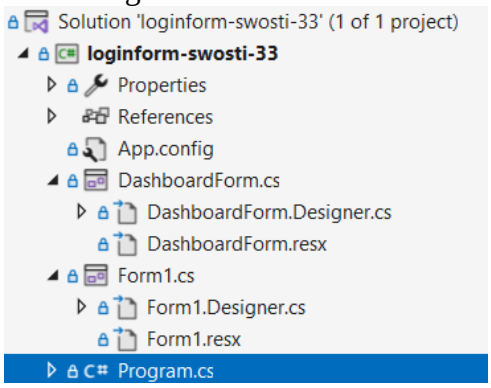
1. Write a program to create a C# Windows Forms application that implements a login form with a username and password field. The form should validate the credentials (username: "admin", password: "1234"), display a success or failure message, and clear the fields on failure. After three failed attempts,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



Source Code :

From1.cs(loginform) :

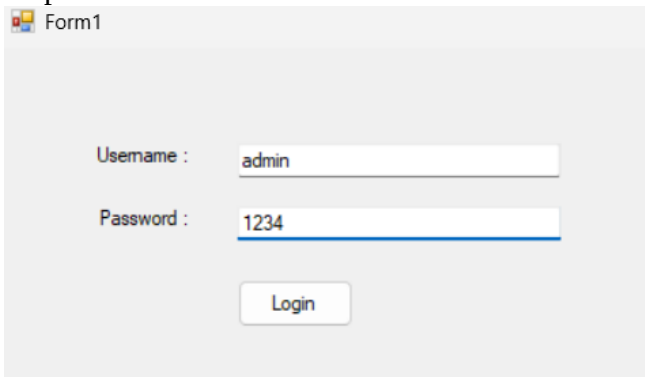
```
using System;
using System.Windows.Forms;
namespace loginform-swosti-33
{
    public partial class Form1 : Form
    {
        int loginAttempts = 0;
        public Form1()
        {
            InitializeComponent();
        }
        private void btnLogin_Click(object sender, EventArgs e)
        {
            string username = txtUsername.Text;
            string password = txtPassword.Text;
            if (username == "admin" && password == "1234")
            {
                lblMessage.Text = "Login Successful!";
                lblMessage.ForeColor = System.Drawing.Color.Green;
                DashboardForm dashboard = new DashboardForm();
                dashboard.Show();
                this.Hide();
   } else  {
                loginAttempts++;
                lblMessage.Text = "Invalid credentials.";
                lblMessage.ForeColor = System.Drawing.Color.Red;
                txtUsername.Clear();
                txtPassword.Clear();
                if (loginAttempts >= 3)
                {
                    btnLogin.Enabled = false;
                    lblMessage.Text = "Too many failed attempts. Login
      disabled.";
                }  }  }  }  }
```

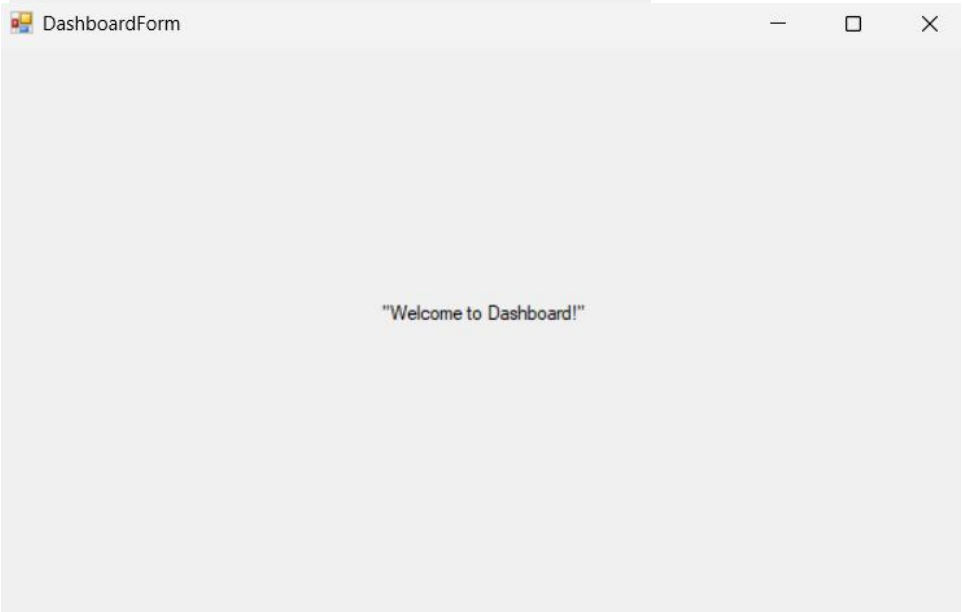
DashboardForm.cs :

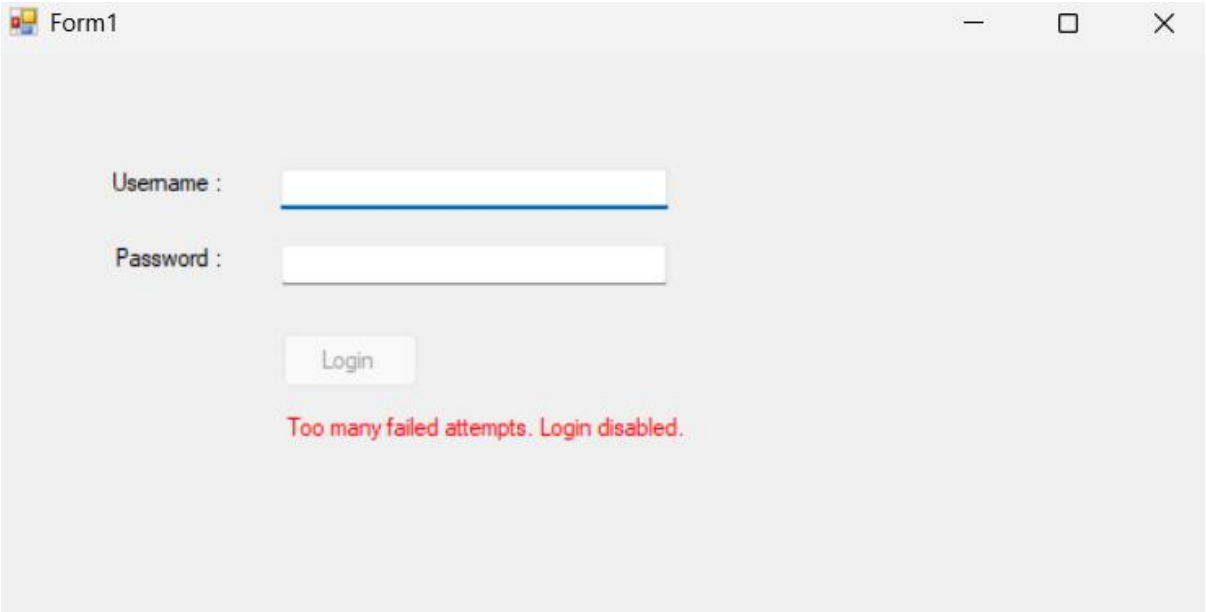
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace loginform-swosti-33
{
    public partial class DashboardForm: Form
    {
        public DashboardForm()
        {
            InitializeComponent();
            this.FormClosed += DashboardForm_FormClosed;
        }
        private void DashboardForm_Load(object sender, EventArgs e)
        {
        }
       private void DashboardForm_FormClosed(object
      sender,FormClosedEventArgs e)
        {
            Application.Exit();
        }  }  }
```

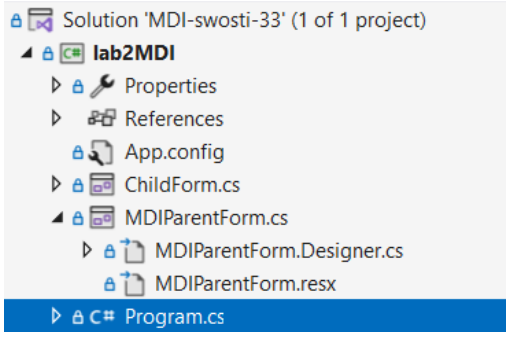
Output :






Form1
Username :
Password :
Login
Too many failed attempts. Login disabled.

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.

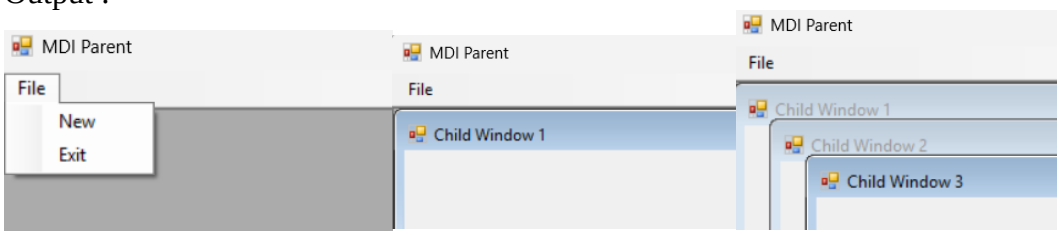


Source Code :

MDI ParentForm.cs :

```
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 lab2MDI
{
    public partial class Form1: Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {   }
        private void newToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ChildForm child = new ChildForm();
            child.MdiParent = this;
            child.Text = "Child Window " + this.MdiChildren.Length;
            child.Show();
        }
        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
} }   }
```

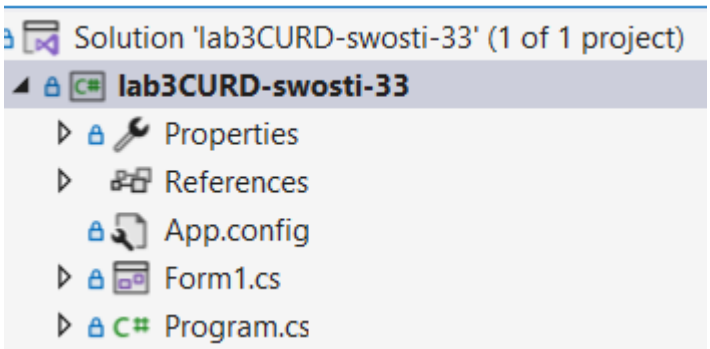
Output :



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.



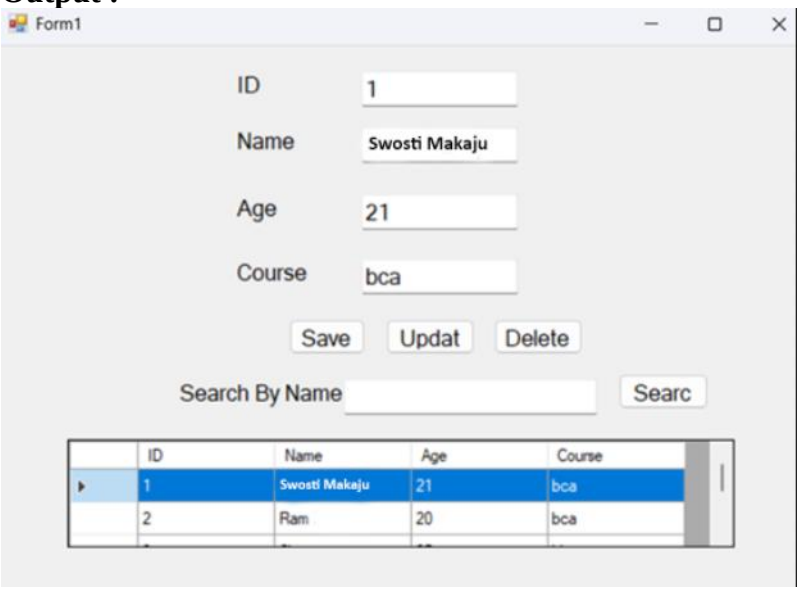
Source Code :

```
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 lab3
{
    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)
        {
```

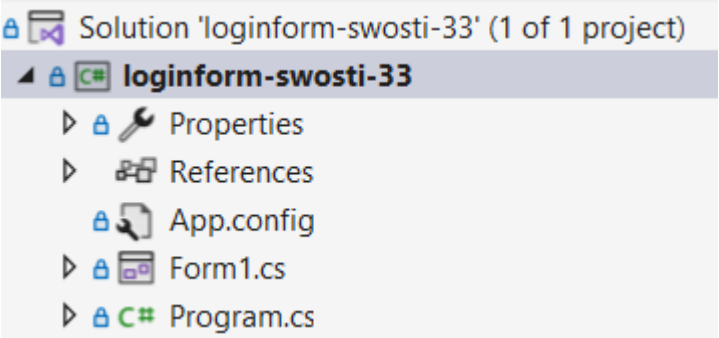
```
    LoadGrid();

}

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 :



4. Write a program to create a C# Windows Forms application with a login form that validates user credentials from a SQL Server database. The form should have username and password fields, and a "Login" button. When the user clicks "Login", check the credentials against a Users table in the database. If valid, open a Dashboard Form and close the login form; otherwise, display an error message. After three failed attempts, disable the login button.



