**Swastik College**

Tribhuvan University

Chardobato, Bhaktapur, Nepal



**Lab Report of Dot Net Technology (CACS-302)**

Faculties of Humanities and Social Science

Tribhuvan University

Kritipur, Nepal

**Submitted By**

Name: **Anjan Khadka**

Roll no: **03 (5th sem)**

**Submitted To:**

Swastik College

Department of Bachelor in Computer Applications

Chardobato, Bhaktapur, Nepal

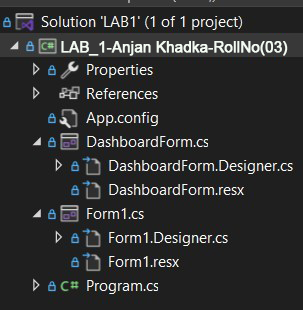
Signature: Internal Examiner External Examiner

**Acknowledgement**

I would like to express my sincere gratitude to my respected teacher, **Mr. Sunil Chaudhary**, for his valuable guidance and support during the Dot Net Technology course. His clear explanations and encouragement have greatly helped me in understanding both the theoretical and practical aspects of the subject, which has been very useful in preparing this lab report.

I am also thankful to my classmates for their cooperation and assistance during the lab sessions. Their help and discussions made it easier to solve problems and complete the practical tasks successfully. This report is a result of the collective learning environment created under the guidance of our teacher and the support of my peers.

**Lab 1: Write a program to create a C# Windows Forms application that implements a login form with a username and password field. The form should validate the credentials (username: "admin", password: "1234"), display a success or failure message, and clear the fields on failure. After three failed attempts, disable the login button. If the login is successful, open a new Dashboard Form and close the login form.**



**Code :**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LAB1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void txtName\_TextChanged(object sender, EventArgs e)

{

}

int attemptCounter = 0;

private void btnLogin\_Click(object sender, EventArgs e)

{

string username = txtName.Text;

string password = txtPassword.Text;

if (username == "admin" && password == "1234")

{

MessageBox.Show(

"Login successful!",

"Success",

MessageBoxButtons.OK,

MessageBoxIcon.Information

);

DashboardForm dashboard = new DashboardForm();

dashboard.Show();

this.Hide(); // Close login form

}

else

{

attemptCounter++;

MessageBox.Show(

"Invalid credentials!",

"Error",

MessageBoxButtons.OK,

MessageBoxIcon.Error

);

txtName.Clear();

txtPassword.Clear();

txtName.Focus();

if (attemptCounter >= 3)

{

btnLogin.Enabled = false;

MessageBox.Show(

"Maximum login attempts reached!",

"Locked Out",

MessageBoxButtons.OK,

MessageBoxIcon.Warning

);

}

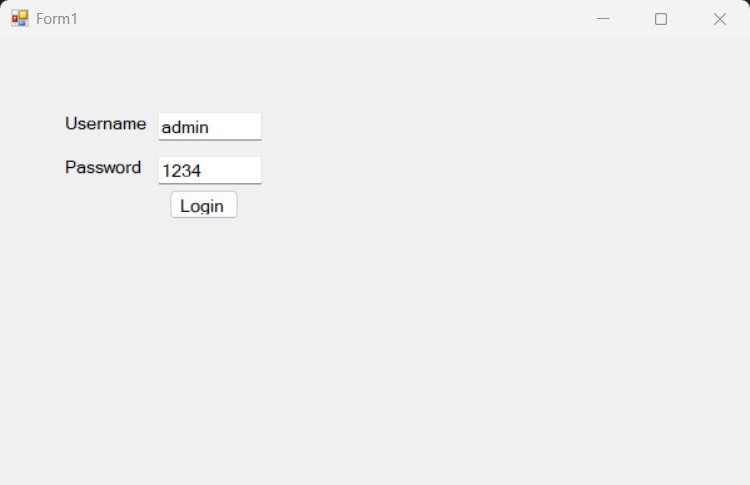
}

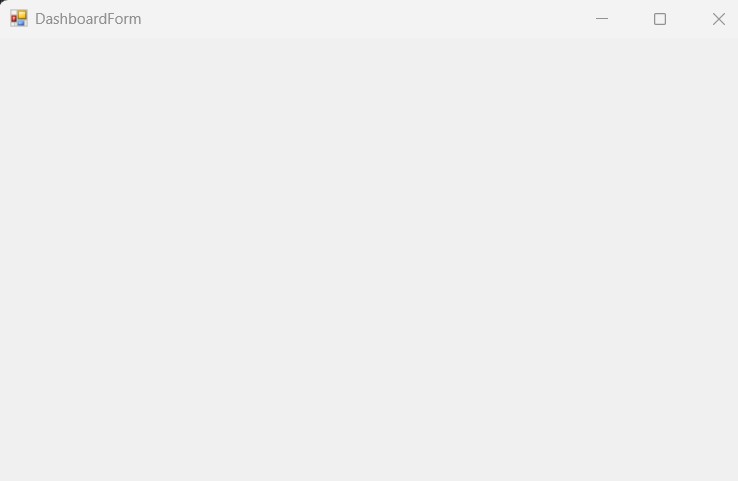
}

}

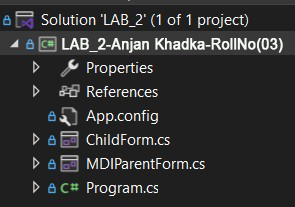
}

# Output





**Lab 2: Write a program to create a C# Windows (GUI) Forms application with an MDI Parent Form that contains a MenuStrip with "New" and "Exit" options. When the user clicks “New”, a Child Form should open inside the MDI Parent. Allow multiple child windows to be opened. When "Exit" is selected, the application should close.**



**Code:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LAB\_2

{

public partial class MDIParentForm : Form

{

private int childFormCount = 0;

public MDIParentForm()

{

InitializeComponent();

IsMdiContainer = true;

}

private void newToolStripMenuItem\_Click(object sender, EventArgs e)

{

ChildForm child = new ChildForm();

child.MdiParent = this;

child.Text = "Child " + (++childFormCount);

child.Show();

}

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

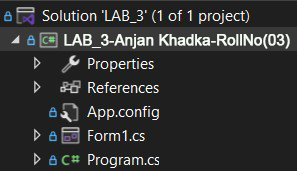
# Output



**Lab 3: Write a program to create a C# Windows Forms application that performs CRUD (Create, Read, Update, Delete) operations on a database table (e.g., a "Students" table with fields: ID, Name, Age, and Course). Implement the following functionalities:**

1. **Create: Allow users to add new records using text fields and a "Save" button.**
2. **Read: Display existing records in a DataGridView.**
3. **Update: Enable users to edit a selected record and update the database.**
4. **Delete: Provide a "Delete" button to remove a selected record.**
5. **Search: Implement a search bar to filter records based on Name or ID dynamically.**

**Use SQL Server as the database and ensure data is saved persistently.**



**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 LAB\_3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void btnSave\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection("Data Source=(localdb)\\mssqllocaldb; database=sdb; Integrated Security=true");

string sql = "insert into Students values(@a,@b,@c)";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@a", txtName.Text);

cmd.Parameters.AddWithValue("@b", txtAge.Text);

cmd.Parameters.AddWithValue("@c", txtCourse.Text);

con.Open();

cmd.ExecuteNonQuery(); // insert delete update

MessageBox.Show("Student Created");

}

private void Form1\_Load(object sender, EventArgs e)

{

LoadGrid();

}

public void LoadGrid()

{

SqlConnection con = new SqlConnection("Data Source=(localdb)\\mssqllocaldb; database=sdb; Integrated Security=true");

string sql = "select \* from Students";

SqlCommand cmd = new SqlCommand(sql, con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

dataGridView1.DataSource = dt;

}

public void ClearControls()

{

txtID.Text = "";

txtName.Text = "";

txtAge.Text = "";

txtCourse.Text = "";

txtName.Focus();

}

private void btnUpdate\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection("Data Source=(localdb)\\mssqllocaldb; database=sdb; Integrated Security=true");

string sql = "update Students set Name=@a, Age=@b, Course=@c where Id=@id";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@id", txtID.Text);

cmd.Parameters.AddWithValue("@a", txtName.Text);

cmd.Parameters.AddWithValue("@b", txtAge.Text);

cmd.Parameters.AddWithValue("@c", txtCourse.Text);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Student Updated");

LoadGrid();

ClearControls();

}

int id = 0;

private void btnDelete\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Are you sure want to Delete?", "Delete",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

SqlConnection con = new SqlConnection("Data Source=(localdb)\\mssqllocaldb; database=sdb; Integrated Security=true");

string sql = "delete from Students where id=@id";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@id", id);

con.Open();

cmd.ExecuteNonQuery(); // insert delete update

MessageBox.Show("Student Deleted");

LoadGrid();

ClearControls();

}

}

private void dataGridView1\_RowHeaderMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

txtID.Text = dataGridView1.CurrentRow.Cells[0].Value.ToString();

txtName.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

txtAge.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

txtCourse.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();

}

private void label5\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void btnSearch\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection("Data Source=(localdb)\\mssqllocaldb; database=sdb; Integrated Security=true");

string sql = "select \* from Students where Name Like @a";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@a", txtSearch.Text + "%");

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

if (dt.Rows.Count > 0)

{

dataGridView1.DataSource = dt;

}

else

{

MessageBox.Show("Record Not Found");

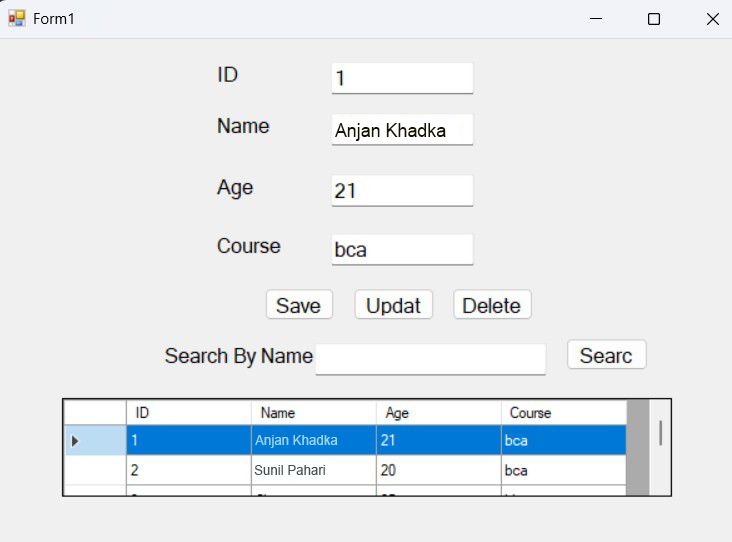
}

}

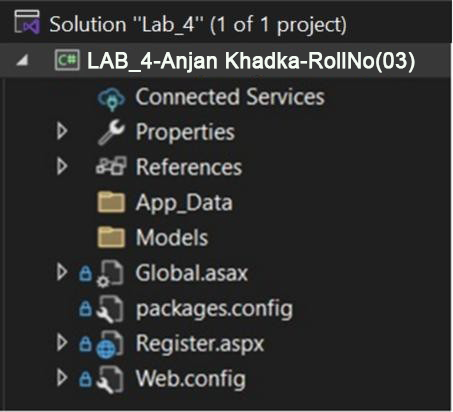
}

}

# Output



**Lab 4: Create an ASP.NET Form (Register.aspx) for user registration with fields for Full Name, Email, Password, Confirm Password, and Age, and apply appropriate ASP.NET validation controls to ensure Full Name is required, Email is required and in a valid format, Password is required with a minimum of 6 characters, Confirm Password matches Password, Age is between 18 and 99, and display a "Registration Successful!" message only when all validations pass along with a ValidationSummary to show all errors.**



# Code :

Register.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Register.aspx.cs" Inherits="Lab\_4.Register" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

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

<table style="width: 84%;">

<tr>

<td>

<asp:Label ID="Label1" runat="server" Text="REGISTRATION FORM"></asp:Label>

<div>

<asp:Label ID="lblSuccess" runat="server" ForeColor="Green" FontBold="True" Visible="false"></asp:Label>

</div>

</td>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

<tr>

<td>Full Name</td>

<td>

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

</td>

<td>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"

ControlToValidate="TextBox1" ErrorMessage="\*Please Enter Full Name"

ForeColor="Red" ValidationGroup="a">

</asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td>

<asp:Label ID="Label8" runat="server" Text="Email"></asp:Label>

</td>

<td>

<asp:TextBox ID="TextBox7" runat="server"></asp:TextBox>

</td>

<td>

<asp:RegularExpressionValidator ID="RegularExpressionValidator4"

runat="server" ControlToValidate="TextBox7" Display="Dynamic"

ErrorMessage="\* Enter Proper Email ID" ForeColor="Red"

SetFocusOnError="True"

ValidationExpression="\w+([-.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*"

ValidationGroup="a">

</asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td>

<asp:Label ID="Label3" runat="server" Text="Password"></asp:Label>

</td>

<td>

<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

</td>

<td>

<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"

ControlToValidate="TextBox2" Display="Dynamic"

ErrorMessage="\* Password must be 8 characters"

ForeColor="Red" SetFocusOnError="True"

ValidationExpression=".{8,}" ValidationGroup="a">

</asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td>

<asp:Label ID="Label4" runat="server" Text="Confirm Password"></asp:Label>

</td>

<td>

<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>

</td>

<td>

<asp:CompareValidator ID="CompareValidator1" runat="server"

ControlToCompare="TextBox2" ControlToValidate="TextBox3"

Display="Dynamic" ErrorMessage="\* Password Does Not Match"

ForeColor="Red" SetFocusOnError="True" ValidationGroup="a">

</asp:CompareValidator>

</td>

</tr>

<tr>

<td>

<asp:Label ID="Label7" runat="server" Text="Age"></asp:Label>

</td>

<td>

<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>

</td>

<td>

<asp:RangeValidator ID="RangeValidator1" runat="server"

ControlToValidate="TextBox6" Display="Dynamic"

ErrorMessage="\* Only 18-99 Age Allowed" ForeColor="Red"

MaximumValue="99" MinimumValue="18"

