num1 + num2;  
            break;  
        case "-":  
            result = num1 - num2;  
            break;  
        case "*":  
            result = num1 * num2;  
            break;  
        case "/":  
            result = num1 / num2;  
            break;  
    }  
    textBox1.Text = result.ToString();  
}
```

```
// Clear button
```

```
private void button16_Click(object sender, EventArgs e)
```

```
{  
    textBox1.Clear();  
    result = 0;  
    num1 = 0;  
    num2 = 0;  
}  
}  
}
```

**OUTPUT:**



b.

AOT (Ahead-of-time) compilation converts Intermediate Language (IL) to machine code before execution, enabling optimizations by linking across modules to create a single executable. Unlike JIT (Just-in-time), AOT doesn't compile at runtime, optimizing instead for performance.

App models include workload-specific APIs like ASP.NET, Entity Framework, WPF, and Windows Forms. ASP.NET has two versions: ASP.NET 4.x, exclusive to .NET Framework, and ASP.NET Core, a cross-platform variant for .NET.

The .NET ecosystem comprises runtime software, development tools, and resources. Core .NET components include the Base Class Library (BCL) and CLR, responsible for memory management and code execution. Cross-platform capabilities allow applications to run on diverse operating systems.

Key .NET implementations include .NET Framework, .NET (formerly .NET Core), and Mono, each offering unique capabilities like small runtimes for mobile. A self-contained app compiled as Native AOT can run without the .NET runtime.

The .NET SDK includes libraries, CLI tools, and essential binaries to create and deploy applications across platforms.