Source code :

```
using System;
using System.Data.SqlClient;
using System.Windows.Forms;
namespace loginform_swosti_33{
   public partial class LoginForm : Form{
      private int failedAttempts = 0;
      private const int MaxAttempts = 3;
      private string connectionString = "Server=.;Database=UserManagement;Integrated Security=True;";
      public LoginForm(){
         InitializeComponent();
         this.Load += LoginForm_Load;
         this.FormClosing += LoginForm_FormClosing;
         this.btnLogin.Click += btnLogin_Click;
      }
      private void LoginForm_Load(object sender, EventArgs e){
         if (!TestDatabaseConnection()){
            var result = MessageBox.Show("Failed to connect to database. Would you like to configure the
            connection
string?",
                                 "Database Connection Error",
                                 MessageBoxButtons.YesNo,
                                 MessageBoxIcon.Error);
            if (result == DialogResult.Yes){
               string newConnectionString = ShowInputDialog(
                  "Enter your SQL Server connection string:",
                  "Database Connection",
                  connectionString);
               if (!string.IsNullOrEmpty(newConnectionString)){
                  connectionString = newConnectionString;
                  if (!TestDatabaseConnection()){
                     MessageBox.Show("Still unable to connect to database. The application will now exit.",
                                 "Error",
                                 MessageBoxButtons.OK,
                                 MessageBoxIcon.Error);
                     Application.Exit();
                  }
               }
            }
            else{
               Application.Exit();
            }
```

```
        }
    }
    private void btnLogin_Click(object sender, EventArgs e){
        string username = txtUsername.Text.Trim();
        string password = txtPassword.Text;

        if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
        {
            MessageBox.Show("Please enter both username and password.", "Error",
          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return;
        }
        if (ValidateUser(username, password)){
            DashboardForm dashboard = new DashboardForm(username);
            dashboard.Show();
            this.Hide();
        }
        else{
            failedAttempts++;
            if (failedAttempts >= MaxAttempts)
            {
                MessageBox.Show("Maximum login attempts reached. The login button has been disabled.",
                        "Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
                btnLogin.Enabled = false;
            }
            else{
                MessageBox.Show($"Invalid username or password. You have {MaxAttempts -
          failedAttempts} attempts
remaining.",
                         "Error",
                         MessageBoxButtons.OK,
                         MessageBoxIcon.Error);  }  }  }
    private void LoginForm_FormClosing(object sender, FormClosingEventArgs e){
        Application.Exit();
    }
    private static string ShowInputDialog(string text, string caption, string defaultValue = ""){
        Form prompt = new Form(){
            Width = 400,
            Height = 150,
            FormBorderStyle = FormBorderStyle.FixedDialog,
            Text = caption,
            StartPosition = FormStartPosition.CenterScreen
        };
        Label textLabel = new Label() { Left = 20, Top = 20, Text = text, Width = 350 };
        TextBox textBox = new TextBox() { Left = 20, Top = 50, Width = 350, Text = defaultValue };
        Button confirmation = new Button() { Text = "OK", Left = 280, Width = 80, Top = 80,
          DialogResult =
DialogResult.OK };
        confirmation.Click += (sender, e) => { prompt.Close(); };
        prompt.Controls.Add(textBox);
        prompt.Controls.Add(confirmation);
        prompt.Controls.Add(textLabel);
        prompt.AcceptButton = confirmation;
        return prompt.ShowDialog() == DialogResult.OK ? textBox.Text : defaultValue;
    }
    private bool TestDatabaseConnection(){
        try{
```

```
            using (SqlConnection connection = new SqlConnection(connectionString)){
                connection.Open();
                return true;
            }
        }
        catch (Exception ex){
            MessageBox.Show($"Connection failed: {ex.Message}\n\nCurrent connection
          string:\n{connectionString}",
                        "Database Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
            return false;
        }
    }
    private bool ValidateUser(string username, string password){
        try{
            using (SqlConnection connection = new SqlConnection(connectionString)){
                string query = "SELECT COUNT(*) FROM Users WHERE Username = @Username AND
          Password =
@Password";
                SqlCommand command = new SqlCommand(query, connection);
                command.Parameters.AddWithValue("@Username", username);
                command.Parameters.AddWithValue("@Password", password);
                connection.Open();
                int count = (int)command.ExecuteScalar();
                return count > 0;
            }
        }
        catch (Exception ex){
            MessageBox.Show($"Error validating user: {ex.Message}", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return false;
        }
}   }   }
```

DashboardForm.cs

```
using System;
using System.Windows.Forms;
namespace LoginApp{
    public partial class DashboardForm : Form{
        public DashboardForm(string username){
            InitializeComponent();
            lblWelcome.Text = $"Welcome, {username}!";
        }
        private void btnLogout_Click(object sender, EventArgs e){
            LoginForm loginForm = new LoginForm();
            loginForm.Show();
            this.Close();
        }
    }
}
```

LoginForm.designer.cs

```
namespace LoginApp{
    partial class LoginForm{
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing){
            if (disposing && (components != null)){
                components.Dispose();
            }
```

```
    base.Dispose(disposing);
}
#region Windows Form Designer generated code
private void InitializeComponent(){
    this.lblUsername = new System.Windows.Forms.Label();
    this.lblPassword = new System.Windows.Forms.Label();
    this.txtUsername = new System.Windows.Forms.TextBox();
    this.txtPassword = new System.Windows.Forms.TextBox();
    this.btnLogin = new System.Windows.Forms.Button();
    this.SuspendLayout();
    this.lblUsername.AutoSize = true;
    this.lblUsername.Location = new System.Drawing.Point(50, 50);
    this.lblUsername.Name = "lblUsername";
    this.lblUsername.Size = new System.Drawing.Size(58, 13);
    this.lblUsername.TabIndex = 0;
    this.lblUsername.Text = "Username:";
    this.lblPassword.AutoSize = true;
    this.lblPassword.Location = new System.Drawing.Point(50, 90);
    this.lblPassword.Name = "lblPassword";
    this.lblPassword.Size = new System.Drawing.Size(56, 13);
    this.lblPassword.TabIndex = 1;
    this.lblPassword.Text = "Password:";
    this.txtUsername.Location = new System.Drawing.Point(120, 47);
    this.txtUsername.Name = "txtUsername";
    this.txtUsername.Size = new System.Drawing.Size(150, 20);
    this.txtUsername.TabIndex = 1;
    this.txtPassword.Location = new System.Drawing.Point(120, 87);
    this.txtPassword.Name = "txtPassword";
    this.txtPassword.PasswordChar = '*';
    this.txtPassword.Size = new System.Drawing.Size(150, 20);
    this.txtPassword.TabIndex = 2;
    this.btnLogin.Location = new System.Drawing.Point(120, 130);
    this.btnLogin.Name = "btnLogin";
    this.btnLogin.Size = new System.Drawing.Size(75, 23);
    this.btnLogin.TabIndex = 3;
    this.btnLogin.Text = "Login";
    this.btnLogin.UseVisualStyleBackColor = true;
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(320, 200);
    this.Controls.Add(this.btnLogin);
    this.Controls.Add(this.txtPassword);
    this.Controls.Add(this.txtUsername);
    this.Controls.Add(this.lblPassword);
    this.Controls.Add(this.lblUsername);
    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
    this.MaximizeBox = false;
    this.Name = "LoginForm";
    this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
    this.Text = "Login";
    this.FormClosing += new
      System.Windows.Forms.FormClosingEventHandler(this.LoginForm_FormClosing);
    this.ResumeLayout(false);
    this.PerformLayout();

}
#endregion
private System.Windows.Forms.Label lblUsername;
private System.Windows.Forms.Label lblPassword;
private System.Windows.Forms.TextBox txtUsername;
```

```
        private System.Windows.Forms.TextBox txtPassword;
        private System.Windows.Forms.Button btnLogin;
    }
}
```

DashboardForm.Designer.cs

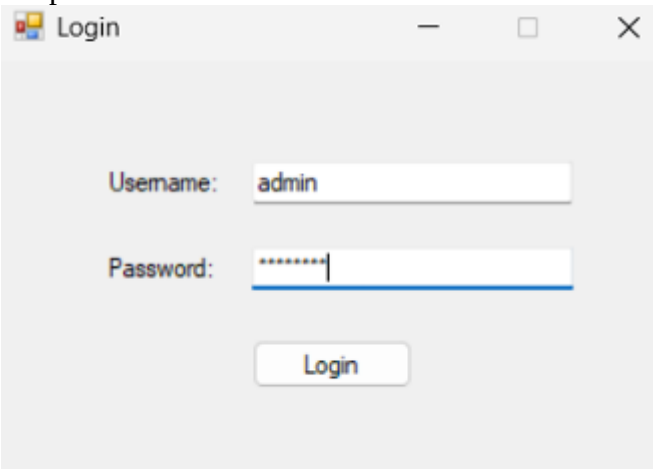
```
namespace LoginApp{
    partial class DashboardForm{
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing){
            if (disposing && (components != null)){
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent(){
            this.lblWelcome = new System.Windows.Forms.Label();
            this.btnLogout = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            this.lblWelcome.AutoSize = true;
            this.lblWelcome.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.lblWelcome.Location = new System.Drawing.Point(50, 50);
            this.lblWelcome.Name = "lblWelcome";
            this.lblWelcome.Size = new System.Drawing.Size(89, 24);
            this.lblWelcome.TabIndex = 0;
            this.lblWelcome.Text = "Welcome";
            this.btnLogout.Location = new System.Drawing.Point(50, 100);
            this.btnLogout.Name = "btnLogout";
            this.btnLogout.Size = new System.Drawing.Size(75, 23);
            this.btnLogout.TabIndex = 1;
            this.btnLogout.Text = "Logout";
            this.btnLogout.UseVisualStyleBackColor = true;
            this.btnLogout.Click += new System.EventHandler(this.btnLogout_Click);
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(400, 300);
            this.Controls.Add(this.btnLogout);
            this.Controls.Add(this.lblWelcome);
            this.Name = "DashboardForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Dashboard";
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.Label lblWelcome;
        private System.Windows.Forms.Button btnLogout;
    }
}
```

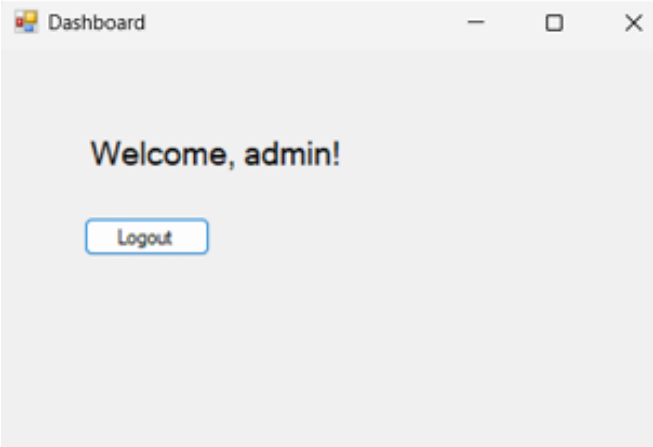
Program.cs

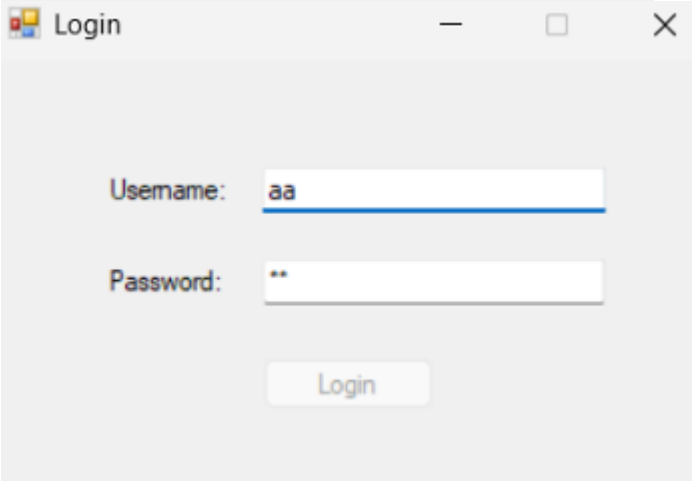
```
using System;
using System.Windows.Forms;
namespace LoginApp{
    static class Program{
        [STAThread]
        static void Main(){
```

```
        Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new LoginForm());
}
}
}
```

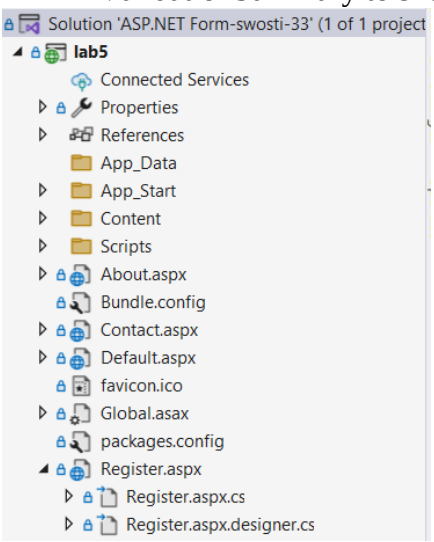
Output :







5. 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.



Source Code :

Register.aspx :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Register.aspx.cs"
       Inherits="YourNamespace.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 style="width: 400px; margin: 50px auto; font-family: sans-
       serif;">
            <h2>User Registration</h2>

            <asp:ValidationSummary ID="ValidationSummary1" runat="server"
       ForeColor="Red" HeaderText="Please fix the following errors:" />

            <label>Full Name:</label><br />
            <asp:TextBox ID="txtFullName" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvFullName" runat="server"
                ControlToValidate="txtFullName"
                ErrorMessage="Full Name is required."
                ForeColor="Red" Display="Dynamic" /><br /><br />

            <label>Email:</label><br />
            <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvEmail" runat="server"
                ControlToValidate="txtEmail"
                ErrorMessage="Email is required."
                ForeColor="Red" Display="Dynamic" />
            <asp:RegularExpressionValidator ID="revEmail" runat="server"
                ControlToValidate="txtEmail"
                ErrorMessage="Enter a valid email address."
```

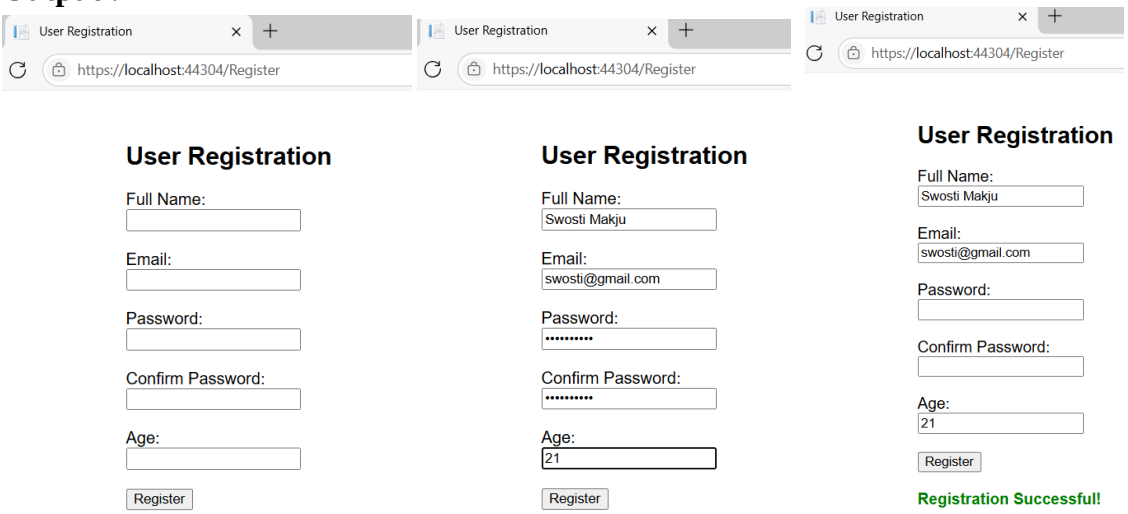
```
                ValidationExpression="^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$"
                ForeColor="Red" Display="Dynamic" /><br /><br />

            <label>Password:</label><br />
            <asp:TextBox ID="txtPassword" runat="server"
        TextMode="Password"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvPassword" runat="server"
                ControlToValidate="txtPassword"
                ErrorMessage="Password is required."
                ForeColor="Red" Display="Dynamic" />
            <asp:RegularExpressionValidator ID="revPassword" runat="server"
                ControlToValidate="txtPassword"
                ErrorMessage="Password must be at least 6 characters."
                ValidationExpression=".{6,}"
                ForeColor="Red" Display="Dynamic" /><br /><br />