SetFocusOnError="True" Type="Integer" ValidationGroup="a">

</asp:RangeValidator>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>

<asp:Button ID="Button1" runat="server" Text="Submit"

ValidationGroup="a" OnClick="Button1\_Click" />

</td>

<td>&nbsp;</td>

</tr>

</table>

<div>

<asp:ValidationSummary ID="ValidationSummary1" runat="server"

ForeColor="Red" ValidationGroup="a" />

</div>

</form>

</body>

</html>

**Register.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Lab\_4

{

public partial class Register : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

// Hide the success message on page load

lblSuccess.Visible = false;

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

if (Page.IsValid)

{

lblSuccess.Text = "Registration Successful!";

lblSuccess.Visible = true;

ClearFields();

}

else

{

lblSuccess.Visible = false;

}

}

private void ClearFields()

{

TextBox1.Text = ""; // Full Name

TextBox7.Text = ""; // Email

TextBox2.Text = ""; // Password

TextBox3.Text = ""; // Confirm Password

TextBox6.Text = ""; // Age

}

}

}

**Web.config**

<?xml version="1.0" encoding="utf-8"?>

<!--

For more information on how to configure your ASP.NET application,

please visit https://go.microsoft.com/fwlink/?LinkId=169433

-->

<configuration>

<system.webServer>

<directoryBrowse enabled="true" />

</system.webServer>

<system.web>

<compilation debug="true" targetFramework="4.7.2" />

<httpRuntime targetFramework="4.7.2" />

</system.web>

<system.codedom>

<compilers>

<compiler language="c#;cs;csharp" extension=".cs"

type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,

Microsoft.CodeDom.Providers.DotNetCompilerPlatform,

Version=2.0.1.0, Culture=neutral,

PublicKeyToken=31bf3856ad364e35"

warningLevel="4"

compilerOptions="/langversion:default /nowarn:1659;1699;1701" />

<compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"

type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,

Microsoft.CodeDom.Providers.DotNetCompilerPlatform,

Version=2.0.1.0, Culture=neutral,

PublicKeyToken=31bf3856ad364e35"

warningLevel="4"

compilerOptions="/langversion:default /nowarn:41008

/define:\_MYTYPE=&quot;Web&quot; /optionInfer+" />

</compilers>

</system.codedom>

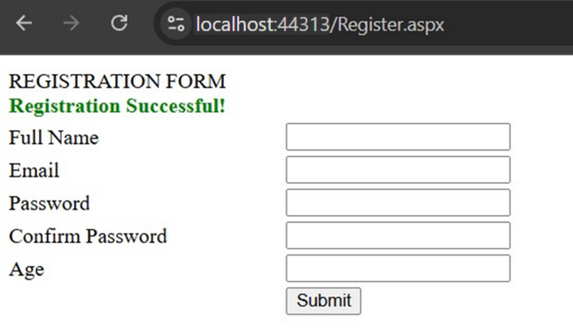
<appSettings>

<add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />

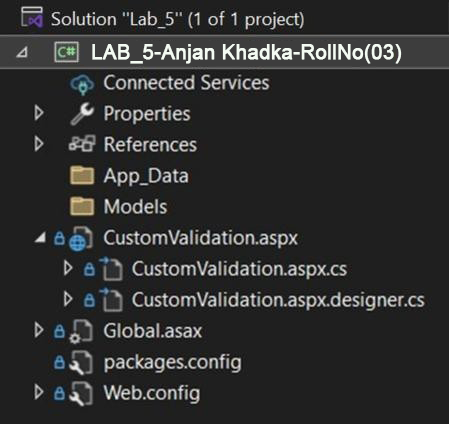
</appSettings>

</configuration>

# Output



**Lab 5: Create an ASP.NET Web Form (CustomValidation.aspx) with a field to enter a username, and use a CustomValidator to ensure that the username does not contain any special characters (only letters and numbers are allowed). Display an appropriate error message if the input is invalid and show a success message only if the input passes the validation.**



# Code :

## **CustomerValidation.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="CustomValidation.aspx.cs" Inherits="Lab\_5.CustomValidation" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

<style type="text/css">

.auto-style1 {

width: 100px;

}

.auto-style2 {

width: 204px;

}

</style>

</head>

<body>

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

<div>

<table style="width: 98%;">

<tr>

<td colspan="3" style="text-align:center;">

<asp:Label ID="lblMessage" runat="server" ForeColor="Green" FontBold="True" Visible="true" />

</td>

</tr>

<tr>

<td class="auto-style1">

<asp:Label ID="Label1" runat="server" Text="Username" />

</td>

<td class="auto-style2">

<asp:TextBox ID="txtUsername" runat="server" Width="194px" />

</td>

<td>

<asp:CustomValidator ID="CustomValidator1" runat="server"

ControlToValidate="txtUsername"

ErrorMessage="\* Username can only contain letters and numbers."

ForeColor="Red"

OnServerValidate="cvUsername\_ServerValidate" />

</td>

</tr>

<tr>

<td class="auto-style1">

<asp:Label ID="Label2" runat="server" Text="Password" />

</td>

<td class="auto-style2">

<asp:TextBox ID="txtPassword" runat="server" Width="196px" TextMode="Password" />

</td>

<td>

<asp:CustomValidator ID="CustomValidator2" runat="server"

ControlToValidate="txtPassword"

ErrorMessage="\* Password must be at least 8 characters long."

ForeColor="Red"

OnServerValidate="cvPassword\_ServerValidate" />

</td>

</tr>

<tr>

<td class="auto-style1">&nbsp;</td>

<td class="auto-style2">

<asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit\_Click" />

</td>

<td></td>

</tr>

</table>

</div>

</form>

</body>

</html>

**CustomValidation.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text.RegularExpressions;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Lab\_5

{

public partial class CustomValidation : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void cvUsername\_ServerValidate(object source, ServerValidateEventArgs args)

{

string pattern = @"^[a-zA-Z0-9]+$";

args.IsValid = Regex.IsMatch(args.Value, pattern);

}

protected void cvPassword\_ServerValidate(object source, ServerValidateEventArgs args)

{

args.IsValid = args.Value.Length >= 8;

}

protected void btnSubmit\_Click(object sender, EventArgs e)

{

if (Page.IsValid)

{

lblMessage.Text = "Username is valid!";

}

else

{

lblMessage.Text = "";

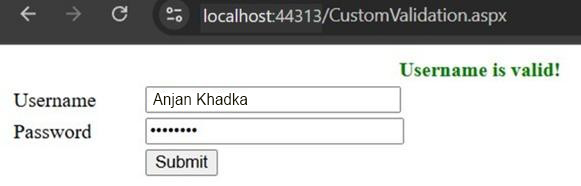
}

}

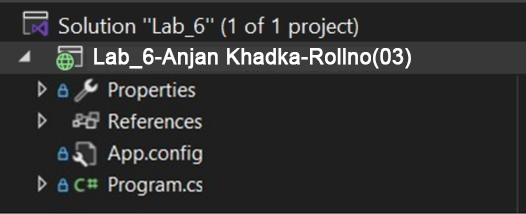
}

**}**

# Output



**Lab 6: Write a program to read two (mxn) matrices, perform addition operation and store result in third matrix.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_6

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter rows: ");

int m = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter columns: ");

int n = Convert.ToInt32(Console.ReadLine());

int[,] first = new int[m, n];

int[,] second = new int[m, n];

int[,] resultant = new int[m, n];

Console.WriteLine("Enter elements of First Matrix:");

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

Console.Write($"First[{i},{j}]: ");

first[i, j] = Convert.ToInt32(Console.ReadLine());

}

}

Console.WriteLine("Enter elements of Second Matrix:");

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

Console.Write($"Second[{i},{j}]: ");

second[i, j] = Convert.ToInt32(Console.ReadLine());

}

}

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

resultant[i, j] = first[i, j] + second[i, j];

}

}

Console.WriteLine("Resultant Matrix:");

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

Console.Write(resultant[i, j] + "\t");

}

Console.WriteLine();

}

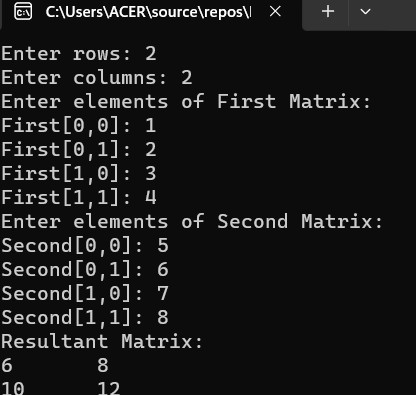
Console.ReadLine();

}

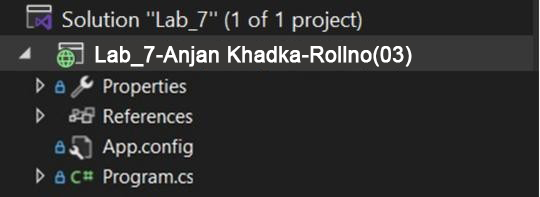
}

}

**Output**



**Lab 7: Write a C# program to read two matrices using jagged arrays, perform addition, and store the result in a third jagged array. Then, display all three matrices.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_7

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter number of rows: ");

int rows = Convert.ToInt32(Console.ReadLine());

int[][] firstMatrix = new int[rows][];

int[][] secondMatrix = new int[rows][];

int[][] resultMatrix = new int[rows][];

Console.Write("Enter number of columns: ");

int cols = Convert.ToInt32(Console.ReadLine());

for (int i = 0; i < rows; i++)

{

firstMatrix[i] = new int[cols];

secondMatrix[i] = new int[cols];

resultMatrix[i] = new int[cols];

}

Console.WriteLine("Enter elements of First Matrix:");

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

Console.Write($"First[{i},{j}]: ");

firstMatrix[i][j] = Convert.ToInt32(Console.ReadLine());

}

}

Console.WriteLine("Enter elements of Second Matrix:");

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

Console.Write($"Second[{i},{j}]: ");

secondMatrix[i][j] = Convert.ToInt32(Console.ReadLine());

}

}

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

resultMatrix[i][j] = firstMatrix[i][j] + secondMatrix[i][j];

}

}

Console.WriteLine("\nFirst Matrix:");

PrintMatrix(firstMatrix);

Console.WriteLine("\nSecond Matrix:");

PrintMatrix(secondMatrix);

Console.WriteLine("\nResultant Matrix (Addition):");

PrintMatrix(resultMatrix);

Console.ReadLine();

}

static void PrintMatrix(int[][] matrix)

{

for (int i = 0; i < matrix.Length; i++)

{

for (int j = 0; j < matrix[i].Length; j++)

{

Console.Write(matrix[i][j] + "\t");

}

Console.WriteLine();

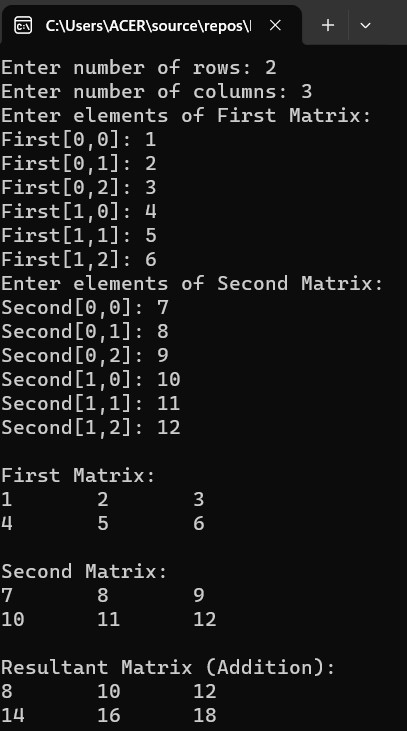
}

}

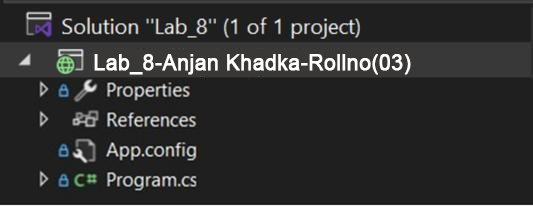
}

}

**Output**



**Lab 8: Write a C# program to read the user's Name, Age, and Country, and display a message using string interpolation in the format: "Hello [Name], you are [Age] years old and live in [Country].".**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_8

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter your name: ");

string name = Console.ReadLine();

Console.Write("Enter your age: ");

int age = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter your country: ");

string country = Console.ReadLine();

Console.WriteLine($"Hello {name}, you are {age} years old and live in {country}.");

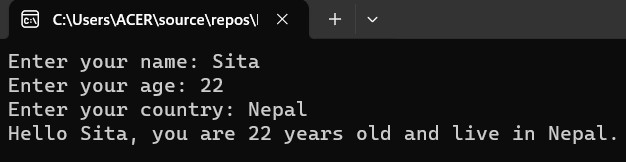
Console.ReadLine(); // Keep console open

}

}

}

**Output**



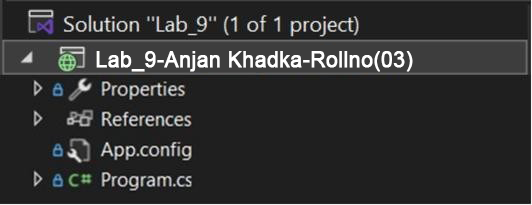
**Lab 9: Write a C# program that reads the user's first name**, **last name**, **age**, **country**, **favorite hobby, and job post. The program should display a personalized message using string interpolation as shown below.**

**Hello, [Full Name]!**

**You are [Age] years old and are [Eligible/Not Eligible] for senior citizen benefits.**

**You currently work as a [Job Title] in [Country].**

**Your favorite hobby is [Favorite Hobby]. That's awesome! "Thank you for sharing your details!"**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_9

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter your first name: ");

string firstName = Console.ReadLine();

Console.Write("Enter your last name: ");

string lastName = Console.ReadLine();

Console.Write("Enter your age: ");

int age = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter your country: ");

string country = Console.ReadLine();

Console.Write("Enter your favorite hobby: ");

string hobby = Console.ReadLine();

Console.Write("Enter your job post/title: ");

string jobTitle = Console.ReadLine();

string eligibility = age >= 60 ? "Eligible" : "Not Eligible";

Console.WriteLine($"\nHello, {firstName} {lastName}!");

Console.WriteLine($"You are {age} years old and are {eligibility} for senior citizen benefits.");

Console.WriteLine($"You currently work as a {jobTitle} in {country}.");

Console.WriteLine($"Your favorite hobby is {hobby}. That's awesome!");

Console.WriteLine("---");

Console.WriteLine("Thank you for sharing your details!");

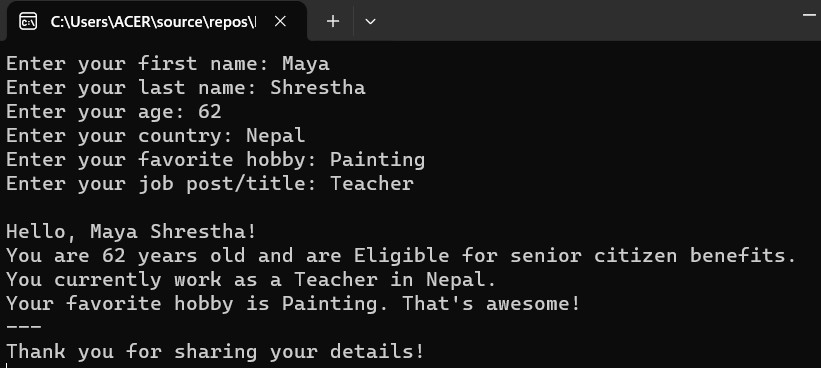
Console.ReadLine();

}

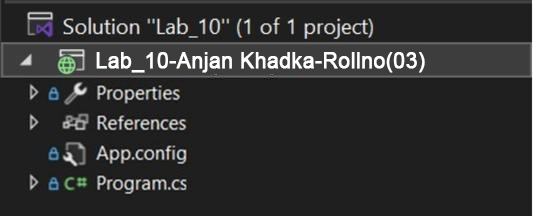
}

}

**Output**



**Lab 10: Write a method name isBalanceArray(int[])**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_10

{

class Program

{

static void Main(string[] args)

{

int[] arrone = { 2, 3, 4, 5 }; // balanced

int[] arrtwo = { 1, 3, 5, 7 }; // not balanced

Console.WriteLine(isBalanceArray(arrone)); // Output: True

Console.WriteLine(isBalanceArray(arrtwo)); // Output: False

Console.ReadLine();

}

public static bool isBalanceArray(int[] a)

{

int count\_even = 0;

int count\_odd = 0;

foreach (int num in a)

{

if (num % 2 == 0)

count\_even++;

else

count\_odd++;

}

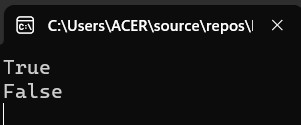
return count\_even == count\_odd;

}

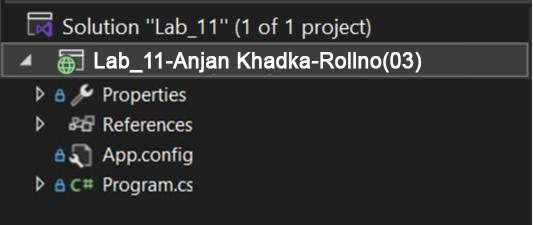
}

}

**Output**



**Lab 11: Write a method hasMirrorEnds(int[]).**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_11

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine(hasMirrorEnds(new int[] { 1, 2, 3, 2, 1 })); // True

Console.WriteLine(hasMirrorEnds(new int[] { 7, 8, 9, 8, 7 })); // True

Console.WriteLine(hasMirrorEnds(new int[] { 1, 2, 3, 4, 5 })); // False

Console.WriteLine(hasMirrorEnds(new int[] { 1, 2, 2, 1 })); // True

Console.WriteLine(hasMirrorEnds(new int[] { 1, 2, 3, 1 })); // False

Console.ReadLine();

}

public static bool hasMirrorEnds(int[] a)

{

int n = a.Length;

for (int i = 0; i < n / 2; i++)

{

if (a[i] != a[n - 1 - i])

{

return false;

}

}

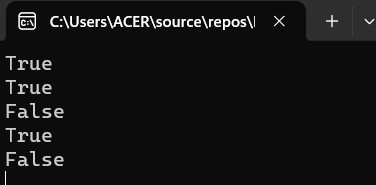
return true;

}

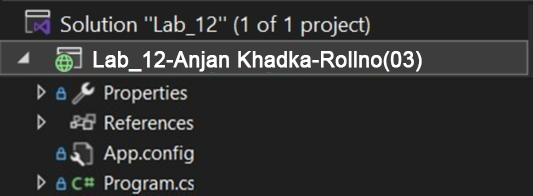
}

}

**Output**



**Lab 12: Write a C# program to initialize and display jagged array elements with sum of each row.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_12

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter number of rows: ");

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

int[][] jaggedArray = new int[rows][];

for (int i = 0; i < rows; i++)

{

Console.Write($"Enter number of elements in row {i + 1}: ");

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

jaggedArray[i] = new int[cols];

for (int j = 0; j < cols; j++)

{

Console.Write($"Enter element [{i + 1},{j + 1}]: ");

jaggedArray[i][j] = int.Parse(Console.ReadLine());

}

}

Console.WriteLine("\nJagged Array Elements and Sum of Each Row:\n");

for (int i = 0; i < jaggedArray.Length; i++)

{

int sum = 0;

Console.Write($"Row {i + 1}: ");

for (int j = 0; j < jaggedArray[i].Length; j++)

{

Console.Write(jaggedArray[i][j] + " ");

sum += jaggedArray[i][j];

}

Console.WriteLine($"=> Sum = {sum}");

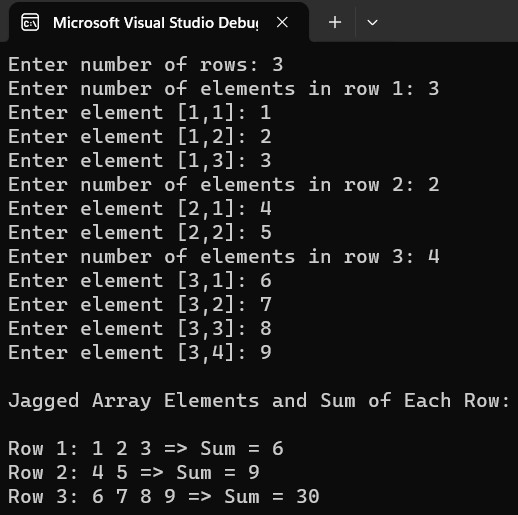
}

}

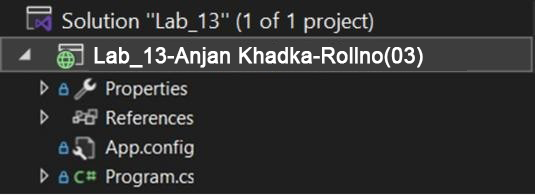
}

}

**Output**



**Lab 13: Write a C# program to find sum of rows in two dimension array.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_13

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter number of rows: ");

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

Console.Write("Enter number of columns: ");

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

int[,] array = new int[rows, cols];

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

Console.Write($"Enter element [{i + 1},{j + 1}]: ");

array[i, j] = int.Parse(Console.ReadLine());

}

}

Console.WriteLine("\nSum of each row:\n");

for (int i = 0; i < rows; i++)

{

int sum = 0;

Console.Write($"Row {i + 1}: ");

for (int j = 0; j < cols; j++)

{

Console.Write(array[i, j] + " ");

sum += array[i, j];

}

Console.WriteLine($"=> Sum = {sum}");

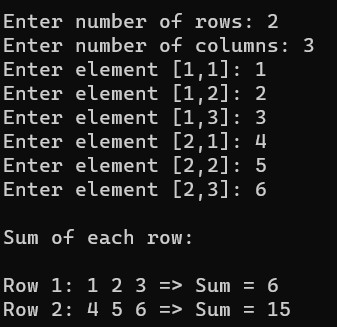
}

}

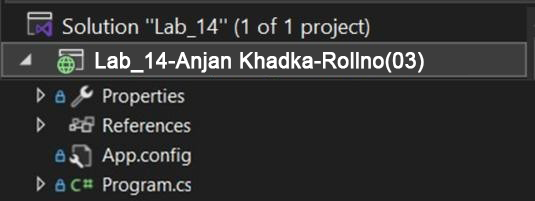
}

}

**Output**



**Lab 14: Write a C# program to swap two number using ref.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_14

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter first number: ");

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

Console.Write("Enter second number: ");

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

Console.WriteLine($"\nBefore Swap: a = {a}, b = {b}");

Swap(ref a, ref b);

Console.WriteLine($"After Swap: a = {a}, b = {b}");

}

static void Swap(ref int x, ref int y)

{

int temp = x;

x = y;

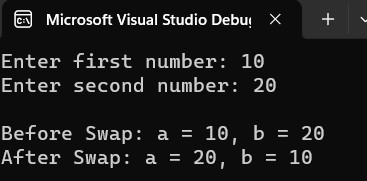
y = temp;

}

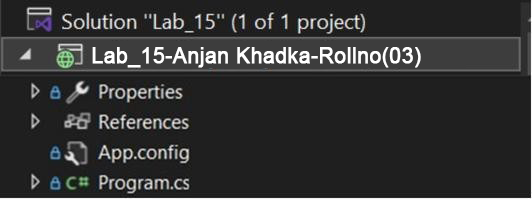
}

}

**Output**



**Lab 15: Write a program to demonstrate the concept of Indexer.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_15

{

class IndexerClass

{

private string[] name = new string[10]; // Changed size to 10

// Indexer

public string this[int index]

{

get { return name[index]; }

set { name[index] = value; }

}

}

class Program

{

static void Main(string[] args)

{

IndexerClass Team = new IndexerClass();

Team[0] = "Ram";

Team[1] = "Shyam";

Team[2] = "Hari";

Team[3] = "Gita";

Team[4] = "Sita";

Team[5] = "Hema";

Team[6] = "Rita";

Team[7] = "Mohan";

Team[8] = "Bikash";

Team[9] = "Bimal";

for (int i = 0; i < 10; i++)

{

Console.WriteLine(Team[i]);

}

Console.ReadLine();

}

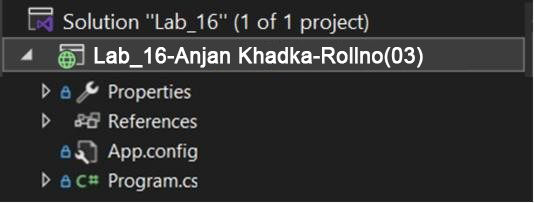
}

}

Output



**Lab 16: Write a C# program to overload Unary operator.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_16

{

class Program

{

static void Main(string[] args)

{

Calculation num = new Calculation(5);

// Unary + operator

Calculation positiveNum = +num;

Console.WriteLine("Unary + : " + positiveNum.Display()); // Output: 5

// Unary - operator

Calculation negatedNum = -num;

Console.WriteLine("Unary - : " + negatedNum.Display()); // Output: -5

// Unary ! operator

Console.WriteLine("Unary ! : " + (!num)); // Output: False

// Unary ++ operator

num++;

Console.WriteLine("Unary ++ : " + num.Display()); // Output: 6

// Unary -- operator

num--;

Console.WriteLine("Unary -- : " + num.Display()); // Output: 5

Console.ReadLine();

}

}

public class Calculation

{

int x;

public Calculation(int x)

{

this.x = x;

}

public static Calculation operator +(Calculation a)

{

return new Calculation(+a.x);

}

public static Calculation operator -(Calculation a)

{

return new Calculation(-a.x);

}

public static bool operator !(Calculation a)

{

return a.x == 0;

}

public static Calculation operator ++(Calculation a)

{

a.x += 1;

return a;

}

public static Calculation operator --(Calculation a)

{

a.x -= 1; // Decrement the value

return a;

}

public int Display()

{

return x;

}

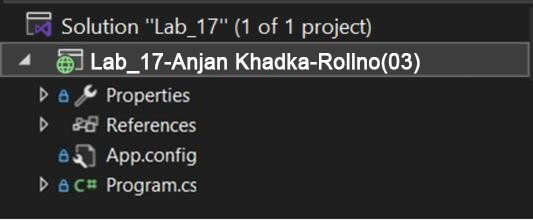
}

}

**Output**



**Lab 17: Write a program to overload Binary Operator**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_17

{

class Program

{

static void Main(string[] args)

{

Calculation a = new Calculation(10);

Calculation b = new Calculation(5);

Calculation sum = a + b;

Console.WriteLine("a + b = " + sum.Display()); // Output: 15

Calculation diff = a - b;

Console.WriteLine("a - b = " + diff.Display()); // Output: 5

Calculation product = a \* b;

Console.WriteLine("a \* b = " + product.Display()); // Output: 50

Calculation quotient = a / b;

Console.WriteLine("a / b = " + quotient.Display()); // Output: 2

Calculation remainder = a % 2;

Console.WriteLine("a % 2 = " + remainder.Display()); // Output: 0

Console.ReadLine();

}

}

public class Calculation

{

int x;

public Calculation(int x)

{

this.x = x;

}

// Binary + operator

public static Calculation operator +(Calculation a, Calculation b)

{

return new Calculation(a.x + b.x);

}

// Binary - operator

public static Calculation operator -(Calculation a, Calculation b)

{

return new Calculation(a.x - b.x);

}

// Binary \* operator

public static Calculation operator \*(Calculation a, Calculation b)

{

return new Calculation(a.x \* b.x);

}

// Binary / operator

public static Calculation operator /(Calculation a, Calculation b)

{

return new Calculation(a.x / b.x);

}

// Modulus operator with scalar

public static Calculation operator %(Calculation a, int scalar)

{

return new Calculation(a.x % scalar);

}

public int Display()

{

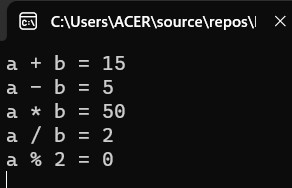
return x;