            <label>Confirm Password:</label><br />
            <asp:TextBox ID="txtConfirmPassword" runat="server"
        TextMode="Password"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvConfirmPassword"
        runat="server"
                ControlToValidate="txtConfirmPassword"
                ErrorMessage="Please confirm your password."
                ForeColor="Red" Display="Dynamic" />
            <asp:CompareValidator ID="cvPasswords" runat="server"
                ControlToCompare="txtPassword"
                ControlToValidate="txtConfirmPassword"
                ErrorMessage="Passwords do not match."
                ForeColor="Red" Display="Dynamic" /><br /><br />

            <label>Age:</label><br />
            <asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvAge" runat="server"
                ControlToValidate="txtAge"
                ErrorMessage="Age is required."
                ForeColor="Red" Display="Dynamic" />
            <asp:RangeValidator ID="rvAge" runat="server"
                ControlToValidate="txtAge"
                ErrorMessage="Age must be between 18 and 99."
                MinimumValue="18"
                MaximumValue="99"
                Type="Integer"
                ForeColor="Red" Display="Dynamic" /><br /><br />

            <asp:Button ID="btnRegister" runat="server" Text="Register"
        OnClick="btnRegister_Click" /><br /><br />

            <asp:Label ID="lblSuccess" runat="server" ForeColor="Green" Font-
        Bold="true"></asp:Label>
          </div>
      </form>
</body>
</html>
```

Register.aspx.cs :

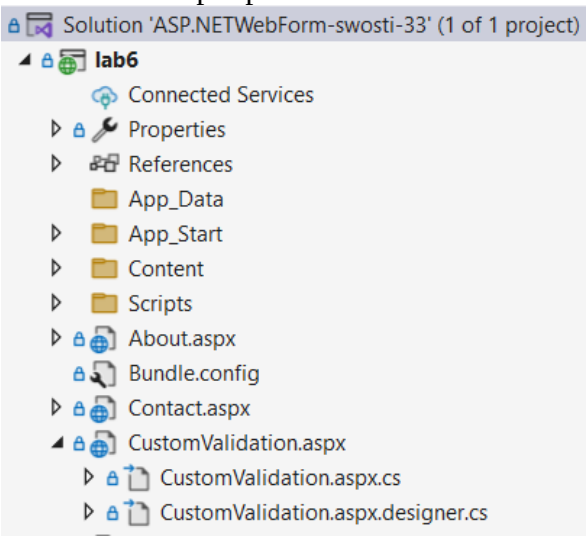
```
using System;

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

```
        {
            lblSuccess.Text = "";
        }

        protected void btnRegister_Click(object sender, EventArgs e)
        {
            if (Page.IsValid)
            {
                lblSuccess.Text = "Registration Successful!";
            }
} } }
```

Output :



6. 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.



Source Code :

CustomValidation.aspx :

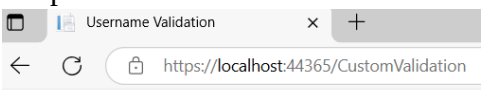
```
<%@ Page Language="C#" AutoEventWireup="true"
        CodeBehind="CustomValidation.aspx.cs"
        Inherits="UsernameValidationApp.CustomValidation" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Username Validation</title>
</head>
<body>
    <form id="form1" runat="server">
        <div style="font-family:sans-serif; margin:50px;">
            <h2>Username Validation</h2>

            <asp:Label ID="lblUsername" runat="server" Text="Enter Username:
      "></asp:Label>
            <asp:TextBox ID="txtUsername" runat="server"></asp:TextBox>

            <asp:CustomValidator ID="cvUsername" runat="server"
                ControlToValidate="txtUsername"
                ErrorMessage="Only letters and numbers are allowed!"
                ForeColor="Red"
                OnServerValidate="cvUsername_ServerValidate"
                Display="Dynamic">
            </asp:CustomValidator>

            <br /><br />
            <asp:Button ID="btnSubmit" runat="server" Text="Submit"
      OnClick="btnSubmit_Click" />

            <br /><br />
            <asp:Label ID="lblMessage" runat="server" Font-
      Bold="true"></asp:Label>
        </div>
    </form>
</body>
```

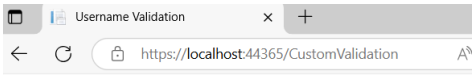
```
</html>
```

CustomValidation.aspx.cs :

```
using System;
using System.Text.RegularExpressions;

namespace UsernameValidationApp
{
    public partial class CustomValidation : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            lblMessage.Text = "";
        }

        protected void cvUsername_ServerValidate(object source,
      System.Web.UI.WebControls.ServerValidateEventArgs args)
        {
            string pattern = @"^[a-zA-Z0-9]+$";
            args.IsValid = Regex.IsMatch(args.Value, pattern);
        }

        protected void btnSubmit_Click(object sender, EventArgs e)
        {
            if (Page.IsValid)
            {
                lblMessage.ForeColor = System.Drawing.Color.Green;
                lblMessage.Text = "Username is valid. Success!";
            }
            else
            {
                lblMessage.ForeColor = System.Drawing.Color.Red;
                lblMessage.Text = "Validation failed. Please correct the
      errors.";
            }
        }    }    }
```

Output :





Username Validation

Enter Username:

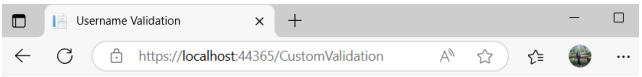
Submit

Username Validation

Enter Username: Swosti123

Submit

Username is valid. Success!



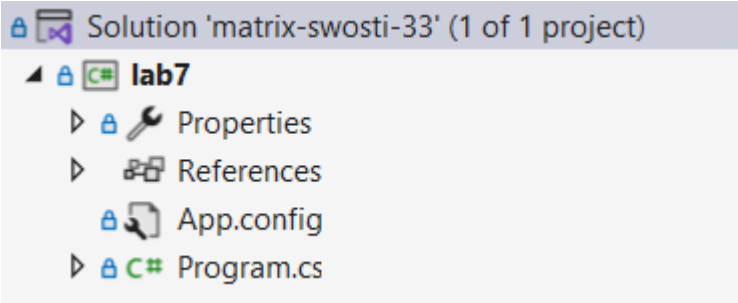
Username Validation

Enter Username: Swosti@122 Only letters and numbers are allowed!

Submit

Validation failed. Please correct the errors.

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



Source Code :

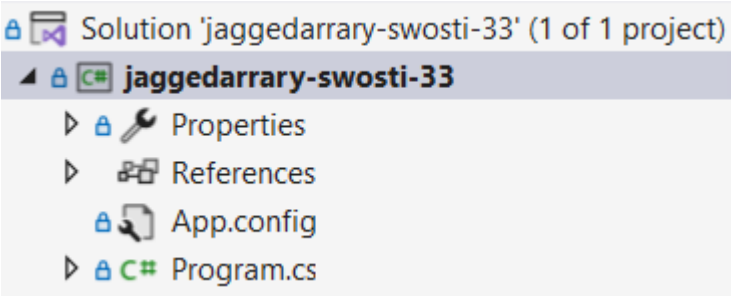
```
namespace lab7
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter rows: ");
            int m = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter columns: ");
            int n = Convert.ToInt32(Console.ReadLine());
            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 :

```
Enter rows: 2
Enter columns: 3
Enter elements of First Matrix:
First[0,0]: 5
First[0,1]: 3
First[0,2]: 1
First[1,0]: 6
First[1,1]: 8
First[1,2]: 4
Enter elements of Second Matrix:
Second[0,0]: 1
Second[0,1]: 2
Second[0,2]: 3
Second[1,0]: 4
Second[1,1]: 7
Second[1,2]: 6
Resultant Matrix:
6       5       4
10      15      10
```

8. 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.



Source Code :

```
using System;
class MatrixAdditionJagged
{
    static void Main()
    {
        Console.Write("Enter number of rows: ");
        int rows = int.Parse(Console.ReadLine());
        int[][] matrixA = new int[rows][];
        int[][] matrixB = new int[rows][];
        int[][] resultMatrix = new int[rows][];
        for (int i = 0; i < rows; i++)
        {
            Console.Write($"Enter number of columns for row {i + 1}: ");
            int cols = int.Parse(Console.ReadLine());

            matrixA[i] = new int[cols];
            matrixB[i] = new int[cols];
            resultMatrix[i] = new int[cols];

            Console.WriteLine($"Enter elements for Matrix A, Row {i + 1}:");
            for (int j = 0; j < cols; j++)
            {
                Console.Write($"A[{i}][{j}] = ");
                matrixA[i][j] = int.Parse(Console.ReadLine());
            }

            Console.WriteLine($"Enter elements for Matrix B, Row {i + 1}:");
            for (int j = 0; j < cols; j++)
            {
                Console.Write($"B[{i}][{j}] = ");
                matrixB[i][j] = int.Parse(Console.ReadLine());
                resultMatrix[i][j] = matrixA[i][j] + matrixB[i][j];
            }
        }
        Console.WriteLine("\nMatrix A:");
        PrintMatrix(matrixA);
        Console.WriteLine("\nMatrix B:");
        PrintMatrix(matrixB);
        Console.WriteLine("\nResult (A + B):");
        PrintMatrix(resultMatrix);
    }
    static void PrintMatrix(int[][] matrix)
    {   for (int i = 0; i < matrix.Length; i++)
        {
            foreach (int val in matrix[i])
            {
                Console.Write(val + "\t");
            }
            Console.WriteLine();
```

```
        }
    }
}
```

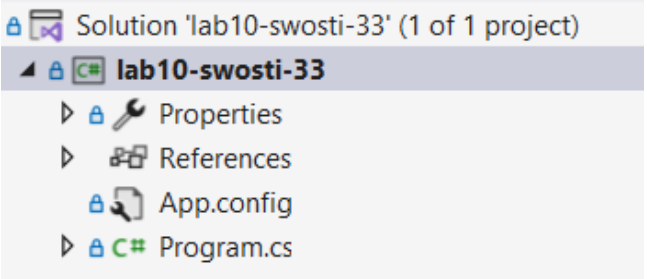
Output :

```
Microsoft Visual Studio Debu

A[0][1] = 2
A[0][2] = 3
Enter elements for Matrix B, Row 1:
B[0][0] = 4
B[0][1] = 5
B[0][2] = 6
Enter number of columns for row 2: 2
Enter elements for Matrix A, Row 2:
A[1][0] = 9
A[1][1] = 8
Enter elements for Matrix B, Row 2:
B[1][0] = 7
B[1][1] = 8

Matrix A:
1       2       3
9       8

Matrix B:
4       5       6
7       8

Result (A + B):
5       7       9
16      16
```

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!

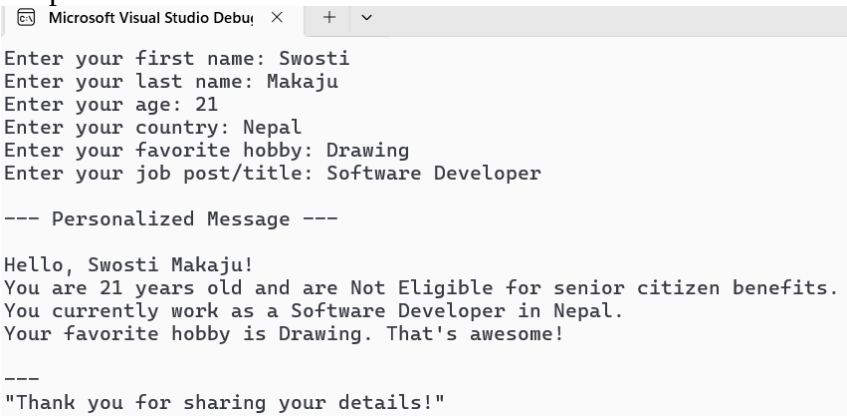
"Thank you for sharing your details!"



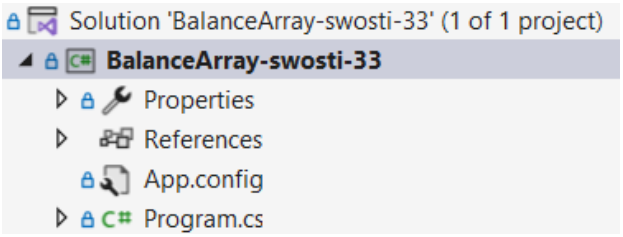
Source Code :

```
using System;
class UserProfile
{
   static void Main()
   {
      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 = int.Parse(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 jobPost = Console.ReadLine();
      string eligibility = age >= 60 ? "Eligible" : "Not Eligible";
      Console.WriteLine("\n--- Personalized Message ---\n");
      Console.WriteLine($"Hello, {firstName} {lastName}!");
      Console.WriteLine($"You are {age} years old and are {eligibility} for senior citizen benefits.");
      Console.WriteLine($"You currently work as a {jobPost} in {country}.");
      Console.WriteLine($"Your favorite hobby is {hobby}. That's awesome!");
      Console.WriteLine("\n---\n\"Thank you for sharing your details!\"");
} }
```

Output :



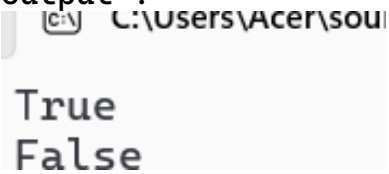
10. Write a method name is BalanceArray(int[] a) that returns true if the array is balanced. An array is called balanced if the number of even elements is equal to the number of odd elements.



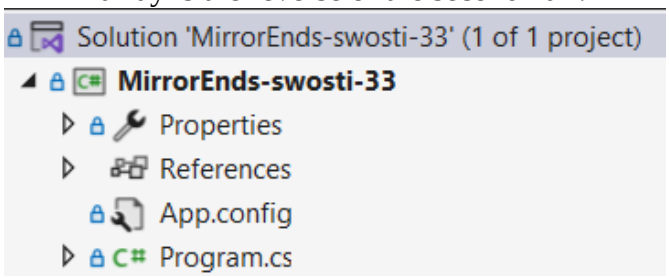
Source Code :

```
  internal class Program
  {
      static void Main(string[] args)
      {
          int[] arrone = { 2, 3, 4, 5 };
          int[] arrtwo = { 1, 3, 5, 7 };

          Console.WriteLine(isBalanceArray(arrone));
          Console.WriteLine(isBalanceArray(arrtwo));
          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 :



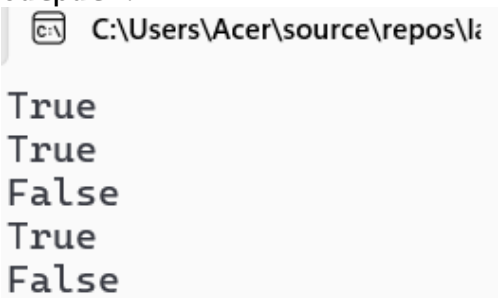
11. Write a method has MirrorEnds(int[] a) that returns true if the first half of the array is the reverse of the second half.