}

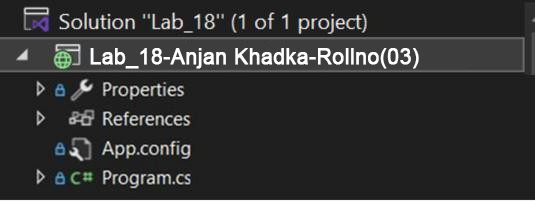
}

}

**Output**



**Lab 18: Write a program to overload Comparison operator**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_18

{

class Program

{

static void Main(string[] args)

{

Point p1 = new Point(3, 4); // Sum = 7

Point p2 = new Point(5, 2); // Sum = 7

Point p3 = new Point(1, 2); // Sum = 3

// Using overloaded comparison operators

Console.WriteLine(p1 == p2); // Output: False (coordinates not the same)

Console.WriteLine(p1 != p3); // Output: True (different points)

Console.WriteLine(p3 < p1); // Output: True (3 < 7)

Console.WriteLine(p1 > p3); // Output: True (7 > 3)

Console.WriteLine(p1 <= p2); // Output: True (sum equal)

Console.WriteLine(p2 >= p3); // Output: True (7 >= 3)

Console.ReadLine();

}

}

class Point

{

public int X;

public int Y;

public Point(int x, int y)

{

X = x;

Y = y;

}

// Overload the == operator

public static bool operator ==(Point p1, Point p2)

{

return p1.X == p2.X && p1.Y == p2.Y;

}

// Overload the != operator

public static bool operator !=(Point p1, Point p2)

{

return !(p1 == p2);

}

// Overload the < operator (sum of coordinates)

public static bool operator <(Point p1, Point p2)

{

return (p1.X + p1.Y) < (p2.X + p2.Y);

}

// Overload the > operator (sum of coordinates)

public static bool operator >(Point p1, Point p2)

{

return (p1.X + p1.Y) > (p2.X + p2.Y);

}

// Overload <= operator

public static bool operator <=(Point p1, Point p2)

{

return (p1.X + p1.Y) <= (p2.X + p2.Y);

}

// Overload >= operator

public static bool operator >=(Point p1, Point p2)

{

return (p1.X + p1.Y) >= (p2.X + p2.Y);

}

// Override Equals and GetHashCode when overloading == and !=

public override bool Equals(object obj)

{

if (obj is Point)

{

Point p = (Point)obj;

return this == p;

}

return false;

}

public override int GetHashCode()

{

return (X, Y).GetHashCode();

}

public override string ToString()

{

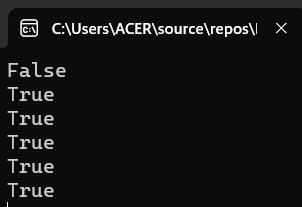
return $"({X}, {Y})";

}

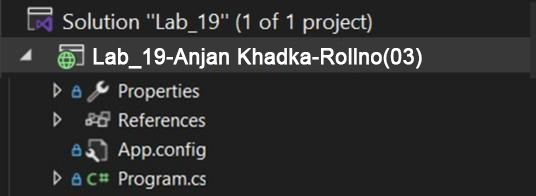
}

}

**Output**



**Lab 19: Write a C# program to overload unary (++) and relation operator (==) operator**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_19

{

class Program

{

static void Main(string[] args)

{

MyNumber num1 = new MyNumber(5);

MyNumber num2 = new MyNumber(5);

// Using the overloaded ++ operator

Console.WriteLine("Before increment: " + num1.Value);

++num1;

Console.WriteLine("After increment: " + num1.Value);

// Using the overloaded == operator

if (num1 == num2)

Console.WriteLine("num1 is equal to num2");

else

Console.WriteLine("num1 is not equal to num2");

Console.ReadLine();

}

}

class MyNumber

{

public int Value;

// Constructor

public MyNumber(int value)

{

Value = value;

}

// Overloading the ++ operator (prefix version)

public static MyNumber operator ++(MyNumber num)

{

num.Value++;

return num;

}

// Overloading the == operator

public static bool operator ==(MyNumber num1, MyNumber num2)

{

return num1.Value == num2.Value;

}

// Overloading the != operator (must be overloaded when == is overloaded)

public static bool operator !=(MyNumber num1, MyNumber num2)

{

return !(num1 == num2);

}

// Overriding Equals and GetHashCode methods

public override bool Equals(object obj)

{

if (obj is MyNumber num)

return Value == num.Value;

return false;

}

public override int GetHashCode()

{

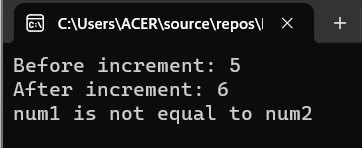
return Value.GetHashCode();

}

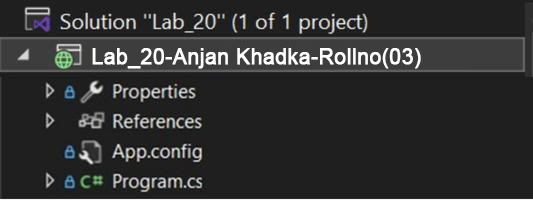
}

}

**Output**



**Lab 20: Write a program to calculate area of rectangle using simple inheritance.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_20

{

class Program

{

static void Main(string[] args)

{

Rectangle Rect = new Rectangle();

Rect.setWidth(5);

Rect.setHeight(7);

// Print the area of the object.

Console.WriteLine("Total area: " + Rect.getArea());

Console.ReadLine();

}

}

class Shape

{

protected int width;

protected int height;

public void setWidth(int w)

{

width = w;

}

public void setHeight(int h)

{

height = h;

}

}

// Derived class

class Rectangle : Shape

{

public int getArea()

{

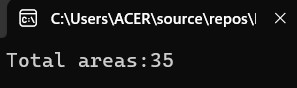
return width \* height;

}

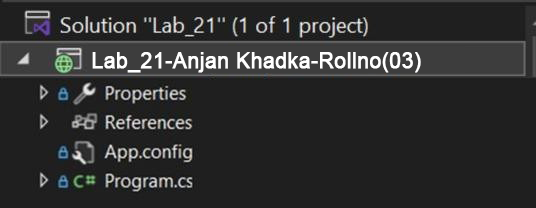
}

}

**Output**



**Lab 21: Write a program to calculate area of rectangle using multiple inheritance.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_21

{

class Program

{

static void Main(string[] args)

{

Rectangle Rect = new Rectangle();

int area;

Rect.setWidth(5);

Rect.setHeight(7);

area = Rect.getArea();

// Print the area of the object

Console.WriteLine("Total area: " + area);

Console.WriteLine("Total paint cost: " + Rect.getCost(area));

Console.ReadLine();

}

}

class Shape

{

protected int width;

protected int height;

public void setWidth(int w)

{

width = w;

}

public void setHeight(int h)

{

height = h;

}

}

// Interface for PaintCost

public interface PaintCost

{

int getCost(int area);

}

// Derived class

class Rectangle : Shape, PaintCost

{

public int getArea()

{

return width \* height;

}

public int getCost(int area)

{

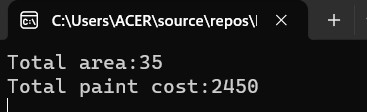
return area \* 70;

}

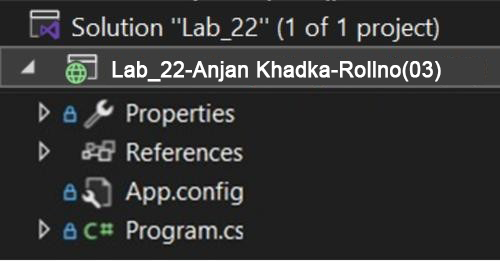
}

}

**Output**



**Lab 22. LINQ program to perform operations.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_22

{

class Program

{

static void Main(string[] args)

{

List<tblEmployee> lst = new List<tblEmployee>();

lst.Add(new tblEmployee() { Id = 1, Name = "Sunil Chaudhary", Gender = "Male", Country = "Nepal", Salary = 50000, RegDate = new DateTime(2024, 8, 27), Dob = new DateTime(1988, 1, 1) });

lst.Add(new tblEmployee() { Id = 2, Name = "Dipesh Shrestha", Gender = "Male", Country = "Nepal", Salary = 30000, RegDate = new DateTime(2024, 7, 27), Dob = new DateTime(1988, 1, 2) });

lst.Add(new tblEmployee() { Id = 3, Name = "Sujan Niraula", Gender = "Male", Country = "China", Salary = 20000, RegDate = new DateTime(2024, 6, 2), Dob = new DateTime(1987, 2, 5) });

lst.Add(new tblEmployee() { Id = 4, Name = "Saru Shrestha", Gender = "Female", Country = "India", Salary = 60000, RegDate = new DateTime(2024, 8, 28), Dob = new DateTime(1999, 9, 9) });

lst.Add(new tblEmployee() { Id = 5, Name = "Bikash Balami", Gender = "Male", Country = "Nepal", Salary = 80000, RegDate = new DateTime(2024, 5, 3), Dob = new DateTime(1989, 5, 6) });

lst.Add(new tblEmployee() { Id = 6, Name = "Niru Adhikari", Gender = "Female", Country = "India", Salary = 30000, RegDate = new DateTime(2024, 5, 5), Dob = new DateTime(1990, 6, 1) });

lst.Add(new tblEmployee() { Id = 7, Name = "Srijana Thapa", Gender = "Female", Country = "China", Salary = 80000, RegDate = new DateTime(2026, 6, 4), Dob = new DateTime(1996, 5, 2) });

lst.Add(new tblEmployee() { Id = 8, Name = "Naresh Dhami", Gender = "Male", Country = "Nepal", Salary = 40000, RegDate = new DateTime(2024, 8, 28), Dob = new DateTime(2000, 2, 2) });

// 1. Fetch all records

Console.WriteLine("1. Fetch all records");

foreach (tblEmployee emp in lst)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n2. Fetch all records from table with Name asc order");

var ascnameList = lst.OrderBy(a => a.Name).ToList();

foreach (var emp in ascnameList)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n3. Fetch all records from table with Name desc order");

var descnameList = lst.OrderByDescending(a => a.Name).ToList();

foreach (var emp in descnameList)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n4. Fetch top 3 records from table");

var top3list = lst.OrderBy(a => a.Name).Take(3).ToList();

foreach (var emp in top3list)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n5. Find average salary from given table");

var avgsalary = lst.Average(a => a.Salary);

Console.WriteLine(avgsalary);

Console.WriteLine("\n6. Fetch all employee whose country is Nepal or China");

var empNepalChinaList = lst.Where(a => a.Country == "Nepal" || a.Country == "China").ToList();

foreach (var emp in empNepalChinaList)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n7. Fetch all records of employee registered in August month");

var empAugList = lst.Where(a => a.RegDate.Month == 8).ToList();

foreach (var emp in empAugList)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n8. Fetch all records of employee registered between 8/26/2024 to 8/28/2024");

DateTime fromDate = new DateTime(2024, 8, 26);

DateTime toDate = new DateTime(2024, 8, 28);

var empDateBetween = lst.Where(a => a.RegDate >= fromDate && a.RegDate <= toDate).ToList();

foreach (var emp in empDateBetween)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n9. Fetch all records ordered by Name asc then by Salary");

var empNameSalaryAsc = lst.OrderBy(a => a.Name).ThenBy(a => a.Salary).ToList();

foreach (var emp in empNameSalaryAsc)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n10. Fetch all records whose country is Nepal and salary above 50000");

var listAboveSalary = lst.Where(a => a.Salary >= 50000 && a.Country == "Nepal").ToList();

foreach (var emp in listAboveSalary)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n11. Sum of salaries of all employees");

var sumSalary = lst.Sum(x => x.Salary);

Console.WriteLine("Sum of Salary: " + sumSalary);

Console.WriteLine("\n12. Max salary from employee table");

var maxSalary = lst.Max(x => x.Salary);

Console.WriteLine("Max Salary: " + maxSalary);

Console.WriteLine("\n13. Min salary from employee table");

var minSalary = lst.Min(x => x.Salary);

Console.WriteLine("Min Salary: " + minSalary);

Console.WriteLine("\n14. Get Id, Name, Salary from table");

var listRowFilter = lst.Select(x => new { x.Id, x.Name, x.Salary }).ToList();

foreach (var emp in listRowFilter)

{

Console.WriteLine("{0} {1} {2}", emp.Id, emp.Name, emp.Salary);

}

Console.WriteLine("\n15. Get Id, Name, 30% of Salary from table");

var listSalaryFilter = lst.Select(x => new { x.Id, x.Name, Salary = x.Salary \* 0.30M }).ToList();

foreach (var emp in listSalaryFilter)

{

Console.WriteLine("{0} {1} {2}", emp.Id, emp.Name, emp.Salary);

}

Console.WriteLine("\n16. Get all records where Name starts with 'S'");

var listStartsWithS = lst.Where(a => a.Name.StartsWith("S")).ToList();

foreach (var emp in listStartsWithS)

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}",

emp.Id, emp.Name, emp.Gender, emp.Country, emp.Salary, emp.RegDate.ToShortDateString(), emp.Dob.ToShortDateString());

}

Console.WriteLine("\n17. Get number of Female employees");

var totalFemale = lst.Count(a => a.Gender == "Female");

Console.WriteLine("Total No of Female: " + totalFemale);

Console.WriteLine("\n18. Get number of Male and Female employees");

var groupGender = lst.GroupBy(x => x.Gender).Select(y => new { Gender = y.Key, Count = y.Count() });

foreach (var emp in groupGender)

{

Console.WriteLine(emp.Gender + ": " + emp.Count);

}

Console.WriteLine("\n19. Sum of salaries as per Gender");

var groupGenderSalary = lst.GroupBy(x => x.Gender).Select(y => new { Gender = y.Key, SumOfSalary = y.Sum(z => z.Salary) });

foreach (var emp in groupGenderSalary)

{

Console.WriteLine(emp.Gender + ": " + emp.SumOfSalary);

}

Console.ReadLine();