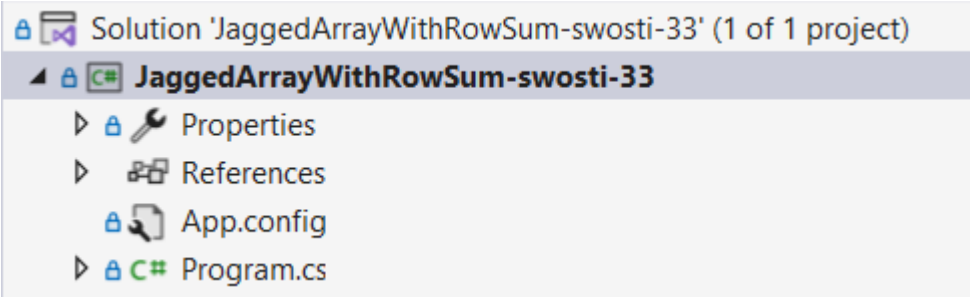
Source Code :

```
internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(hasMirrorEnds(new int[] { 1, 2, 3, 2, 1 }));
      Console.WriteLine(hasMirrorEnds(new int[] { 7, 8, 9, 8, 7 }));
        Console.WriteLine(hasMirrorEnds(new int[] { 1, 2, 3, 4, 5 }));
        Console.WriteLine(hasMirrorEnds(new int[] { 1, 2, 3, 1 }));
        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 :



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



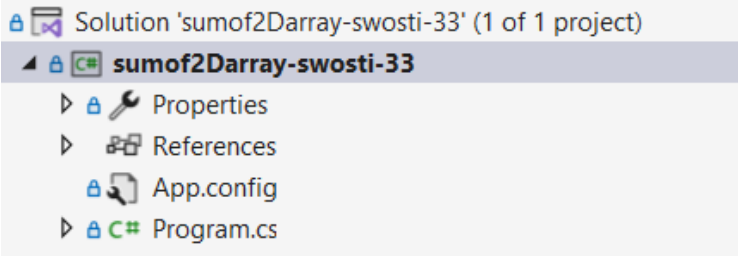
Source Code :

```
using System;
class JaggedArrayWithRowSum
{
    static void Main()
    {
        int[][] jaggedArray = new int[][]
        {
            new int[] { 1, 2, 3 },
            new int[] { 4, 5 },
            new int[] { 6, 7, 8, 9 }
        };
        Console.WriteLine("Jagged Array Elements and Row Sums:\n");
        for (int i = 0; i < jaggedArray.Length; i++)
        {
            int rowSum = 0;
            Console.Write("Row " + (i + 1) + ": ");
            for (int j = 0; j < jaggedArray[i].Length; j++)
            {
                Console.Write(jaggedArray[i][j] + " ");
                rowSum += jaggedArray[i][j];
            }
            Console.WriteLine("=> Sum = " + rowSum);
        }
    }
}
```

Output :

```
Jagged Array Elements and Row Sums:

Row 1: 1 2 3 => Sum = 6
Row 2: 4 5 => Sum = 9
Row 3: 6 7 8 9 => Sum = 30
```

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



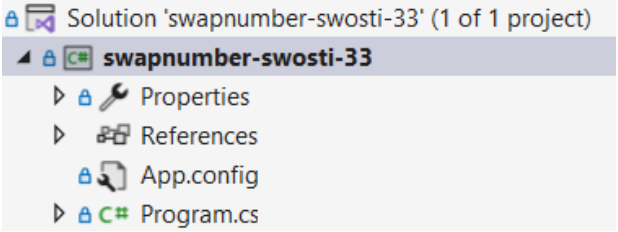
Source Code :

```
using System;
class RowSum2DArray
{
   static void Main()
   {
      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[,] matrix = new int[rows, cols];
      Console.WriteLine("\nEnter matrix elements:");
      for (int i = 0; i < rows; i++)
      {
         for (int j = 0; j < cols; j++)
         {
            Console.Write($"Element [{i},{j}]: ");
            matrix[i, j] = int.Parse(Console.ReadLine());
         }
      }
      Console.WriteLine("\nMatrix and Row Sums:");
      for (int i = 0; i < rows; i++)
      {
         int rowSum = 0;
         Console.Write("Row " + (i + 1) + ": ");
         for (int j = 0; j < cols; j++)
         {
            Console.Write(matrix[i, j] + " ");
            rowSum += matrix[i, j];
         }
         Console.WriteLine("=> Sum = " + rowSum);
      }
   }
}
```

Output :

```
Enter number of rows: 3
Enter number of columns: 2

Enter matrix elements:
Element [0,0]: 4
Element [0,1]: 6
Element [1,0]: 2
Element [1,1]: 1
Element [2,0]: 7
Element [2,1]: 5

Matrix and Row Sums:
Row 1: 4 6 => Sum = 10
Row 2: 2 1 => Sum = 3
Row 3: 7 5 => Sum = 12
```

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



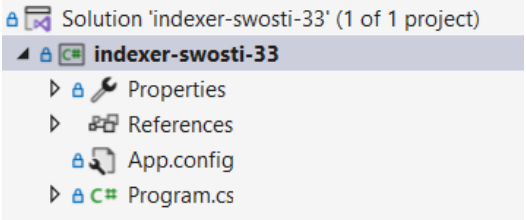
Source Code :

```
using System;
class SwapUsingRef
{
    static void Main()
    {
        int a, b;
        Console.Write("Enter first number (a): ");
        a = int.Parse(Console.ReadLine());
        Console.Write("Enter second number (b): ");
        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 :

```
Enter first number (a): 11
Enter second number (b): 22

Before swap: a = 11, b = 22
After swap: a = 22, b = 11
```

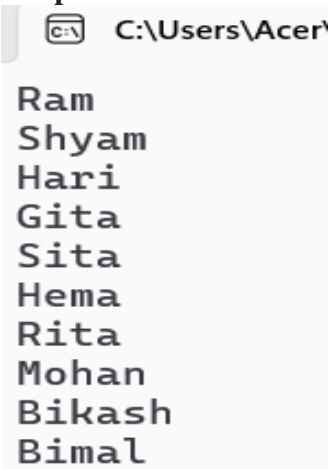
15. Write a program to demonstrate the concept of Indexer.



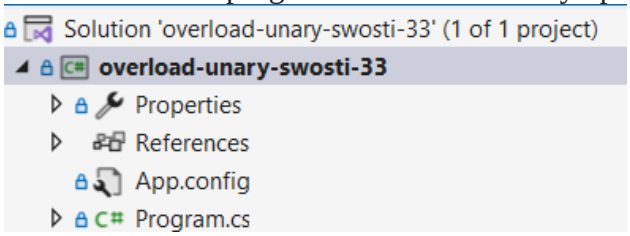
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace indexer_swosti_33
{
    class IndexerClass
    {
        private string[] name = new string[10];
        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 :



16. Write C# program to overload Unary operator.



Source Code :

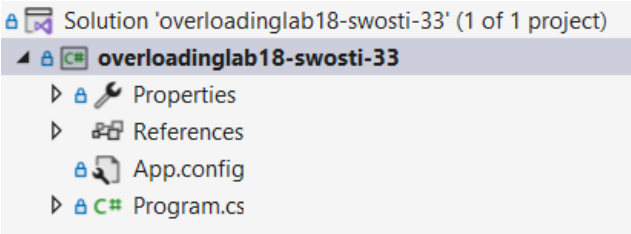
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace overload_unary_swosti_33
{
    class Program
    {
        static void Main(string[] args)
        {
            Calculation num = new Calculation(5);
            // Unary + operator
            Calculation positiveNum = +num;
            Console.WriteLine("Unary + : " + positiveNum.Display());
      Calculation negatedNum = -num;
            Console.WriteLine("Unary - : " + negatedNum.Display());
            Console.WriteLine("Unary ! : " + (!num));
            num++;
            Console.WriteLine("Unary ++ : " + num.Display
            num--;
            Console.WriteLine("Unary -- : " + num.Display());
      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;
        return a;
    }
    public int Display()
    {
        return x;
    }
}
```

Output :

```
C:\Users\Acer\source\repos\c

Unary + : 5
Unary - : -5
Unary ! : False
Unary ++ : 6
Unary -- : 5
```

17. Perform binary operator overloading to add two rectangle and check they are equal in C#.



Source Code :

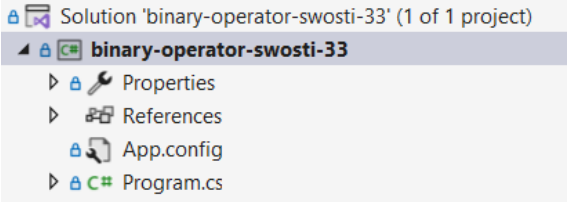
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace overloadinglab18_swosti_33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Rectangle rect1 = new Rectangle(20, 5);
            Rectangle rect2 = new Rectangle(20, 5);
            Console.WriteLine("Rectangle 1:");
            rect1.Display();
            Console.WriteLine("Rectangle 2:");
            rect2.Display();
            Rectangle result = rect1 + rect2;
      Console.WriteLine("Result of Addition:");
            result.Display();
            Console.WriteLine("Is Rectangle 1 equal to Rectangle 2?");
            Console.WriteLine(rect1 == rect2 ? "Yes" : "No");
            Console.ReadLine();
        }
    }
    class Rectangle
    {
        public int Width, Height;

        public Rectangle(int width, int height)
        {
            Width = width;
            Height = height;
        }
        public static Rectangle operator +(Rectangle r1, Rectangle r2)
        {
            return new Rectangle(r1.Width + r2.Width, r1.Height + r2.Height);
        }
        public static bool operator ==(Rectangle r1, Rectangle r2)
        {
            return r1.Width == r2.Width && r1.Height == r2.Height;
        }
        public static bool operator !=(Rectangle r1, Rectangle r2)
        {
            return !(r1 == r2);
        }
        public override bool Equals(object obj)
        {
            if (obj is Rectangle)
            {
```

```
            Rectangle r = (Rectangle)obj;
            return this == r;
        }
        return false;
    }

    public override int GetHashCode()
    {
        return (Width, Height).GetHashCode();
    }

    public void Display()
    {
        Console.WriteLine($"Width: {Width}, Height: {Height}");
    }
}  }
```

Output :

```
C:\Users\Acer\source\repos\c

Rectangle 1:
Width: 20, Height: 5
Rectangle 2:
Width: 20, Height: 5
Result of Addition:
Width: 40, Height: 10
Is Rectangle 1 equal to Rectangle 2?
Yes
```

18. Write a program to overload Binary operator.



Source Code :

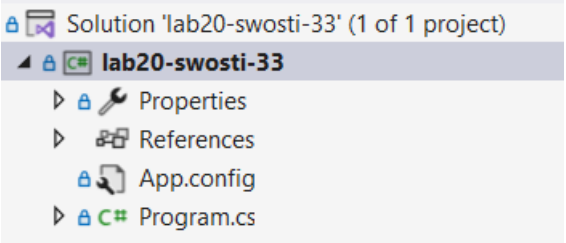
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace binary_operator_swosti_33
{
    class Program
    {
        static void Main(string[] args)
        {
            Calculation a = new Calculation(10);
            Calculation b = new Calculation(5);
            Calculation sum = a + b;
            Console.WriteLine("a + b = " + sum.Display());
            Calculation diff = a - b;
            Console.WriteLine("a - b = " + diff.Display());
            Calculation product = a * b;
            Console.WriteLine("a * b = " + product.Display());
            Calculation quotient = a / b;
            Console.WriteLine("a / b = " + quotient.Display());
            Calculation reminder = a % 2;
            Console.WriteLine("a % 2 = " + reminder.Display());
            Console.ReadLine();
        }
    }
    public class Calculation
    {
        int x;
        public Calculation(int x)
        {
            this.x = x;
        }
        public static Calculation operator +(Calculation a, Calculation b)
        {
            return new Calculation(a.x + b.x);
        }
        public static Calculation operator -(Calculation a, Calculation b)
        {
            return new Calculation(a.x - b.x);
        }
        public static Calculation operator *(Calculation a, Calculation b)
        {
            return new Calculation(a.x * b.x);
        }
        public static Calculation operator /(Calculation a, Calculation b)
        {
            return new Calculation(a.x / b.x);
        }
        public static Calculation operator %(Calculation a, int scalar)
        {
            return new Calculation(a.x % scalar);
        }
```

```
        public int Display()
        {
            return x;
        }
    }
}
```

Output :

```
C:\Users\Acer\source\repos\b

a + b = 15
a - b = 5
a * b = 50
a / b = 2
a % 2 = 0
```

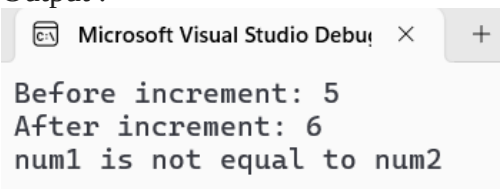
19. Write a C# program to overload unary (++) and relation operator (==) operator.



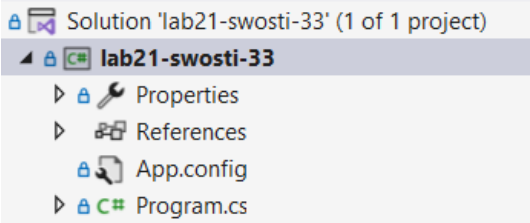
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
namespace lab20_swosti_33
{
    class Program
    {
        static void Main(string[] args)
        {
            MyNumber num1 = new MyNumber(5);
            MyNumber num2 = new MyNumber(5);
            Console.WriteLine("Before increment: " + num1.Value);
            ++num1;
            Console.WriteLine("After increment: " + num1.Value);
            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;
        public MyNumber(int value)
        {    Value = value;
        }
        public static MyNumber operator ++(MyNumber num)
        {   num.Value++;
            return num;
        }
        public static bool operator ==(MyNumber num1, MyNumber num2)
        {   return num1.Value == num2.Value;
        }
        public static bool operator !=(MyNumber num1, MyNumber num2)
        {   return !(num1 == num2);
        }
        public override bool Equals(object obj)
        {   MyNumber num = (MyNumber)obj;
            return Value == num.Value;
        }
        public override int GetHashCode()
        {   return Value.GetHashCode();
        }  }  }
```

Output :



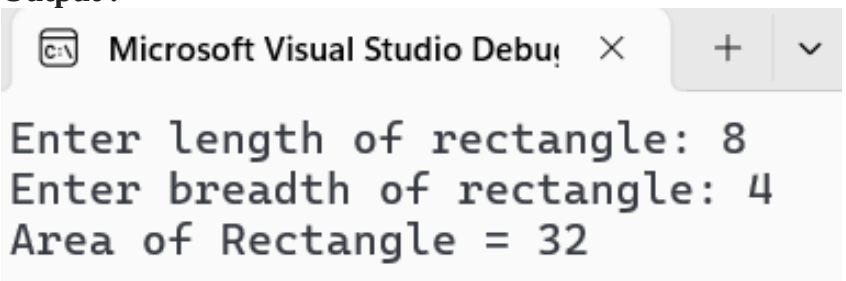
20. Write a program to calculate area of rectangle using simple inheritance.