}

}

public class tblEmployee

{

public int Id { get; set; }

public string Name { get; set; }

public string Gender { get; set; }

public string Country { get; set; }

public int Salary { get; set; }

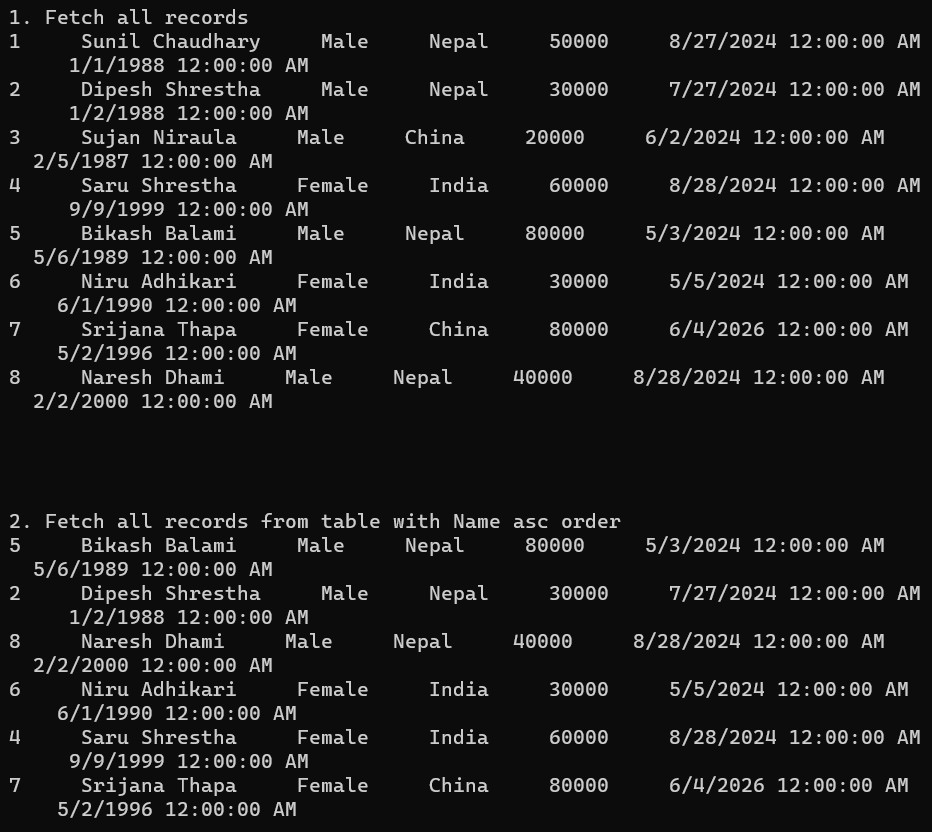
public DateTime RegDate { get; set; }

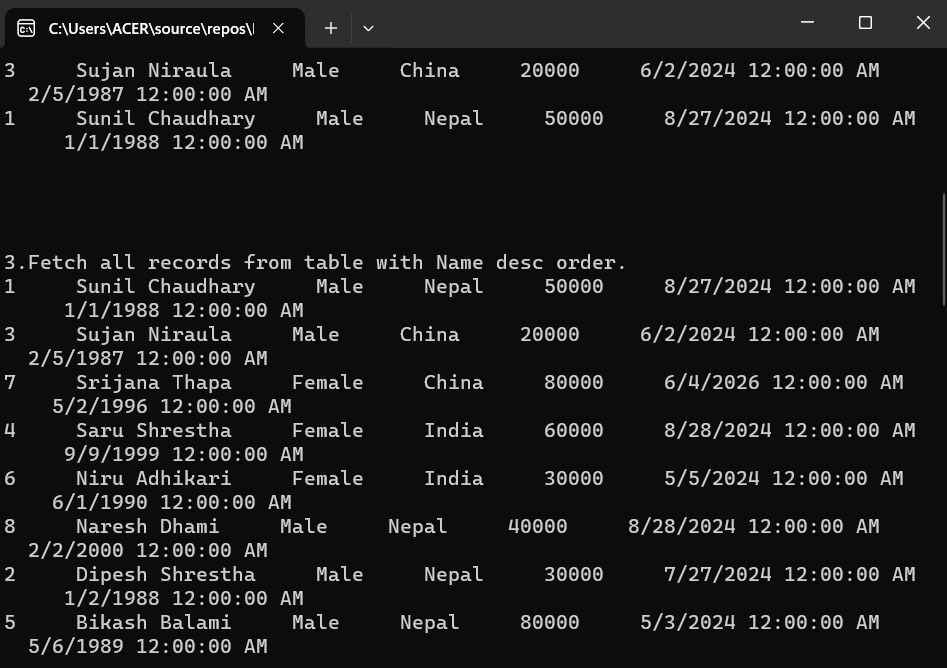
public DateTime Dob { get; set; }

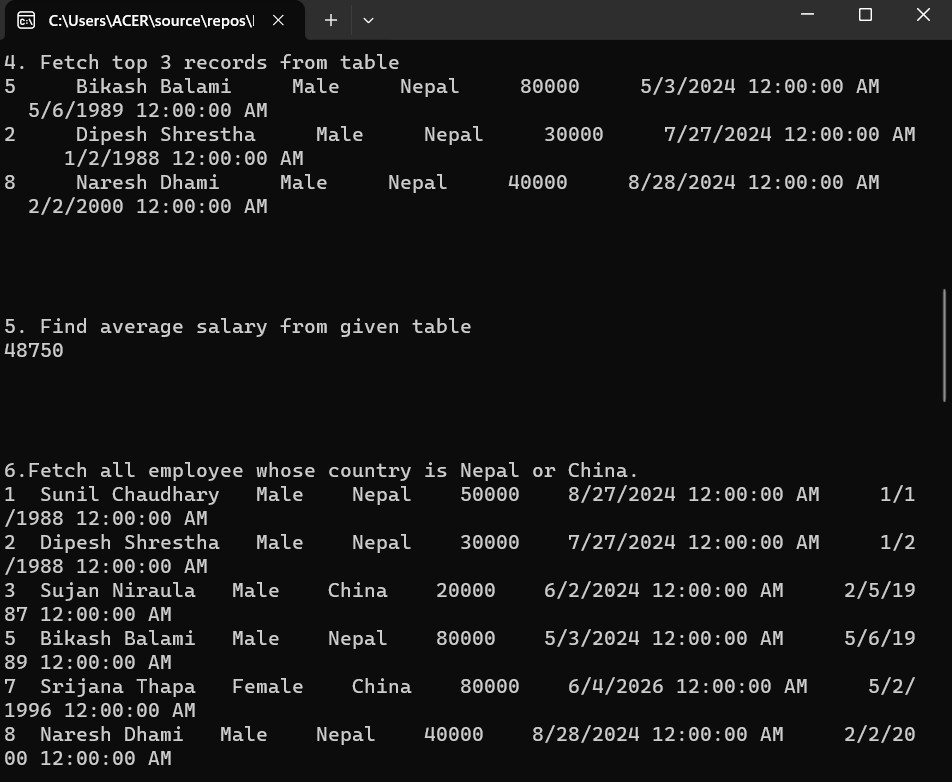
}

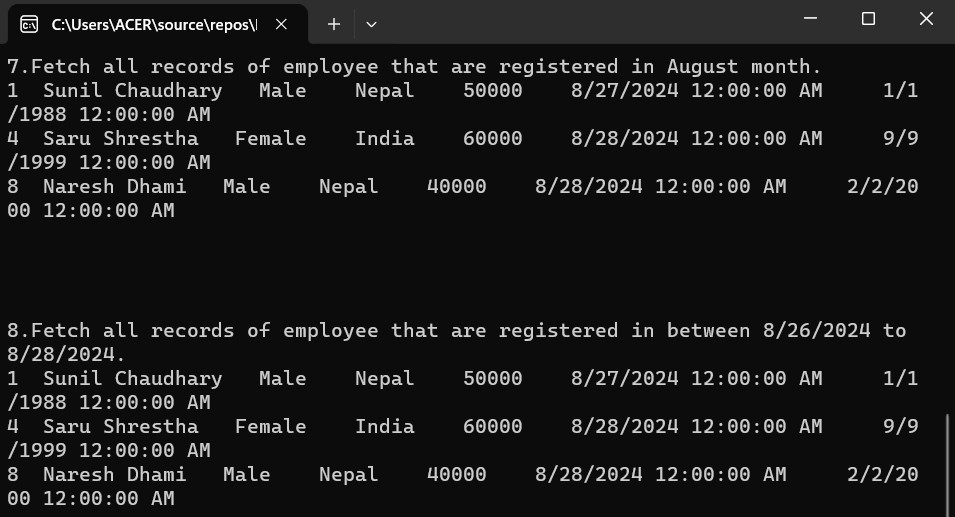
}

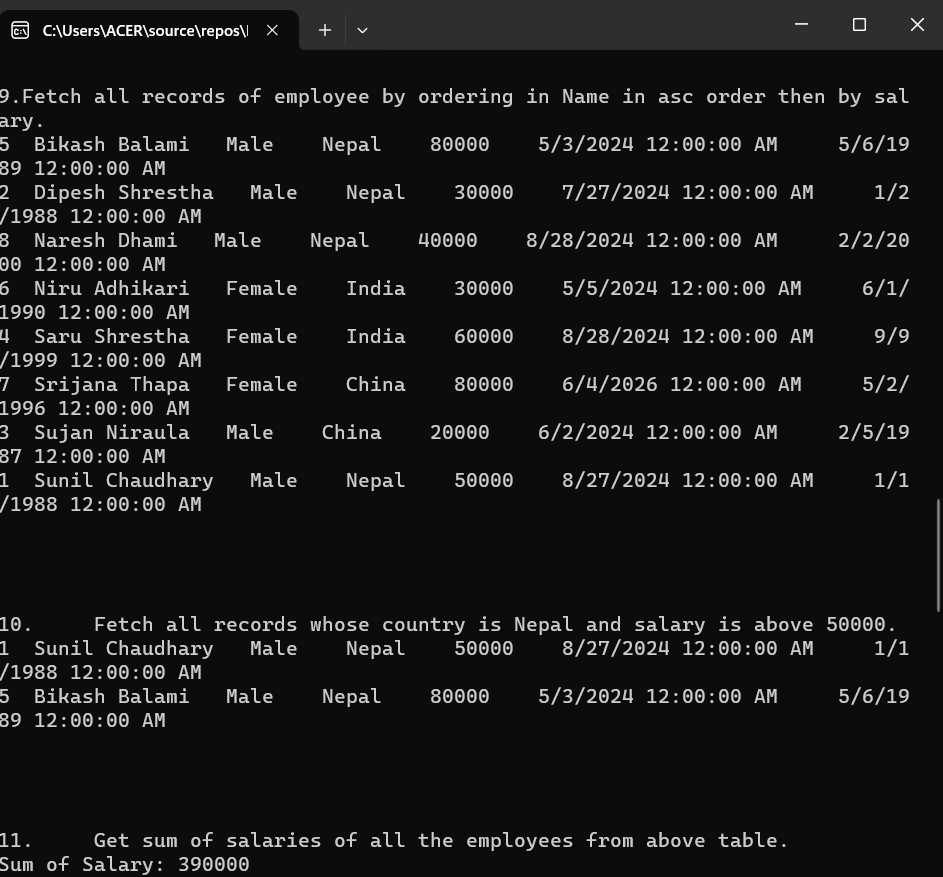
**Output**

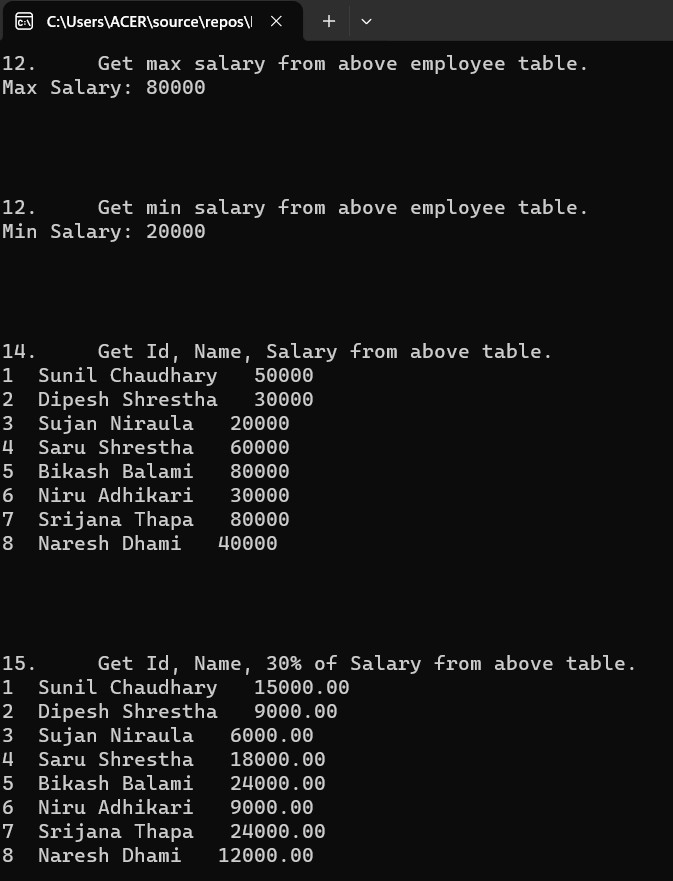


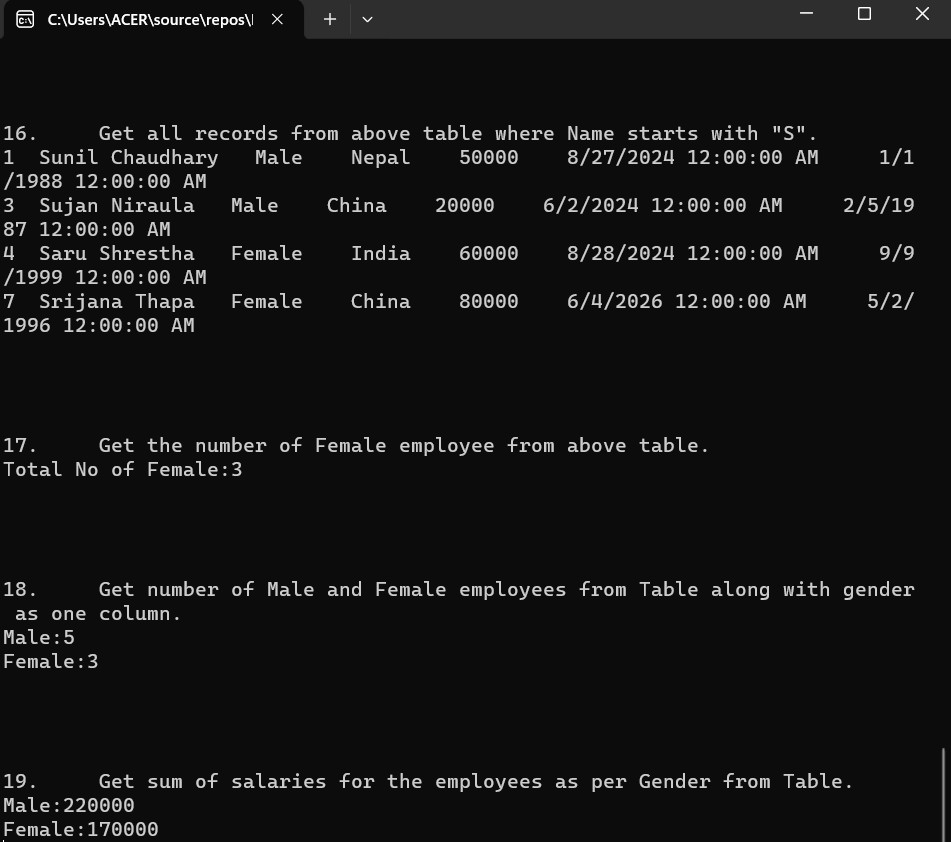




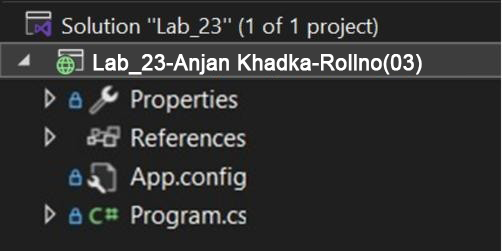








**Lab 23. Write a simple GUI program on how event is handled using delegates.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_23

{

// Delegate declaration

public delegate void DelEventHandler();

internal class Program

{

// Event declaration using the delegate

public event DelEventHandler add;

// Constructor

public Program()

{

// Subscribe the Initiate method to the event

add = new DelEventHandler(Initiate);

// Raise the event

add();

}

// Event handler method

private void Initiate()

{

Console.WriteLine("Event Initiated");

}

static void Main(string[] args)

{

// Create instance of Program, which triggers the event

new Program();

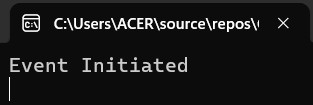
Console.ReadLine();

}

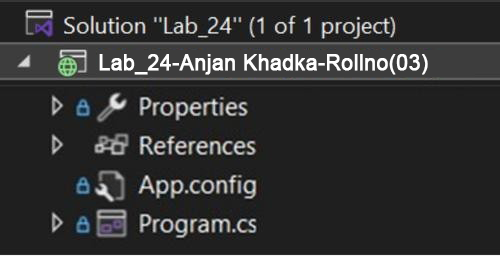
}

}

**Output**



**Lab 24: Write a simple program how event is handles using delegates.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab\_24

{

// Delegate declaration

public delegate void DelEventHandler();

internal class Program : Form

{

// Event declaration using the delegate

public event DelEventHandler add;

// Constructor

public Program()

{

// Create a button

Button btn = new Button();

btn.Parent = this;

btn.Text = "Hit Me";

btn.Location = new System.Drawing.Point(100, 100);

// Subscribe button click event

btn.Click += new EventHandler(onClick);

// Subscribe Initiate method to the custom event

add += new DelEventHandler(Initiate);

// Raise the event

add();

}

// Event handler for custom delegate

private void Initiate()

{

Console.WriteLine("Event Initiated");

}

// Button click event handler

public void onClick(object sender, EventArgs e)

{

MessageBox.Show("You Clicked me");

}

static void Main(string[] args)

{

// Run the Windows Form

Application.Run(new Program());

}

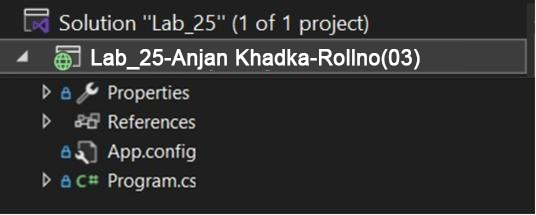
}

}

**Output**



**Lab 25: Write a C# program which store values on enumerations.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_25

{

class Program

{

// Enum for Departments

enum Department

{

Departmentone = 1,

Departmenttwo,

Departmentthree,

}

// Enum for Colleges

enum College

{

Collegeone = 1,

Collegetwo,

Collegethree,

}

// Method to list all departments

static void ListDepartments()

{

Console.WriteLine("Departments:");

foreach (var dept in Enum.GetValues(typeof(Department)))

{

Console.WriteLine(dept);

}

}

// Method to list all colleges

static void ListColleges()

{

Console.WriteLine("\nColleges:");

foreach (var college in Enum.GetValues(typeof(College)))

{

Console.WriteLine(college);

}

}

static void Main(string[] args)

{

ListDepartments();

ListColleges();

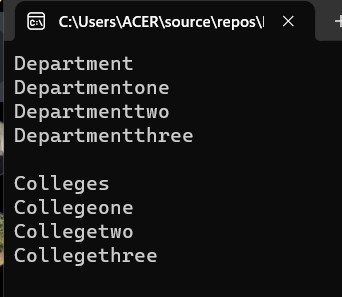
Console.ReadLine();

}

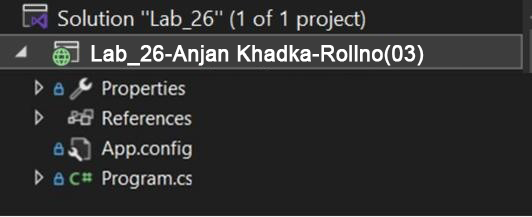
}

}

**Output**



**Lab 26: Create a C# program that stores values in an enumeration VehicleType and displays the fuel type for each vehicle (3.g. Car = Petrol, BIke = Petrol, Bus = Diesel)**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_26

{

class Program

{

// Enum for vehicle types

enum VehicleType

{

Car = 1,

Bike,

Bus,

Truck,

Van

}

// Method to display fuel type based on vehicle

static void DisplayFuelType(VehicleType vehicle)

{

switch (vehicle)

{

case VehicleType.Car:

case VehicleType.Bike:

Console.WriteLine($"{vehicle} : Petrol");

break;

case VehicleType.Bus:

case VehicleType.Truck:

case VehicleType.Van:

Console.WriteLine($"{vehicle} : Diesel");

break;

default:

Console.WriteLine("Unknown vehicle type");

break;

}

}

static void Main(string[] args)

{

Console.WriteLine("Enter vehicle type (1 for Car, 2 for Bike, 3 for Bus, 4 for Truck, 5 for Van):");

int vehicleNo = Convert.ToInt32(Console.ReadLine());

if (Enum.IsDefined(typeof(VehicleType), vehicleNo))

{

VehicleType vehicle = (VehicleType)vehicleNo;

DisplayFuelType(vehicle);

}

else

{

Console.WriteLine("Invalid vehicle type entered!");

}

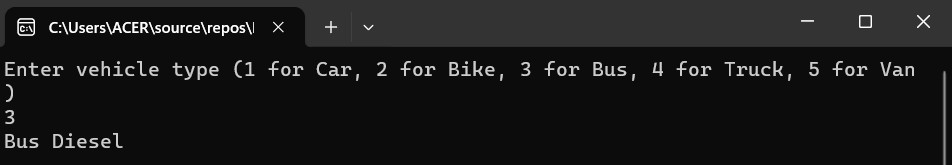
Console.ReadLine();

}

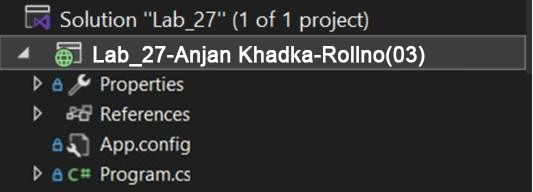
}

}

**Output**



**Lab 27: Write a C# program to create multidimensional array to store the marks of three student in different subjects. First student has marks of 3 subjects, second student has marks of 4 subjects and Third student has marks of 2 subjects, Display the subject marks and average marks for each student**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_27

{

class Program

{

static void Main(string[] args)

{

int[][] studentMarks = new int[3][];

// Assigning marks

studentMarks[0] = new int[] { 85, 90, 78 }; // 3 subjects

studentMarks[1] = new int[] { 75, 88, 92, 80 }; // 4 subjects

studentMarks[2] = new int[] { 90, 87 }; // 2 subjects

// Displaying marks and average for each student

for (int i = 0; i < studentMarks.Length; i++)

{

Console.WriteLine($"Student {i + 1} marks:");

int total = 0;

for (int j = 0; j < studentMarks[i].Length; j++)

{

Console.WriteLine($" Subject {j + 1}: {studentMarks[i][j]}");

total += studentMarks[i][j];

}

double average = (double)total / studentMarks[i].Length;

Console.WriteLine($" Average Marks: {average:F2}\n");

}

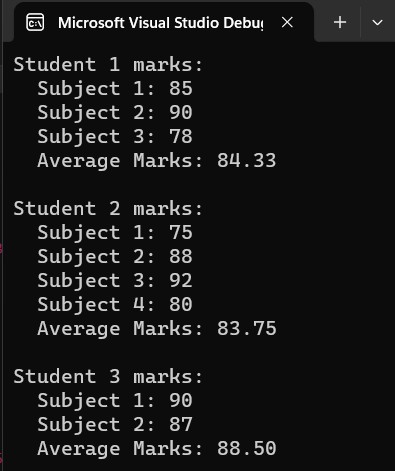
Console.ReadLine(); // Keeps console open

}

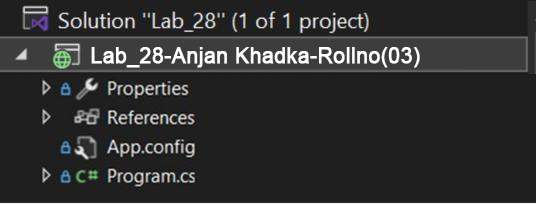
}

}

# Output



**Lab 28 : Write a C# program to achieve dynamic binding using virtual method in C#**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_28

{

class Animal

{

// Virtual method - can be overridden in derived classes

public virtual void Speak()

{

Console.WriteLine("The animal makes a sound.");

}

}

class Dog : Animal

{

// Override the Speak method

public override void Speak()

{

Console.WriteLine("The dog barks.");

}

}

class Cat : Animal

{

// Override the Speak method

public override void Speak()

{

Console.WriteLine("The cat meows.");

}

}

class Program

{

static void Main()

{

// Base class reference to derived class objects

Animal myAnimal;

myAnimal = new Dog();

myAnimal.Speak();

myAnimal = new Cat();

myAnimal.Speak();

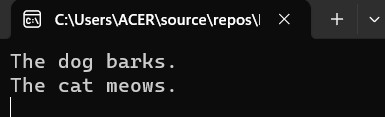
Console.ReadLine();

}

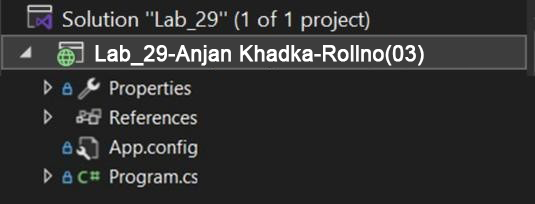
}

}

# Output



**Lab 29: Write a C# program to select odd and divisible by 3 number from list of numbers (1-30) using LINQ query.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_29

{

class Program

{

static void Main(string[] args)

{

// Generate numbers from 1 to 30

List<int> numbers = Enumerable.Range(1, 30).ToList();

// Use LINQ to select numbers that are odd and divisible by 3

var filteredNumbers = numbers.Where(n => n % 2 != 0 && n % 3 == 0);

Console.WriteLine("Odd numbers divisible by 3 (from 1 to 30):");

foreach (var num in filteredNumbers)

{

Console.Write(num + " ");

}

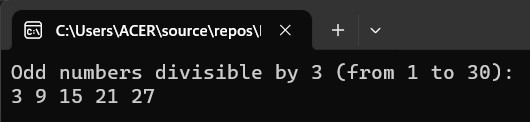
Console.ReadLine();

}

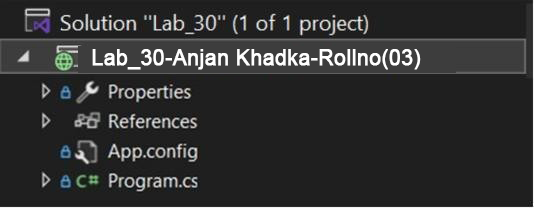
}

}

# Output



**Lab 30: Write a C# program to achieve dynamic binding using abstract method.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_30

{

// Abstract class

abstract class Shape

{

// Abstract method: must be overridden in derived classes

public abstract void Draw();

}

// Derived class Circle

class Circle : Shape

{

public override void Draw()

{

Console.WriteLine("Drawing a Circle.");

}

}

// Derived class Rectangle

class Rectangle : Shape

{

public override void Draw()

{

Console.WriteLine("Drawing a Rectangle.");

}

}

class Program

{

static void Main()

{

// Base class reference to derived class object

Shape shape;

shape = new Circle();

shape.Draw();

shape = new Rectangle();

shape.Draw();

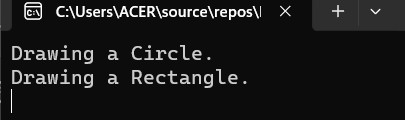
Console.ReadLine();