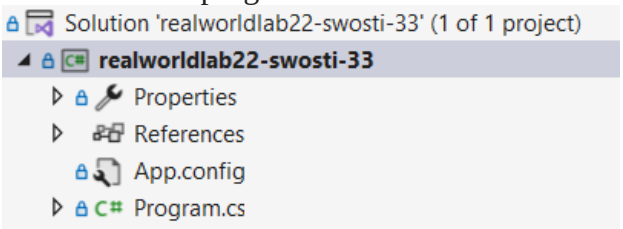
Source Code :

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

namespace lab21_swosti_33
{
    class Rectangle
    {
        protected double length;
        protected double breadth;

        public void SetDimensions(double l, double b)
        {
            length = l;
            breadth = b;
        }
    }
    class AreaCalculator : Rectangle
    {
        public double CalculateArea()
        {
            return length * breadth;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            AreaCalculator rect = new AreaCalculator();
            Console.Write("Enter length of rectangle: ");
            double l = Convert.ToDouble(Console.ReadLine());
            Console.Write("Enter breadth of rectangle: ");
            double b = Convert.ToDouble(Console.ReadLine());
            rect.SetDimensions(l, b);
            double area = rect.CalculateArea();
            Console.WriteLine("Area of Rectangle = " + area);
            Console.ReadLine();
        }
    }
}
```

Output :



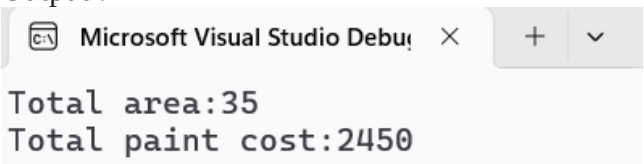
21. Write a program to calculate area of rectangle using multiple inheritance.



Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace realworldlab22_swosti_33
{
    class Program
    {
        static void Main(string[] args)
        {
            Rectangle Rect = new Rectangle();
            int area;
            Rect.setWidth(5);
            Rect.setHeight(7);
            area = Rect.getArea();
            Console.WriteLine("Total area:" + Rect.getArea());
            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;
       }}   public interface PaintCost
    {
        int getCost(int area);
    }
    class Rectangle : Shape, PaintCost
    {
        public int getArea()
        {
            return (width * height);
        }
        public int getCost(int area)
        {
            return area * 70;
}  }  }
```

Output :



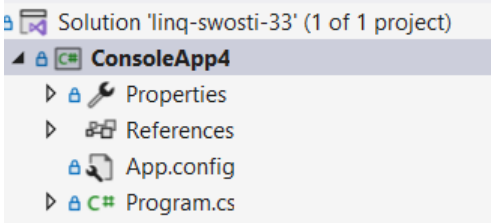

```
Total area:35
Total paint cost:2450
```

22. You have given table with following data(tblstudent).

Id	Name	Gender	Country	Salary	RegDate	Dob
1	Sunil Chaudhary	Male	Nepal	40000	8/27/2024	8/27/1999
2	Suraj Sapkota	Male	Nepal	50000	8/27/2024	1/3/1998
3	Sita Thapa	Female	China	80000	8/27/2024	2/24/1997
4	Mohan Ghimire	Male	India	30000	8/26/2024	5/26/1995
5	Dina Shrestha	Female	Nepal	20000	8/25/2024	8/27/1994
6	Ritesh Kafle	Male	India	50000	8/25/2024	8/27/1997
7	Rima Nepal	Female	India	90000	8/26/2024	8/27/2000

Use LINQ to perform following operation:

1. Fetch all records from table.
2. Fetch all records from table with Name asc order.
3. Fetch all records from table with Name desc order.
4. Fetch top 3 records from table.
5. Find average salary from given table.
6. Fetch all employee whose country is Nepal and China.
7. Fetch all records of employee that are registered in August month.
8. Fetch all records of employee that are registered in between 8/26/2024 to 8/28/2024.
9. Fetch all records of employee by ordering in Name in asc order then by salary.
10. Fetch all records whose country is Nepal and salary is above 60000.
11. Get sum of salaries of all the employees from above table.
12. Get max salary from above employee table.
13. Get min salary from above employee table.
14. Get Id, Name, Salary from above table.
15. Get Id, Name, 30% of Salary from above table.
16. Get all records from above table where Name starts with “S”.
17. Get the number of Female employee from above table.
18. Get number of Male and Female employees from Table along with gender as one column.
19. Get sum of salaries for the employees as per Gender from Table.



Source Code :

Program.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApp4
{
    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 = 40000, RegDate = new DateTime(2024, 8, 27), Dob = new DateTime(1999, 8,
              27) });
            lst.Add(new tblEmployee() { Id = 2, Name = "Suraj Sapkota", Gender = "Male", Country =
              "Nepal", Salary = 50000, RegDate = new DateTime(2024, 8, 27), Dob = new DateTime(1988, 1,
              3) });
```

```
lst.Add(new tblEmployee() { Id = 3, Name = "Sita Thapa", Gender = "Female", Country = "China",
  Salary = 80000, RegDate = new DateTime(2024, 8, 27), Dob = new DateTime(1997, 2, 24) });
lst.Add(new tblEmployee() { Id = 4, Name = "Mohan Ghimire ", Gender = "Male", Country =
  "India", Salary = 30000, RegDate = new DateTime(2024, 8, 26), Dob = new DateTime(1995, 5,
  26) });
lst.Add(new tblEmployee() { Id = 5, Name = "Dina Shrestha", Gender = "Female", Country =
  "Nepal", Salary = 20000, RegDate = new DateTime(2024, 8, 25), Dob = new DateTime(1994, 8,
  27) });
lst.Add(new tblEmployee() { Id = 6, Name = "Ritesh Kafle", Gender = "Male", Country = "India",
  Salary = 50000, RegDate = new DateTime(2024, 8, 25), Dob = new DateTime(1997, 8, 27) });
lst.Add(new tblEmployee() { Id = 7, Name = "Rima Nepal", Gender = "Female", Country = "India",
  Salary = 90000, RegDate = new DateTime(2026, 8, 24), Dob = new DateTime(2000, 8, 27) });
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, emp.Dob);
}
Console.WriteLine("2.Fetch all records from table with Name asc order.");
List<tblEmployee> ascnameList = lst.OrderBy(a => a.Name).ToList();
foreach (tblEmployee emp in ascnameList)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("3.Fetch all records from table with Name desc order.");
List<tblEmployee> descnameList = lst.OrderByDescending(a => a.Name).ToList();
foreach (tblEmployee emp in descnameList)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("4.Fetch top 3 records from table");
List<tblEmployee> top3list = lst.OrderBy(a => a.Name).Take(3).ToList();
foreach (tblEmployee emp in top3list)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("5. Find average salary from given table");
var avgsalary = lst.Average(a => a.Salary);
Console.WriteLine(avgsalary);
Console.WriteLine("6.Fetch all employee whose country is Nepal or China.");
List<tblEmployee> empnepalchinalist = lst.Where(a => a.Country == "Nepal" || a.Country ==
  "China").ToList();
foreach (tblEmployee emp in empnepalchinalist)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("7.Fetch all records of employee that are registered in August month.");
List<tblEmployee> empaugList = lst.Where(a => a.RegDate.Month == 8).ToList();
foreach (tblEmployee emp in empaugList)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("8.Fetch all records of employee that are registered in between 8/26/2024 to
  8/28/2024.");
DateTime fromdate = new DateTime(2024, 8, 26);
```

```
DateTime todate = new DateTime(2024, 8, 28);
List<tblEmployee> empdatebetween = lst.Where(a => a.RegDate >= fromdate && a.RegDate <=
  todate).ToList();
foreach (tblEmployee emp in empdatebetween)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("9.Fetch all records of employee by ordering in Name in asc order then by
  salary.");
List<tblEmployee> emp_name_salary_asc = lst.OrderBy(a => a.Name).ThenBy(a =>
  a.Salary).ToList();
foreach (tblEmployee emp in emp_name_salary_asc)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("10.\tFetch all records whose country is Nepal and salary is above 50000.");
List<tblEmployee> listabovesalary = lst.Where(a => a.Salary >= 50000 && a.Country ==
  "Nepal").ToList();
foreach (tblEmployee emp in listabovesalary)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("11.\tGet sum of salaries of all the employees from above table.");
var employee_sum_salary = lst.Sum(x => x.Salary);
Console.WriteLine("Sum of Salary: " + employee_sum_salary);
Console.WriteLine("12.\tGet max salary from above employee table.");
var maxsalary = lst.Max(x => x.Salary);
Console.WriteLine("Max Salary: " + maxsalary);
Console.WriteLine("13.\tGet min salary from above employee table.");
var minsalary = lst.Min(x => x.Salary);
Console.WriteLine("Min Salary: " + minsalary);
Console.WriteLine("14.\tGet Id, Name, Salary from above table.");
var listrowfilter = lst.Select(x => new { Id = x.Id, Name = x.Name, Salary = x.Salary }).ToList();
foreach (var emp in listrowfilter)
{
   Console.WriteLine("{0}  {1}   {2}", emp.Id, emp.Name, emp.Salary);
}
Console.WriteLine("15.\tGet Id, Name, 30% of Salary from above table.");
var listsalaryfilter = lst.Select(x => new { Id = x.Id, Name = 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("16.\tGet all records from above table where Name starts with “S”.");
List<tblEmployee> list_starts_with_s = lst.Where(a =>
  a.Name.StartsWith("S")).ToList();//EndsWith, Contains
foreach (tblEmployee emp in list_starts_with_s)
{
   Console.WriteLine("{0}  {1}   {2}    {3}    {4}    {5}     {6}", emp.Id, emp.Name, emp.Gender,
  emp.Country, emp.Salary, emp.RegDate, emp.Dob);
}
Console.WriteLine("17.\tGet the number of Female employee from above table.");
var totaL_No_female = lst.Where(a => a.Gender == "Female").ToList().Count;
Console.WriteLine("Total No of Female:" + totaL_No_female);
Console.WriteLine("18.\tGet number of Male and Female employees from Table along with gender
  as one column.");
```

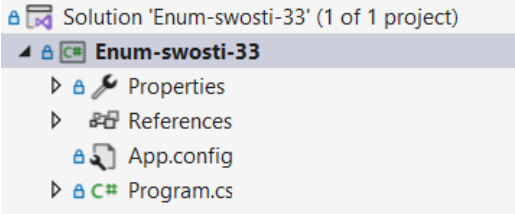
```
        var groupgender = lst.GroupBy(x => x.Gender).Select(y => new { Gender = y.Key, count =
          y.Count() });
        foreach (var emp in groupgender)
        {
            Console.WriteLine(emp.Gender + ":" + emp.count);
        }
        Console.WriteLine("19.\tGet sum of salaries for the employees as per Gender from Table.");
        var groupgender_Salary = lst.GroupBy(x => x.Gender).Select(y => new { Gender = y.Key,
          sumofsalary = y.Sum(z => z.Salary) });
        foreach (var emp in groupgender_Salary)
        {
            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 decimal Salary { get; set; }
    public DateTime RegDate { get; set; }
    public DateTime Dob { get; set; }
}
```

Output :

```
C:\Users\Acer\source\repos\C

1. Fetch all records
1  Sunil Chaudhary   Male    Nepal    40000    8/27/2024 12:00:00 AM     8/27/1999 12:00:00 AM
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
3  Sita Thapa   Female    China    80000    8/27/2024 12:00:00 AM     2/24/1997 12:00:00 AM
4  Mohan Ghimire    Male    India    30000    8/26/2024 12:00:00 AM     5/26/1995 12:00:00 AM
5  Dina Shrestha   Female    Nepal    20000    8/25/2024 12:00:00 AM     8/27/1994 12:00:00 AM
6  Ritesh Kafle   Male    India    50000    8/25/2024 12:00:00 AM     8/27/1997 12:00:00 AM
7  Rima Nepal   Female    India    90000    8/24/2026 12:00:00 AM     8/27/2000 12:00:00 AM
2.Fetch all records from table with Name asc order.
5  Dina Shrestha   Female    Nepal    20000    8/25/2024 12:00:00 AM     8/27/1994 12:00:00 AM
4  Mohan Ghimire    Male    India    30000    8/26/2024 12:00:00 AM     5/26/1995 12:00:00 AM
7  Rima Nepal   Female    India    90000    8/24/2026 12:00:00 AM     8/27/2000 12:00:00 AM
6  Ritesh Kafle   Male    India    50000    8/25/2024 12:00:00 AM     8/27/1997 12:00:00 AM
3  Sita Thapa   Female    China    80000    8/27/2024 12:00:00 AM     2/24/1997 12:00:00 AM
1  Sunil Chaudhary   Male    Nepal    40000    8/27/2024 12:00:00 AM     8/27/1999 12:00:00 AM
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
3.Fetch all records from table with Name desc order.
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
1  Sunil Chaudhary   Male    Nepal    40000    8/27/2024 12:00:00 AM     8/27/1999 12:00:00 AM
3  Sita Thapa   Female    China    80000    8/27/2024 12:00:00 AM     2/24/1997 12:00:00 AM
6  Ritesh Kafle   Male    India    50000    8/25/2024 12:00:00 AM     8/27/1997 12:00:00 AM
7  Rima Nepal   Female    India    90000    8/24/2026 12:00:00 AM     8/27/2000 12:00:00 AM
4  Mohan Ghimire    Male    India    30000    8/26/2024 12:00:00 AM     5/26/1995 12:00:00 AM
5  Dina Shrestha   Female    Nepal    20000    8/25/2024 12:00:00 AM     8/27/1994 12:00:00 AM
4.Fetch top 3 records from table
5  Dina Shrestha   Female    Nepal    20000    8/25/2024 12:00:00 AM     8/27/1994 12:00:00 AM
4  Mohan Ghimire    Male    India    30000    8/26/2024 12:00:00 AM     5/26/1995 12:00:00 AM
7  Rima Nepal   Female    India    90000    8/24/2026 12:00:00 AM     8/27/2000 12:00:00 AM
5. Find average salary from given table
51428.571428571428571428571429
```