}

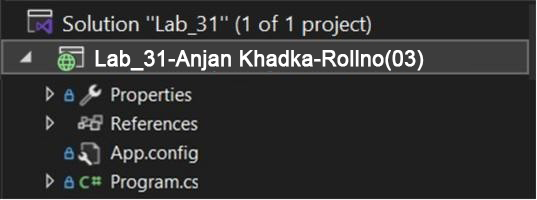
}

}

# Output



**Lab 31: Write a C# program to call member function and constructor of parent class using base keyword.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_31

{

class Parent

{

public Parent()

{

Console.WriteLine("Parent constructor called");

}

public void Display()

{

Console.WriteLine("Display method of Parent class");

}

}

class Child : Parent

{

public Child() : base()

{

Console.WriteLine("Child constructor called");

}

public void Show()

{

base.Display();

}

}

class Program

{

static void Main()

{

Child obj = new Child();

obj.Show();

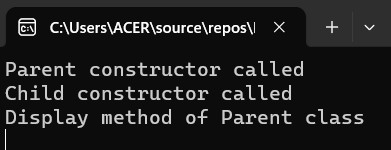
Console.ReadLine();

}

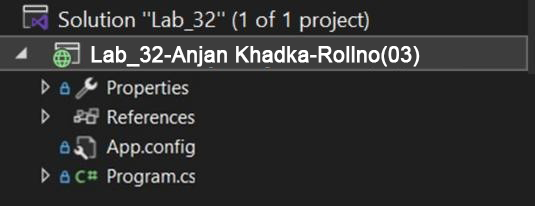
}

}

# Output



**Lab 32: Write a simple program to add and subtract two digit using multicast delegates**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_32

{

// Delegate declaration

delegate void Operation(int a, int b);

class Program

{

static void Add(int a, int b)

{

Console.WriteLine("Addition: " + (a + b));

}

static void Subtract(int a, int b)

{

Console.WriteLine("Subtraction: " + (a - b));

}

static void Main()

{

// Multicast delegate

Operation op = Add;

op += Subtract;

// Invoke delegate

op(10, 5);

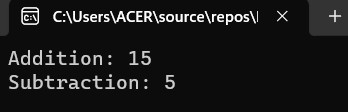
Console.ReadLine();

}

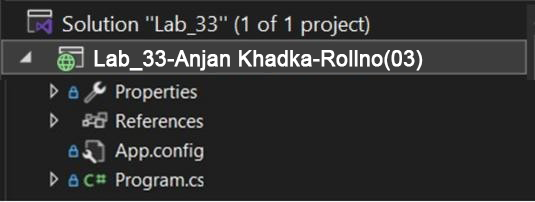
}

}

# Output



**Lab 33: Write a C# program which stores values in two struct, Department and college. it uses two function to display the data contained in department and college structure**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_33

{

struct Department

{

public string DeptName;

public int DeptCode;

public void DisplayDepartment()

{

Console.WriteLine("Department Name: " + DeptName);

Console.WriteLine("Department Code: " + DeptCode);

}

}

struct College

{

public string CollegeName;

public string Location;

public void DisplayCollege()

{

Console.WriteLine("College Name: " + CollegeName);

Console.WriteLine("Location: " + Location);

}

}

class Program

{

static void Main()

{

Department dept = new Department();

dept.DeptName = "Computer Science";

dept.DeptCode = 101;

College col = new College();

col.CollegeName = "National College";

col.Location = "Kathmandu";

dept.DisplayDepartment();

col.DisplayCollege();

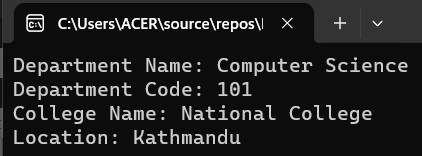
Console.ReadLine();

}

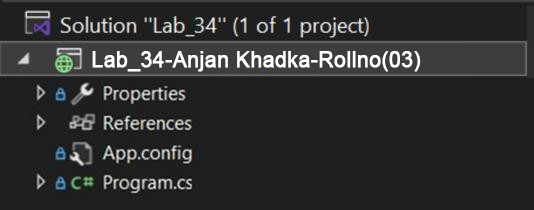
}

}

# Output



**Lab 34: Write a simple program to create generic class with generic constructor, generic member variable, generic property and generic method.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_34

{

class GenericClass<T>

{

private T value;

public GenericClass(T val)

{

value = val;

}

public T Data

{

get { return value; }

set { this.value = value; }

}

public void Display(T input)

{

Console.WriteLine("Value: " + input);

}

}

class Program

{

static void Main()

{

GenericClass<int> obj1 = new GenericClass<int>(10);

Console.WriteLine("Property: " + obj1.Data);

obj1.Display(25);

GenericClass<string> obj2 = new GenericClass<string>("Hello");

Console.WriteLine("Property: " + obj2.Data);

obj2.Display("World");

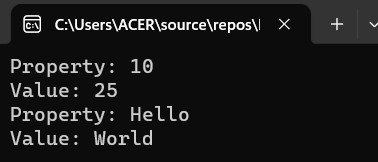
Console.ReadLine();

}

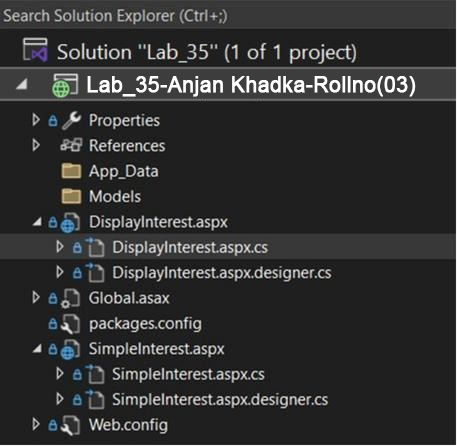
}

}

# Output



**Lab 35. Write a program to create form for calculating simple interest in one ASP.NET page and display the simple interest in another page of ASP.NET.**



# Code :

**SimpleInterest.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="SimpleInterest.aspx.cs" Inherits="Lab\_35.SimpleInterest" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Simple Interest Calculator</title>

<style type="text/css">

.auto-style3 { width: 186px; height: 26px; }

.auto-style5 { height: 26px; }

.auto-style7 { width: 186px; height: 29px; }

.auto-style8 { height: 29px; }

</style>

</head>

<body>

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

<div>

<table style="width: 33%;">

<tr>

<td colspan="2">Simple Interest</td>

</tr>

<tr>

<td class="auto-style3">

<asp:Label ID="Label1" runat="server" Text="Principal"></asp:Label>

</td>

<td class="auto-style5">

<asp:TextBox ID="txtPrincipal" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td class="auto-style7">

<asp:Label ID="Label2" runat="server" Text="Time"></asp:Label>

</td>

<td class="auto-style8">

<asp:TextBox ID="txtTime" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td>

<asp:Label ID="Label3" runat="server" Text="Rate"></asp:Label>

</td>

<td>

<asp:TextBox ID="txtRate" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td colspan="2">

<asp:Button ID="btnCalculate" runat="server" Text="Calculate" OnClick="btnCalculate\_Click"/>

</td>

</tr>

</table>

</div>

</form>

</body>

</html>

**SimpleInterest.aspx.cs**

using System;

using System.Web.UI;

namespace Lab\_35

{

public partial class SimpleInterest : System.Web.UI.Page

{

protected void btnCalculate\_Click(object sender, EventArgs e)

{

double principal = Convert.ToDouble(txtPrincipal.Text);

double rate = Convert.ToDouble(txtRate.Text);

double time = Convert.ToDouble(txtTime.Text);

double interest = (principal \* rate \* time) / 100;

Session["Interest"] = interest;

Response.Redirect("DisplayInterest.aspx");

}

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**DisplayInterest.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DisplayInterest.aspx.cs" Inherits="Lab\_35.DisplayInterest" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Interest Result</title>

</head>

<body>

<h2>Calculated Simple Interest:</h2>

<asp:Label ID="lblInterest" runat="server"></asp:Label>

</body>

</html>

**DisplayInterest.aspx.cs**

using System;

using System.Web.UI;

namespace Lab\_35

{

public partial class DisplayInterest : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["Interest"] != null)

{

lblInterest.Text = Session["Interest"].ToString();

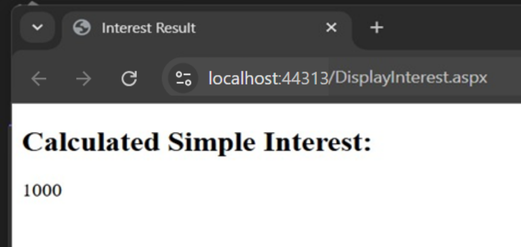
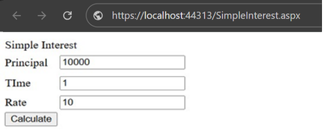
}

}

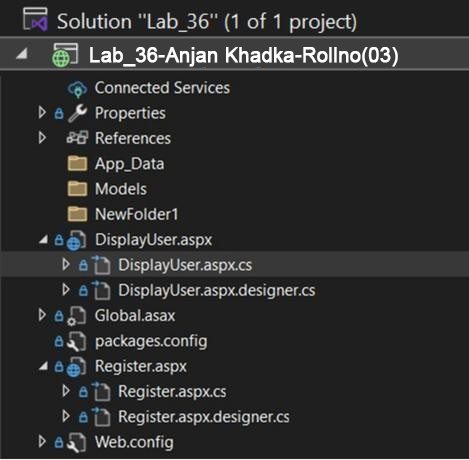
}

}

# Output



**Lab 36: Write a program to create user registration form in one ASP.NET web page and display filled data in another page.**



# Code :

## **Register.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Register.aspx.cs" Inherits="Lab\_36.Register" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>User Registration</title>

</head>

<body>

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

<div>

Name: <asp:TextBox ID="txtName" runat="server"></asp:TextBox><br />

Email: <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox><br />

Age: <asp:TextBox ID="txtAge" runat="server"></asp:TextBox><br />

<asp:Button ID="btnSubmit" runat="server" Text="Register" OnClick="btnSubmit\_Click" />

</div>

</form>

</body>

</html>

**Register.aspx.cs**

using System;

using System.Web.UI;

namespace Lab\_36

{

public partial class Register : System.Web.UI.Page

{

protected void btnSubmit\_Click(object sender, EventArgs e)

{

Session["Name"] = txtName.Text;

Session["Email"] = txtEmail.Text;

Session["Age"] = txtAge.Text;

Response.Redirect("DisplayUser.aspx");

}

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

## **DispalyUser.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DisplayUser.aspx.cs" Inherits="Lab\_36.DisplayUser" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>User Info</title>

</head>

<body>

<h2>Registered User Info:</h2>

<asp:Label ID="lblInfo" runat="server"></asp:Label>

</body>

</html>

**DisplayUser.aspx.cs**

using System;

using System.Web.UI;

namespace Lab\_36

{

public partial class DisplayUser : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["Name"] != null && Session["Email"] != null && Session["Age"] != null)

{

lblInfo.Text = $"Name: {Session["Name"]}<br />" +

$"Email: {Session["Email"]}<br />" +

$"Age: {Session["Age"]}";

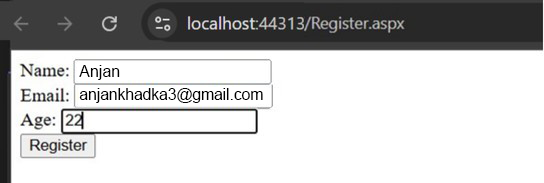
}

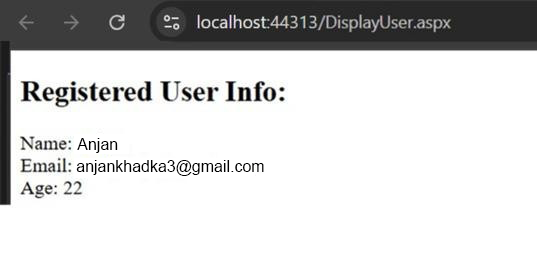
}

}

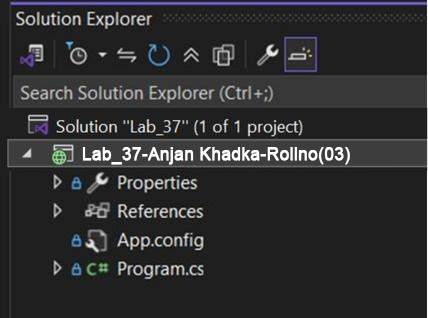
}

# Output





**Lab 37: Write a C# program create generic delegates and generic properties.**



**Code :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_37

{

// Generic delegate

delegate T GenericDelegate<T>(T val);

class GenericClass<T>

{

private T data;

// Generic property

public T Data

{

get { return data; }

set { data = value; }

}

// Generic method

public T Display(T input)

{

return input;

}

}

class Program

{

static void Main()

{

// Integer example

GenericClass<int> obj = new GenericClass<int>();

obj.Data = 100;

Console.WriteLine("Generic Property: " + obj.Data);

GenericDelegate<int> del = obj.Display;

Console.WriteLine("Generic Delegate Output: " + del(200));

// String example

GenericClass<string> strObj = new GenericClass<string>();

strObj.Data = "Hello";

Console.WriteLine("Generic Property: " + strObj.Data);

GenericDelegate<string> strDel = strObj.Display;

Console.WriteLine("Generic Delegate Output: " + strDel("World"));

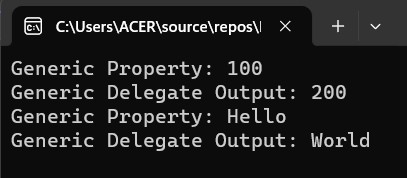
Console.ReadLine();

}

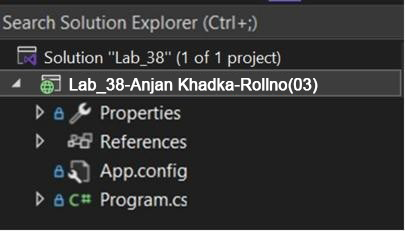
}

}

**Output**



**Lab 38: Write a c# program to achieve polymorphism using delegates.**



**Code :**

using System;

// Delegate declaration

delegate void Operation(int a, int b);

class Calculator

{

public void Add(int a, int b)