```
6.Fetch all employee whose country is Nepal or China.
1  Sunil Chaudhary   Male    Nepal    40000    8/27/2024 12:00:00 AM     8/27/1999 12:00:00 AM
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
3  Sita Thapa   Female    China    80000    8/27/2024 12:00:00 AM     2/24/1997 12:00:00 AM
5  Dina Shrestha   Female    Nepal    20000    8/25/2024 12:00:00 AM     8/27/1994 12:00:00 AM
7.Fetch all records of employee that are registered in August month.
1  Sunil Chaudhary   Male    Nepal    40000    8/27/2024 12:00:00 AM     8/27/1999 12:00:00 AM
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
3  Sita Thapa   Female    China    80000    8/27/2024 12:00:00 AM     2/24/1997 12:00:00 AM
4  Mohan Ghimire    Male    India    30000    8/26/2024 12:00:00 AM     5/26/1995 12:00:00 AM
5  Dina Shrestha   Female    Nepal    20000    8/25/2024 12:00:00 AM     8/27/1994 12:00:00 AM
6  Ritesh Kafle   Male    India    50000    8/25/2024 12:00:00 AM     8/27/1997 12:00:00 AM
7  Rima Nepal   Female    India    90000    8/24/2026 12:00:00 AM     8/27/2000 12:00:00 AM
8.Fetch all records of employee that are registered in between 8/26/2024 to 8/28/2024.
1  Sunil Chaudhary   Male    Nepal    40000    8/27/2024 12:00:00 AM     8/27/1999 12:00:00 AM
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
3  Sita Thapa   Female    China    80000    8/27/2024 12:00:00 AM     2/24/1997 12:00:00 AM
4  Mohan Ghimire    Male    India    30000    8/26/2024 12:00:00 AM     5/26/1995 12:00:00 AM
9.Fetch all records of employee by ordering in Name in asc order then by salary.
5  Dina Shrestha   Female    Nepal    20000    8/25/2024 12:00:00 AM     8/27/1994 12:00:00 AM
4  Mohan Ghimire    Male    India    30000    8/26/2024 12:00:00 AM     5/26/1995 12:00:00 AM
7  Rima Nepal   Female    India    90000    8/24/2026 12:00:00 AM     8/27/2000 12:00:00 AM
6  Ritesh Kafle   Male    India    50000    8/25/2024 12:00:00 AM     8/27/1997 12:00:00 AM
3  Sita Thapa   Female    China    80000    8/27/2024 12:00:00 AM     2/24/1997 12:00:00 AM
1  Sunil Chaudhary   Male    Nepal    40000    8/27/2024 12:00:00 AM     8/27/1999 12:00:00 AM
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
10.     Fetch all records whose country is Nepal and salary is above 50000.
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
11.     Get sum of salaries of all the employees from above table.
Sum of Salary: 360000
12.     Get max salary from above employee table.
Max Salary: 90000
13.     Get min salary from above employee table.
Min Salary: 20000
14.     Get Id, Name, Salary from above table.
1  Sunil Chaudhary   40000
2  Suraj Sapkota   50000
3  Sita Thapa   80000
4  Mohan Ghimire    30000
5  Dina Shrestha   20000
6  Ritesh Kafle   50000
7  Rima Nepal   90000
15.     Get Id, Name, 30% of Salary from above table.
1  Sunil Chaudhary   12000.00
2  Suraj Sapkota   15000.00
3  Sita Thapa   24000.00
4  Mohan Ghimire    9000.00
5  Dina Shrestha   6000.00
6  Ritesh Kafle   15000.00
7  Rima Nepal   27000.00
16.     Get all records from above table where Name starts with "S".
1  Sunil Chaudhary   Male    Nepal    40000    8/27/2024 12:00:00 AM     8/27/1999 12:00:00 AM
2  Suraj Sapkota   Male    Nepal    50000    8/27/2024 12:00:00 AM     1/3/1988 12:00:00 AM
3  Sita Thapa   Female    China    80000    8/27/2024 12:00:00 AM     2/24/1997 12:00:00 AM
17.     Get the number of Female employee from above table.
Total No of Female:3
18.     Get number of Male and Female employees from Table along with gender as one column.
Male:4
Female:3
19.     Get sum of salaries for the employees as per Gender from Table.
Male:170000
Female:190000
```

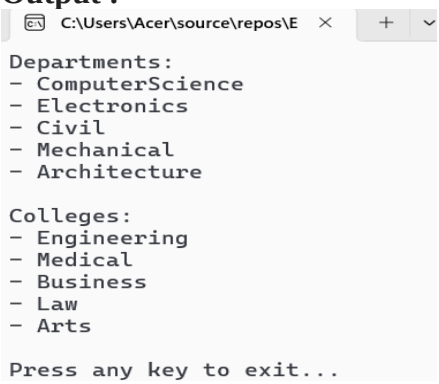
23. Write a C# program which stores values in two enumerations. Department and college it uses two functions to display the data content in department and college enumerations.



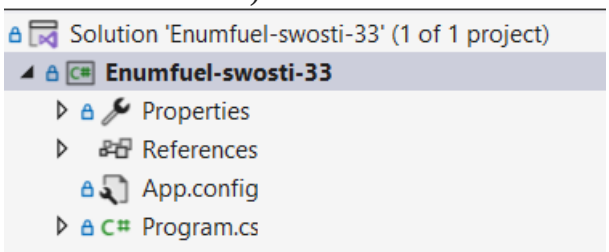
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Enum_swosti_33
{
    enum Department
    {   ComputerScience,
        Electronics,
        Civil,
        Mechanical,
        Architecture
}  enum College
    {   Engineering,
        Medical,
        Business,
        Law,
        Arts
}     class Program
    {    static void DisplayDepartments()
        {   Console.WriteLine("Departments:");
            foreach (string dept in Enum.GetNames(typeof(Department)))
            {
                Console.WriteLine($"- {dept}");
            }
        }  static void DisplayColleges()
        {   Console.WriteLine("Colleges:");
            foreach (string college in Enum.GetNames(typeof(College)))
            {
                Console.WriteLine($"- {college}");
            }
        }  static void Main(string[] args)
        {   DisplayDepartments();
            Console.WriteLine();
            DisplayColleges();
            Console.WriteLine("\nPress any key to exit...");
            Console.ReadKey();
       } } }
```

Output :



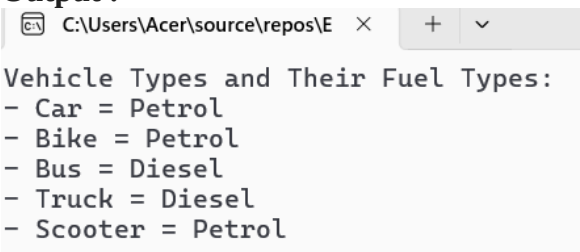
24. Write a C# program that stores values in an enumeration. VehicleType and displays the fuel type for each vehicle (e.g. Car=Petrol, Bike= Petrol, Bus=Diesel).



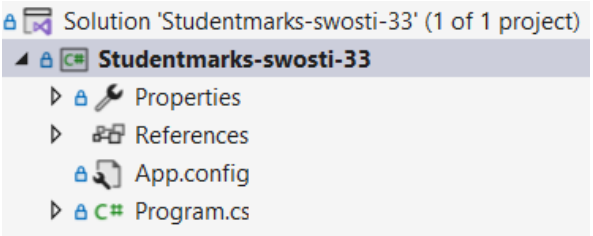
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Enumfuel_swosti_33
{    enum VehicleType  {
        Car,
        Bike,
        Bus,
        Truck,
        Scooter
    } class Program
    {
        static string GetFuelType(VehicleType vehicle)
        {
            switch (vehicle)
            {
                case VehicleType.Car:
                case VehicleType.Bike:
                case VehicleType.Scooter:
                    return "Petrol";
                case VehicleType.Bus:
                case VehicleType.Truck:
                    return "Diesel";
                default:
                    return "Unknown";
            }
        } static void Main(string[] args)
        {
            Console.WriteLine("Vehicle Types and Their Fuel Types:");
            foreach (VehicleType vehicle in
      Enum.GetValues(typeof(VehicleType)))
            {
                string fuelType = GetFuelType(vehicle);
                Console.WriteLine($"- {vehicle} = {fuelType}");
            }
            Console.WriteLine("\nPress any key to exit...");
            Console.ReadKey();
        } } }
```

Output :



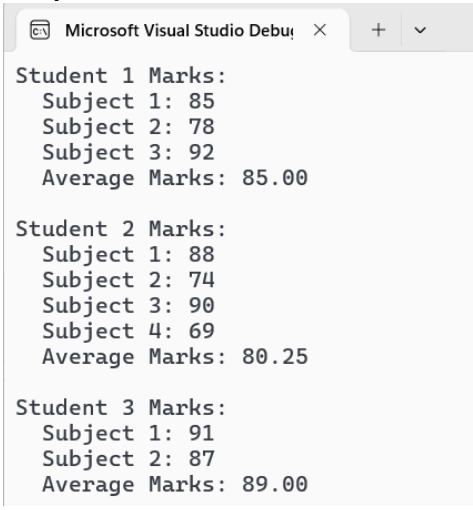
25. Write a C# program to create multidimensional array to store the marks of three student in different subjects. First student has marks of 3 subjects, second student has marks of 4 subjects and Third student has marks of 2 subjects, Display the subject marks and average marks for each student.



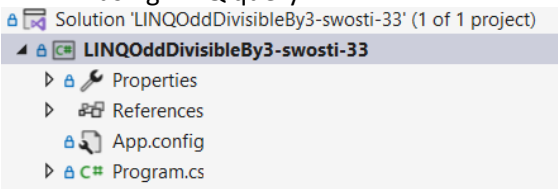
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Studentmarks_swosti_33
{
    class Program
    {
        static void Main(string[] args)
        {
            int[][] studentMarks = new int[3][];
            studentMarks[0] = new int[] { 85, 78, 92 };
            studentMarks[1] = new int[] { 88, 74, 90, 69 };
            studentMarks[2] = new int[] { 91, 87 };
            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");
            }    }  }   }
```

Output :



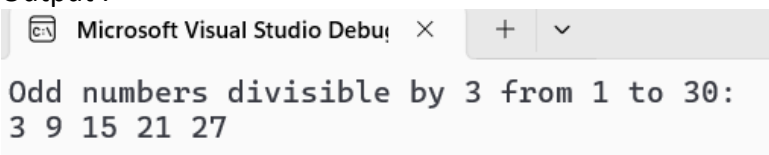
26. Write a C# program to select odd and divisible by 3 number from list of numbers (1-30) using LINQ query.



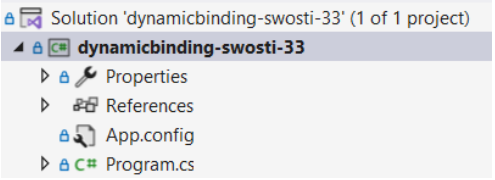
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace LINQOddDivisibleBy3_swosti_33
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> numbers = Enumerable.Range(1, 30).ToList();
            var result = from num in numbers
                         where num % 2 != 0 && num % 3 == 0
                         select num;
            Console.WriteLine("Odd numbers divisible by 3 from 1 to 30:");
            foreach (var number in result)
            {
                Console.Write(number + " ");
            }
            Console.WriteLine();
        }
 }

}
```

Output :



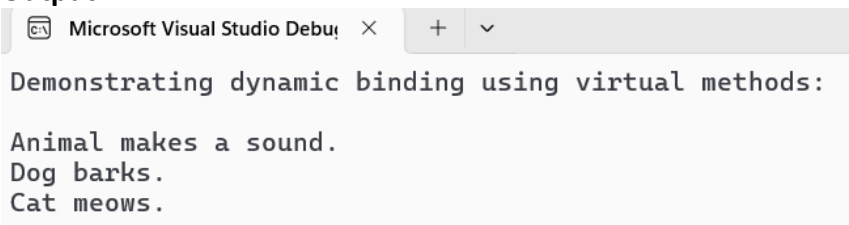
27. Write a C# program to achieve dynamic binding using virtual method in C#.



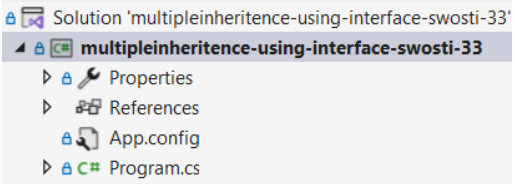
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace dynamicbinding_swosti_33
{
    class Animal
    {
        public virtual void Speak()
        {
            Console.WriteLine("Animal makes a sound.");
        }
    }
    class Dog : Animal
    {
        public override void Speak()
        {
            Console.WriteLine("Dog barks.");
        }
    }
    class Cat : Animal
    {
        public override void Speak()
        {
            Console.WriteLine("Cat meows.");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Animal[] animals = new Animal[3];
            animals[0] = new Animal();
            animals[1] = new Dog();
            animals[2] = new Cat();
            Console.WriteLine("Demonstrating dynamic binding using virtual
      methods:\n");
            foreach (Animal animal in animals)
            {
                animal.Speak();
            }     }        }            }
```

Output :



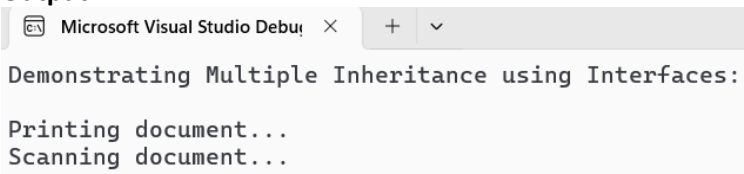
28. Write a program to achieve multiple inheritance using interface.