{

Console.WriteLine("Add: " + (a + b));

}

public void Subtract(int a, int b)

{

Console.WriteLine("Subtract: " + (a - b));

}

public void Multiply(int a, int b)

{

Console.WriteLine("Multiply: " + (a \* b));

}

}

class Program

{

static void Main()

{

Calculator calc = new Calculator();

Operation op;

// Using delegate to call Add

op = calc.Add;

op(10, 5);

// Using delegate to call Subtract

op = calc.Subtract;

op(10, 5);

// Using delegate to call Multiply

op = calc.Multiply;

op(10, 5);

Console.ReadLine();

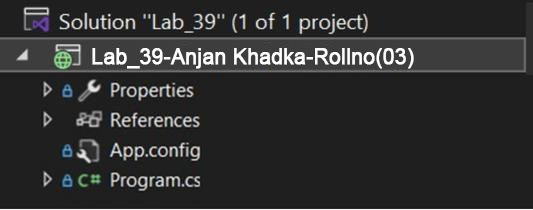
}

}

# Output



**Lab 39: Write a program to read an input string from the user and write the vowels of that string in VOWEL.TXT and consonants in CONSONANT.TXT**



**Code :**

using System;

using System.IO;

namespace Lab\_39

{

class Program

{

static void Main()

{

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

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

// Create StreamWriter objects for vowels and consonants

using (StreamWriter vowelWriter = new StreamWriter("VOWEL.TXT"))

using (StreamWriter consonantWriter = new StreamWriter("CONSONANT.TXT"))

{

foreach (char c in input)

{

if ("aeiou".Contains(c))

vowelWriter.Write(c);

else if (char.IsLetter(c))

consonantWriter.Write(c);

}

}

Console.WriteLine("Data written to VOWEL.TXT and CONSONANT.TXT.");

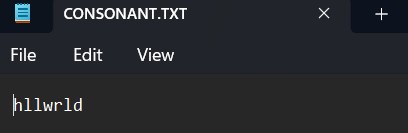
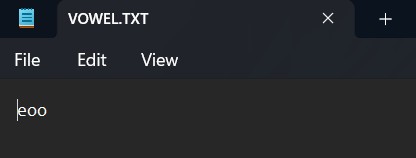
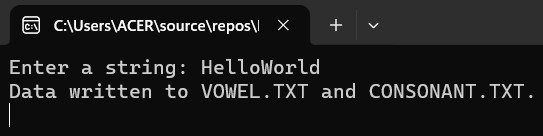
Console.ReadLine();

}

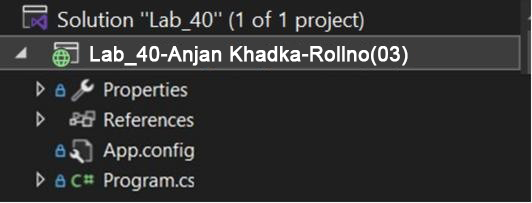
}

}

# Output



**Lab 40: Create a C# program that takes a sentence as input from User and capitalizes the first letter of each word and write to output.txt.**



**Code :**

using System;

using System.IO;

using System.Globalization;

namespace Lab\_40

{

class Program

{

static void Main()

{

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

string sentence = Console.ReadLine();

// Convert the sentence to Title Case

TextInfo ti = CultureInfo.CurrentCulture.TextInfo;

string result = ti.ToTitleCase(sentence.ToLower());

// Write the formatted sentence to a file

File.WriteAllText("output.txt", result);

Console.WriteLine("Formatted sentence written to output.txt.");

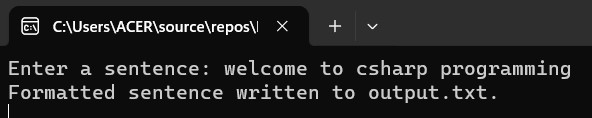
Console.ReadLine();

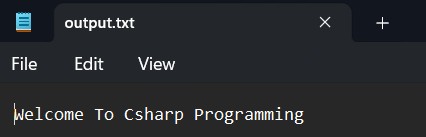
}

}

}

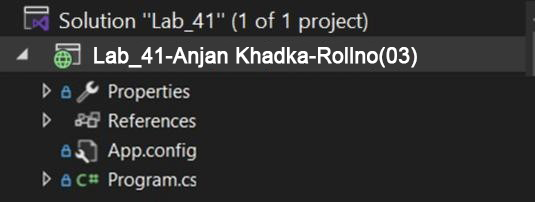
# Output





**Lab 41: Write a C# program to connect database swastikDB and insert 5 student record in student table with fields(id,name,email,gender) and display student record**

**whose gender is “female”**



**Code :**

using System;

using System.Data;

using System.Data.SqlClient;

namespace Lab\_41

{

class Program

{

static void Main(string[] args)

{

Student std = new Student();

// Input 5 students

for (int i = 0; i < 5; i++)

{

Console.WriteLine($"Student: {i + 1}");

Console.Write("Enter Name: ");

string name = Console.ReadLine();

Console.Write("Enter Email: ");

string email = Console.ReadLine();

Console.Write("Enter Gender: ");

string gender = Console.ReadLine();

std.CreateStudent(name, email, gender);

}

// Retrieve students with Gender = "female"

DataTable dt = std.GetStudent();

Console.WriteLine("\nId\tName\t\tEmail\t\tGender");

foreach (DataRow row in dt.Rows)

{

Console.WriteLine($"{row["Id"]}\t{row["Name"]}\t{row["Email"]}\t{row["Gender"]}");

}

Console.ReadLine();

}

}

public class Student

{

// Insert a new student into the database

public void CreateStudent(string name, string email, string gender)

{

string connectionString = @"Data Source=(localdb)\mssqllocaldb;Database=swastikDB;Integrated Security=True;";

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("INSERT INTO Student (Name, Email, Gender) VALUES (@a, @b, @c)", con);

cmd.Parameters.AddWithValue("@a", name);

cmd.Parameters.AddWithValue("@b", email);

cmd.Parameters.AddWithValue("@c", gender);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

Console.WriteLine("Student Saved Successfully");

}

// Retrieve students where Gender = "female"

public DataTable GetStudent()

{

string connectionString = @"Data Source=(localdb)\mssqllocaldb;Database=swastikDB;Integrated Security=True;";

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("SELECT \* FROM Student WHERE Gender = @gender", con);

cmd.Parameters.AddWithValue("@gender", "female");

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

return dt;

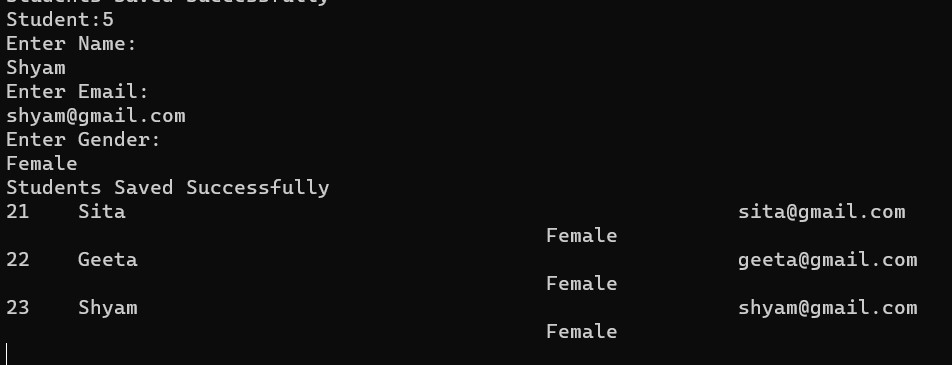
}

}

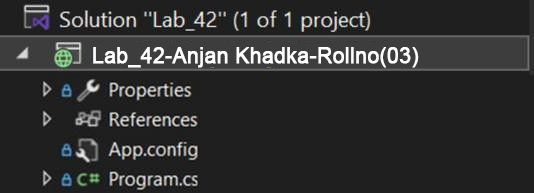
}

}

# Output



**Lab 42: Write a C# program to add Two Box Volume using the binary operator.**



**Code :**

using System;

namespace Lab\_42

{

class Box

{

public int length, width, height;

public Box(int l, int w, int h)

{

length = l;

width = w;

height = h;

}

public int Volume()

{

return length \* width \* height;

}

// Overload + operator to combine two boxes

public static Box operator +(Box b1, Box b2)

{

return new Box(

b1.length + b2.length,

b1.width + b2.width,

b1.height + b2.height

);

}

}

class Program

{

static void Main()

{

Box box1 = new Box(2, 3, 4);

Box box2 = new Box(1, 2, 3);

Box box3 = box1 + box2;

Console.WriteLine("Combined Volume: " + box3.Volume());

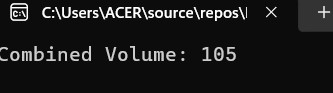
Console.ReadLine();

}

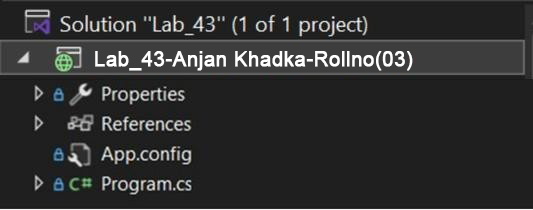
}

}

# Output



**Lab 43: Write a C# program to create a class Time which represents time.The class should have three fields for hours, minutes and seconds. It should have constructor to initialize hours, minutes and seconds and method displayTime() to print current time.Overload following operators.**



**Code :**

using System;

namespace Lab\_43

{

class Time

{

public int hours, minutes, seconds;

public Time(int h, int m, int s)

{

hours = h;

minutes = m;

seconds = s;

}

public void DisplayTime()

{

Console.WriteLine($"{hours:D2}:{minutes:D2}:{seconds:D2}");

}

// Overload + operator to add two Time objects

public static Time operator +(Time t1, Time t2)

{

int sec = t1.seconds + t2.seconds;

int min = t1.minutes + t2.minutes + sec / 60;

int hr = t1.hours + t2.hours + min / 60;

return new Time(hr % 24, min % 60, sec % 60);

}

// Overload == operator

public static bool operator ==(Time t1, Time t2)

{

return (t1.hours == t2.hours && t1.minutes == t2.minutes && t1.seconds == t2.seconds);

}

// Overload != operator

public static bool operator !=(Time t1, Time t2)

{

return !(t1 == t2);

}

public override bool Equals(object obj)

{

Time t = (Time)obj;

return this == t;

}

public override int GetHashCode()

{

return (hours, minutes, seconds).GetHashCode();

}

}

class Program

{

static void Main()

{

Time t1 = new Time(2, 45, 50);

Time t2 = new Time(1, 20, 30);

Time t3 = t1 + t2;

Console.Write("Time 1: ");

t1.DisplayTime();

Console.Write("Time 2: ");

t2.DisplayTime();

Console.Write("Time 1 + Time 2: ");

t3.DisplayTime();

Console.WriteLine("Time 1 == Time 2? " + (t1 == t2));

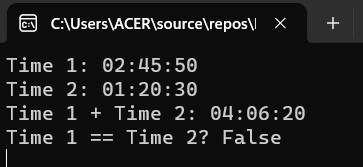
Console.ReadLine();

}

}

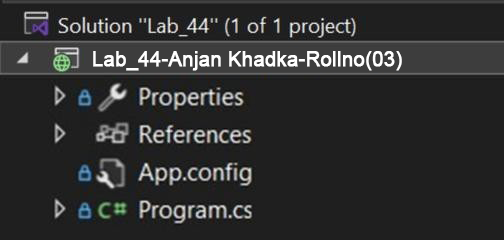
}

# Output



**Lab 44: Write a C# program to perform (CRUD) Operation from given table**

**(tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) gender and salary).**



**Code :**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace Lab\_44

{

class Program

{

static void Main(string[] args)

{

Employee emp = new Employee();

for (int i = 0; i < 5; i++)

{

Console.WriteLine($"Employee#:{i + 1}");

Console.WriteLine("Enter Name:");

string name = Console.ReadLine();

Console.WriteLine("Enter Address:");

string address = Console.ReadLine();

Console.WriteLine("Enter Gender:");

string gender = Console.ReadLine();

Console.WriteLine("Enter Salary:");

decimal salary = Convert.ToDecimal(Console.ReadLine());

emp.CreateEmployee(name, address, gender, salary);

}

DataTable dt = emp.GetAllEmployee();

for (int i = 0; i < dt.Rows.Count; i++)

{

Console.WriteLine($"{dt.Rows[i]["Id"]} {dt.Rows[i]["Name"]} {dt.Rows[i]["Address"]}");

}

Console.ReadLine();

}

}

public class Employee

{

private string connectionString = "Data Source=(localdb)\\mssqllocaldb; Database=SW\_DB; Integrated Security=true;";

public void CreateEmployee(string name, string address, string gender, decimal salary)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("INSERT INTO tblEmployee(Name, Address, Gender, Salary) VALUES (@a, @b, @c, @d)", con);

cmd.Parameters.AddWithValue("@a", name);

cmd.Parameters.AddWithValue("@b", address);

cmd.Parameters.AddWithValue("@c", gender);

cmd.Parameters.AddWithValue("@d", salary);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

Console.WriteLine("Employee Saved Successfully");

}

}

public void UpdateEmployee(int id, string name, string address, string gender, decimal salary)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(

"UPDATE tblEmployee SET Name=@a, Address=@b, Gender=@c, Salary=@d WHERE Id=@e", con);

cmd.Parameters.AddWithValue("@a", name);

cmd.Parameters.AddWithValue("@b", address);

cmd.Parameters.AddWithValue("@c", gender);

cmd.Parameters.AddWithValue("@d", salary);

cmd.Parameters.AddWithValue("@e", id);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

Console.WriteLine("Employee Updated Successfully");

}

}

public void DeleteEmployee(int id)

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("DELETE FROM tblEmployee WHERE Id=@id", con);

cmd.Parameters.AddWithValue("@id", id);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

Console.WriteLine("Employee Deleted Successfully");

}

}

public DataTable GetAllEmployee()

{

using (SqlConnection con = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand("SELECT \* FROM tblEmployee", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

return dt;

}

}

}

}

Output