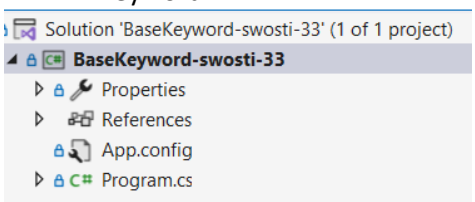
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace multipleinheritence_using_interface_swosti_33
{
    interface IPrintable
    {
        void Print();
    }
    interface IScannable
    {
        void Scan();
    }
    class MultiFunctionPrinter : IPrintable, IScannable
    {
        public void Print()
        {
            Console.WriteLine("Printing document...");
        }
        public void Scan()
        {
            Console.WriteLine("Scanning document...");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MultiFunctionPrinter mfp = new MultiFunctionPrinter();

            Console.WriteLine("Demonstrating Multiple Inheritance using
      Interfaces:\n");
            mfp.Print();
            mfp.Scan();

        }  }  }
```

Output :



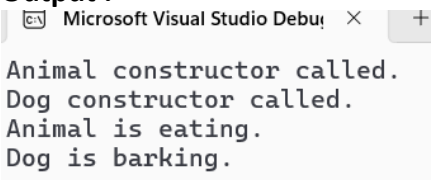
29. Write a C# program to call member function and constructor of parent class using base keyword.



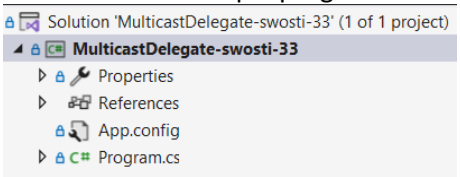
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace BaseKeyword_swosti_33
{
    class Animal
    {
        public Animal()
        {
            Console.WriteLine("Animal constructor called.");
        }
        public void Eat()
        {
            Console.WriteLine("Animal is eating.");
        }
    }
    class Dog : Animal
    {
        public Dog() : base()          {
            Console.WriteLine("Dog constructor called.");
        }
        public void ShowBehavior()
        {
            base.Eat();
            Console.WriteLine("Dog is barking.");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Dog myDog = new Dog();
            myDog.ShowBehavior();
        }
}
  }
```

Output :



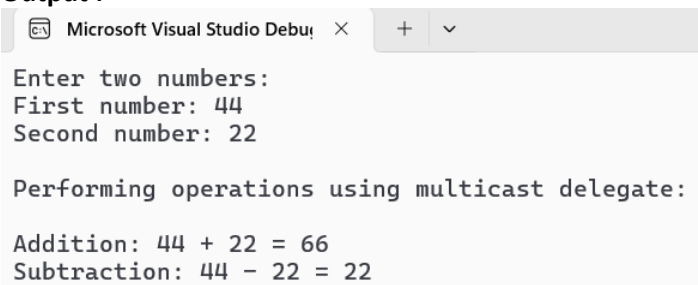
30. Write a simple program to add and subtract two digit using multicast delgates.



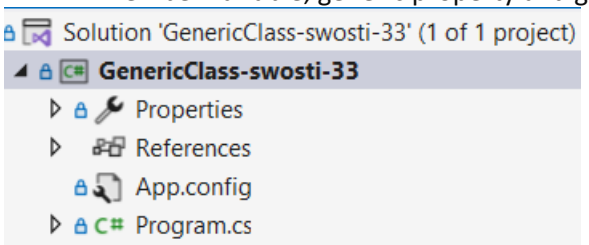
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace MulticastDelegate_swosti_33
{
    public delegate void MathOperation(int x, int y);
    class Program
    {
        public static void Add(int a, int b)
        {
            Console.WriteLine("Addition: {0} + {1} = {2}", a, b, a + b);
        }
        public static void Subtract(int a, int b)
        {
            Console.WriteLine("Subtraction: {0} - {1} = {2}", a, b, a - b);
        }
        static void Main(string[] args)
        {
            MathOperation operation = Add;
            operation += Subtract;
            Console.WriteLine("Enter two numbers:");
            Console.Write("First number: ");
            int num1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Second number: ");
            int num2 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("\nPerforming operations using multicast
      delegate:\n");
            operation(num1, num2);          }
    }
}
```

Output :



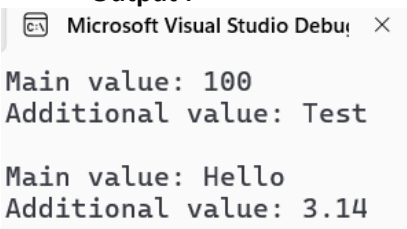
31. Write a simple program to create generic class with generic constructor, generic member variable, generic property and generic method.



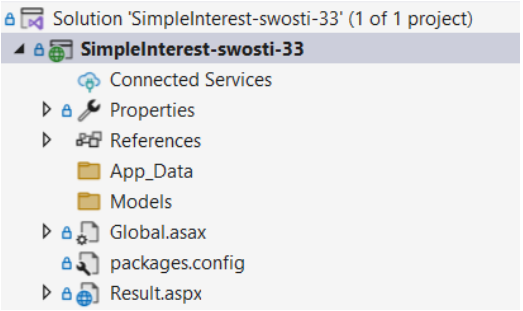
Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace GenericClass_swosti_33
{
    class MyGenericClass<T>
    {
        private T _value;
        public MyGenericClass(T value)
        {
            _value = value;
        }
        public T Value
        {
            get { return _value; }
            set { _value = value; }
        }
        public void Display<U>(U additionalValue)
        {
            Console.WriteLine($"Main value: {_value}");
            Console.WriteLine($"Additional value: {additionalValue}");
        }   }
    class Program
    {
        static void Main(string[] args)
        {
            MyGenericClass<int> intObj = new MyGenericClass<int>(100);
            intObj.Display<string>("Test");
            Console.WriteLine();
            MyGenericClass<string> stringObj = new
      MyGenericClass<string>("Hello");
            stringObj.Display<double>(3.14);
        }
    }       }
```

Output :



32. Write a program to create form for calculating simple interest in one ASP.NET page and display the simple interest.



Source Code :

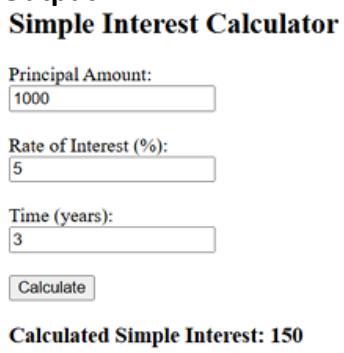
Result.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
        Inherits="SimpleInterestApp.Result" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Simple Interest Calculator</title>
</head>
<body>
<form id="form1" runat="server" style="margin:50px;">
<h2>Simple Interest Calculator</h2>
Principal Amount:<br />
<asp:TextBox ID="txtPrincipal" runat="server"></asp:TextBox><br /><br />
Rate of Interest (%):<br />
<asp:TextBox ID="txtRate" runat="server"></asp:TextBox><br /><br />
Time (years):<br />
<asp:TextBox ID="txtTime" runat="server"></asp:TextBox><br /><br />
<asp:Button ID="btnCalculate" runat="server" Text="Calculate" OnClick="btnCalculate_Click" /><br /><br />
<asp:Label ID="lblResult" runat="server" Font-Bold="True" Font-Size="Large"></asp:Label>
</form>
</body>
</html>
```

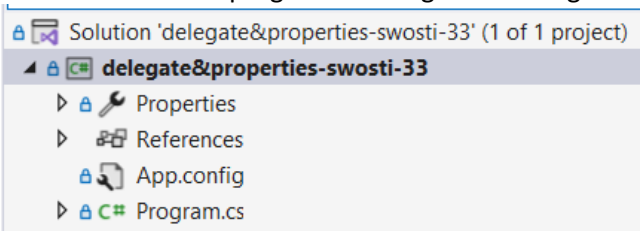
Result.aspx.cs

```
using System;
namespace SimpleInterestApp{
public partial class Default : System.Web.UI.Page{
protected void btnCalculate_Click(object sender, EventArgs e){
try{
double principal = Convert.ToDouble(txtPrincipal.Text);
double rate = Convert.ToDouble(txtRate.Text);
double time = Convert.ToDouble(txtTime.Text);
double interest = (principal * rate * time) / 100;
lblResult.Text = "Calculated Simple Interest: " + interest;}
catch{
lblResult.Text = "Please enter valid numeric values!";
}}}}
```

Output :



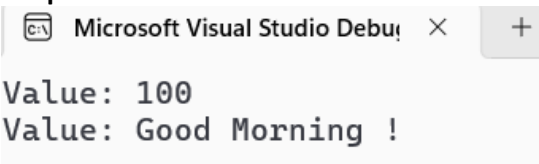
33. Write a C# program create generic delegates and generic properties.



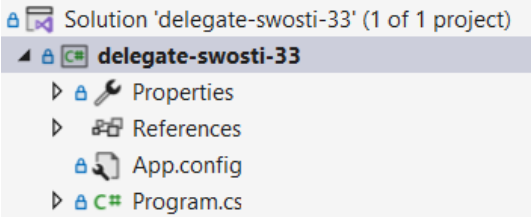
Source Code :

```
using System;
public delegate T MyGenericDelegate<T>(T value);
class Program
{
    public static T Display<T>(T data)
    {
        Console.WriteLine("Value: " + data);
        return data;
    }
    static void Main()
    {
        MyGenericDelegate<int> intDelegate = new
      MyGenericDelegate<int>(Display);
        intDelegate(100);
        MyGenericDelegate<string> stringDelegate = new
      MyGenericDelegate<string>(Display);
        stringDelegate("Good Morning !");
    }  }
```

Output :



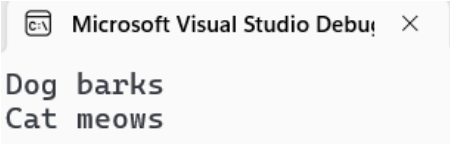
34. Write a C# program to achieve polymorphism using delegates.



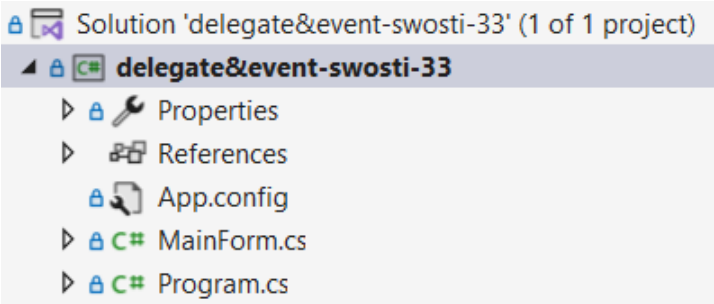
Source Code :

```
using System;
class Program
{
    delegate void SpeakDelegate();
    static void Main()
    {
        SpeakDelegate dogSpeak = () => Console.WriteLine("Dog barks");
        SpeakDelegate catSpeak = () => Console.WriteLine("Cat meows");
        dogSpeak();
        catSpeak();
    }  }
```

Output :



35. C# program that demonstrates the use of delegate and event. Create a Windows Forms application with a button. When the button is clicked, an event is raised, and a message is displayed.



Source Code :

MainForm.cs:

```
using System;
using System.Windows.Forms;
namespace delegate_event_swosti_33{
public partial class MainForm : Form{
public event Action<string> ButtonClicked;
public MainForm(){
InitializeComponent();
ButtonClicked += OnButtonClicked;
}
private void OnButtonClicked(string message){
MessageBox.Show(message, "Event Triggered");
}
private void btnClickMe_Click(object sender, EventArgs e){
ButtonClicked?.Invoke("Button clicked! Event raised!");
}  } }
```

MainForm.Designer.cs:

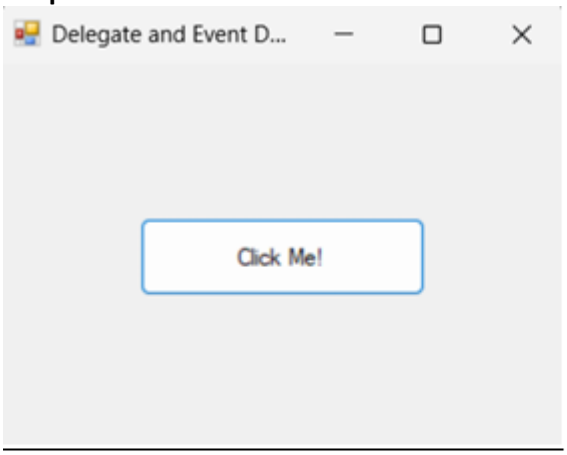
```
namespace delegate_event_swosti_33{
partial class MainForm{
private System.ComponentModel.IContainer components = null;
private Button btnClickMe;
protected override void Dispose(bool disposing){
if (disposing && components != null)
components.Dispose();
base.Dispose(disposing);
}
private void InitializeComponent(){
this.btnClickMe = new Button();
this.SuspendLayout();
this.btnClickMe.Location = new System.Drawing.Point(100, 100);
this.btnClickMe.Name = "btnClickMe";
this.btnClickMe.Size = new System.Drawing.Size(200, 50);
this.btnClickMe.Text = "Click Me!";
this.btnClickMe.Click += this.btnClickMe_Click;
this.ClientSize = new System.Drawing.Size(400, 250);
this.Controls.Add(this.btnClickMe);
this.Text = "Delegate Demo";
this.ResumeLayout(false);
} } }
```

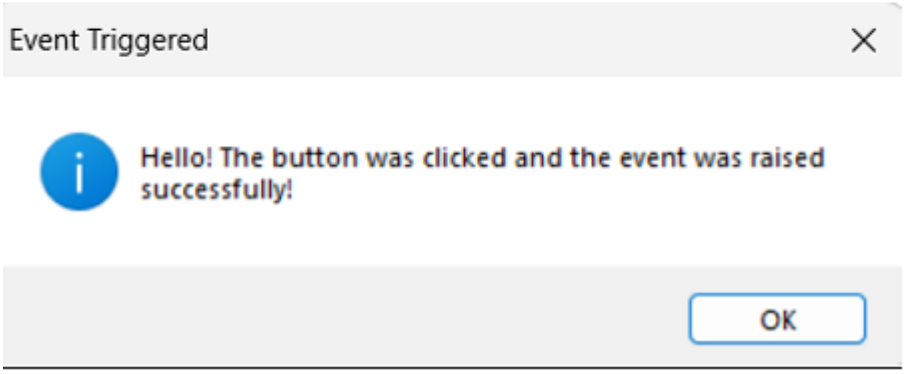
Program.cs

```
using System;
using System.Windows.Forms;
namespace delegate_event_swosti_33{
```

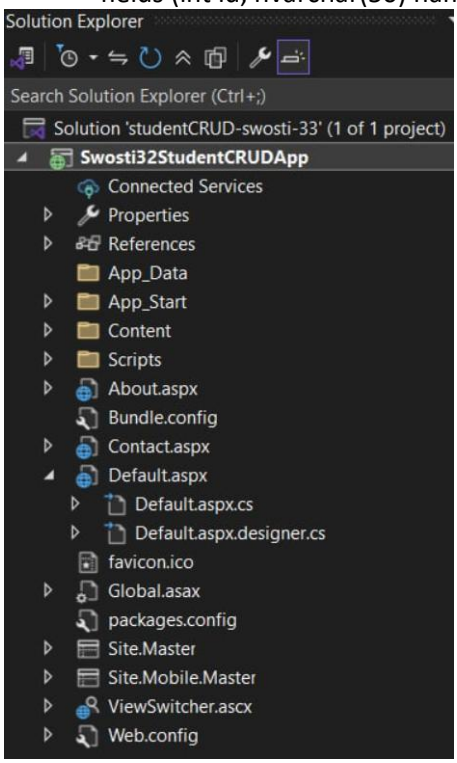
```
internal static class Program{
[STAThread]
static void Main(){
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new MainForm());
} } }
```

Output :





36. Write a C# program to perform (CRUD) Operation from given table (tblStudent) with fields (int id, nvarchar(50) name, int age, nvarchar(50) gender).



Source Code :

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Swosti32StudentCRUDApp.Default" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Student CRUD</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h2>Student CRUD Operations</h2>
ID: <asp:TextBox ID="txtId" runat="server"></asp:TextBox><br /><br />
Name: <asp:TextBox ID="txtName" runat="server"></asp:TextBox><br /><br />
Age: <asp:TextBox ID="txtAge" runat="server"></asp:TextBox><br /><br />
Gender: <asp:TextBox ID="txtGender" runat="server"></asp:TextBox><br /><br /><br />
<asp:Button ID="btnCreate" runat="server" Text="Create" OnClick="btnCreate_Click" />
<asp:Button ID="btnRead" runat="server" Text="Read" OnClick="btnRead_Click" />
<asp:Button ID="btnUpdate" runat="server" Text="Update" OnClick="btnUpdate_Click" />
<asp:Button ID="btnDelete" runat="server" Text="Delete" OnClick="btnDelete_Click" /><br /><br />
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="true"></asp:GridView>
</div>
</form>
</body>
</html>
```

Default.aspx.cs

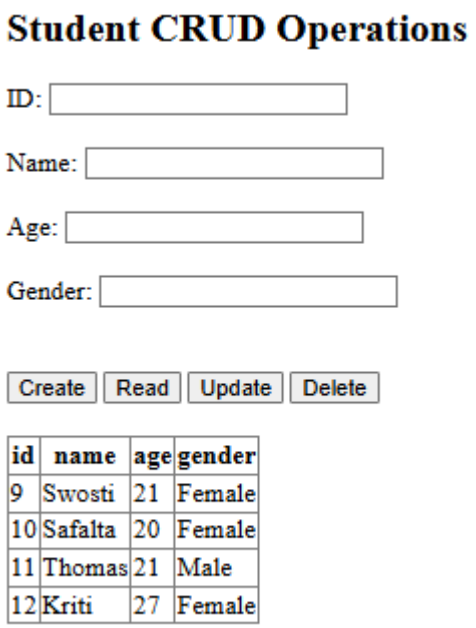
```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Web.UI;
namespace Swosti32StudentCRUDApp {
public partial class Default : Page{
```

```
string connectionString = @"Server=localhost\SQLEXPRESS;Database=swastikDB;Integrated Security=True;";
protected void Page_Load(object sender, EventArgs e){
if (!IsPostBack){
LoadData();
}
}
protected void btnCreate_Click(object sender, EventArgs e){
using (SqlConnection con = new SqlConnection(connectionString)){
string query = "INSERT INTO tblStudent (name, age, gender) VALUES (@name, @age, @gender)";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@name", txtName.Text);
            cmd.Parameters.AddWithValue("@age", txtAge.Text);
            cmd.Parameters.AddWithValue("@gender", txtGender.Text);
            con.Open();
            cmd.ExecuteNonQuery();
        }
        LoadData();
        ClearFields();
    }
    protected void btnRead_Click(object sender, EventArgs e){
        LoadData();
    }
    protected void btnUpdate_Click(object sender, EventArgs e){
        using (SqlConnection con = new SqlConnection(connectionString)){
            string query = "UPDATE tblStudent SET name=@name, age=@age, gender=@gender WHERE id=@id";
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@id", txtId.Text);
            cmd.Parameters.AddWithValue("@name", txtName.Text);
            cmd.Parameters.AddWithValue("@age", txtAge.Text);
            cmd.Parameters.AddWithValue("@gender", txtGender.Text);
            con.Open();
            cmd.ExecuteNonQuery();
        }
        LoadData();
        ClearFields();
    }
    protected void btnDelete_Click(object sender, EventArgs e){
        using (SqlConnection con = new SqlConnection(connectionString)){
            string query = "DELETE FROM tblStudent WHERE id=@id";
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@id", txtId.Text);
            con.Open();
            cmd.ExecuteNonQuery();
        }
        LoadData();
        ClearFields();
    }
    private void LoadData(){
        using (SqlConnection con = new SqlConnection(connectionString)){
            SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM tblStudent", con);
            DataTable dt = new DataTable();
            da.Fill(dt);
            GridView1.DataSource = dt;
            GridView1.DataBind();
        }
    }
    private void ClearFields()
    {
        txtId.Text = "";
        txtName.Text = "";
```

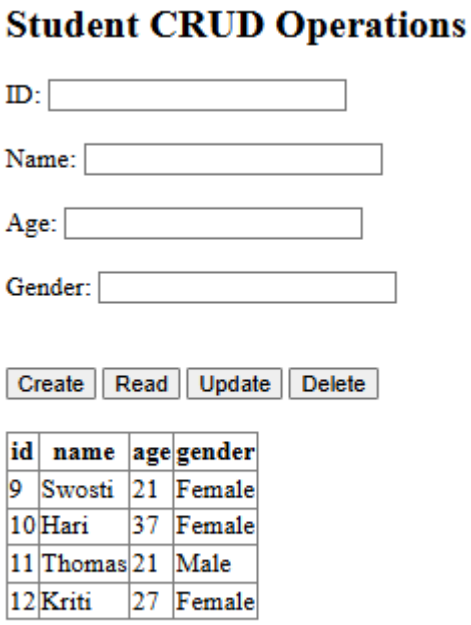
```
        txtAge.Text = "";
        txtGender.Text = "";
} } }
```

Output :

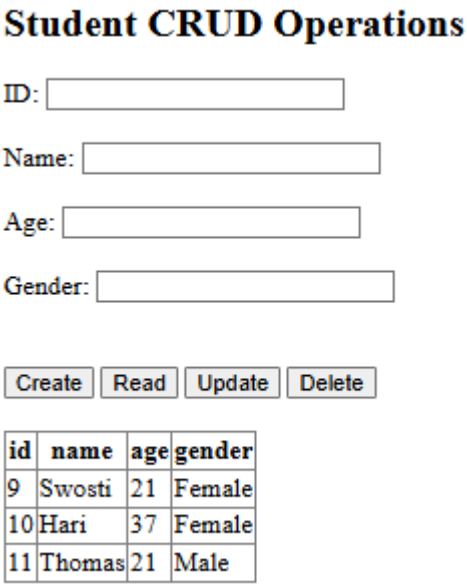
Create



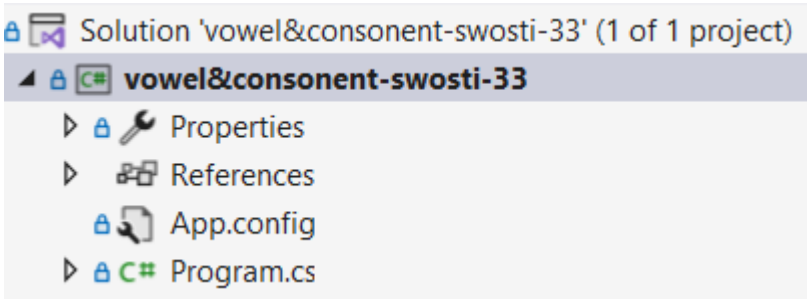
Update



Delete



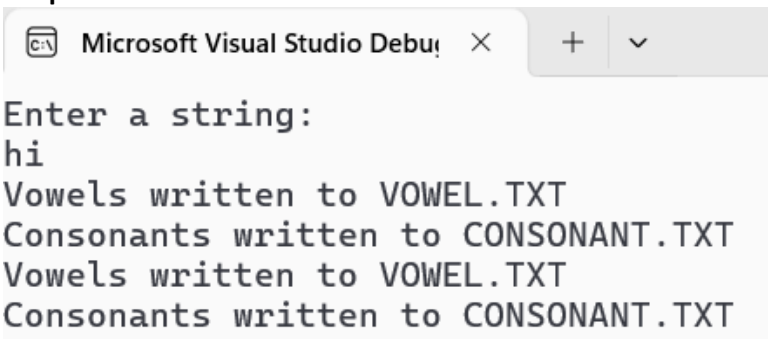
37. Write a program to read an input string from the user and write the vowels of that string in VOWEL.TXT and consonants in CONSONANT.TXT.



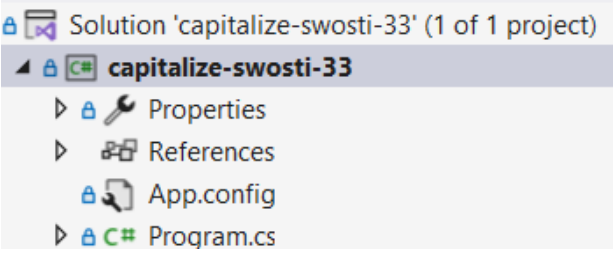
Source Code :

```
using System;
using System.IO;
using System.Linq;
class Program
{
    static void Main()
    {
        Console.WriteLine("Enter a string:");
        string input = Console.ReadLine();
        string vowels = "";
        string consonants = "";
        foreach (char c in input)
        {
            if (char.IsLetter(c))
            {
                char lowerChar = char.ToLower(c);
                if ("aeiou".Contains(lowerChar))
                    vowels += c;
                else
                    consonants += c;
            }
            File.WriteAllText("VOWEL.TXT", vowels);
            File.WriteAllText("CONSONANT.TXT", consonants);
            Console.WriteLine("Vowels written to VOWEL.TXT");
            Console.WriteLine("Consonants written to CONSONANT.TXT");
        }
    }
}
```

Output :



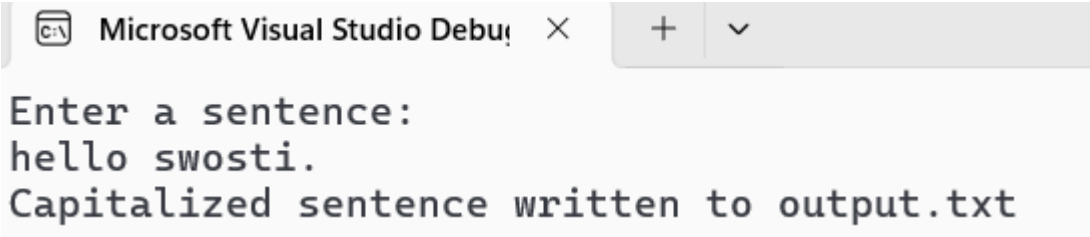
38. Create a C# program that takes a sentence as input from User and capitalizes the first letter of each word and write to output.txt.



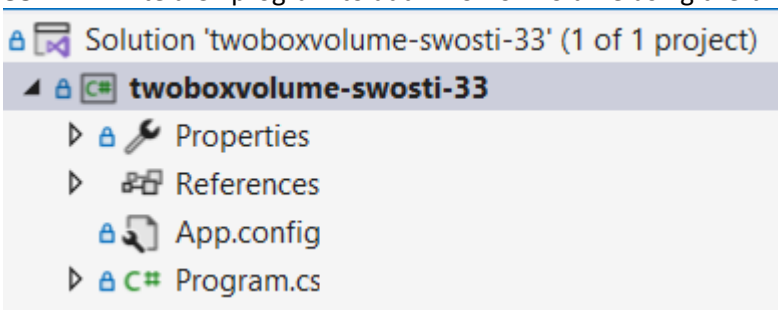
Source Code :

```
using System;
using System.IO;
using System.Globalization;
class Program
{
    static void Main()
    {
        Console.WriteLine("Enter a sentence:");
        string input = Console.ReadLine();
        TextInfo textInfo = CultureInfo.CurrentCulture.TextInfo;
        string capitalized = textInfo.ToTitleCase(input.ToLower());
        File.WriteAllText("output.txt", capitalized);
        Console.WriteLine("Capitalized sentence written to output.txt");
    }
```

Output :



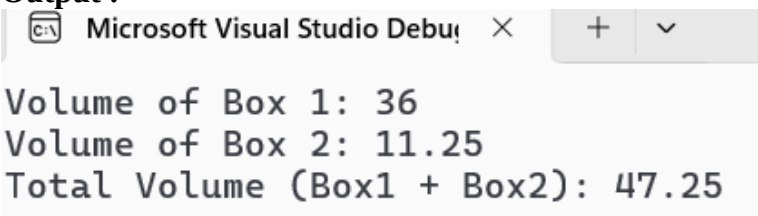
39. Write a C# program to add Two Box Volume using the binary operator.



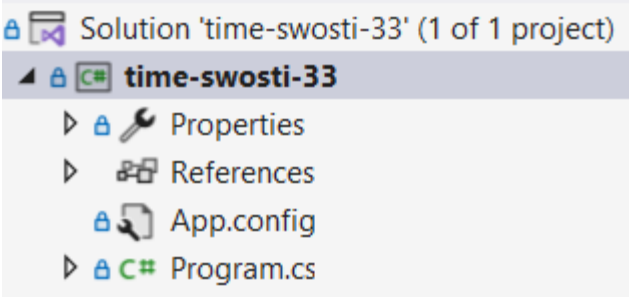
Source Code :

```
using System;
class Box
{
    public double Length { get; }
    public double Width { get; }
    public double Height { get; }
    public Box(double length, double width, double height)
    {
        Length = length;
        Width = width;
        Height = height;
    }
    public double Volume()
    {
        return Length * Width * Height;
    }
    public static double operator +(Box b1, Box b2)
    {
        return b1.Volume() + b2.Volume();
    }
}
class Program
{
    static void Main()
    {
        Box box1 = new Box(3, 3, 4);
        Box box2 = new Box(1.5, 2.5, 3);
        double totalVolume = box1 + box2;
        Console.WriteLine($"Volume of Box 1: {box1.Volume()}");
        Console.WriteLine($"Volume of Box 2: {box2.Volume()}");
        Console.WriteLine($"Total Volume (Box1 + Box2): {totalVolume}");
    }
}
```

Output :



40. 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.



Source Code :

```
using System;
class Time
{
   public int Hours { get; set; }
   public int Minutes { get; set; }
   public int Seconds { get; set; }
   public Time(int h, int m, int s)
   {
      Hours = h;
      Minutes = m;
      Seconds = s;
      NormalizeTime();
   }
   public void DisplayTime()
   {
      Console.WriteLine($"{Hours:D2}:{Minutes:D2}:{Seconds:D2}");
   }
   private void NormalizeTime()
   {
      if (Seconds >= 60)
      {
         Minutes += Seconds / 60;
         Seconds %= 60;
      }
      if (Minutes >= 60)
      {
         Hours += Minutes / 60;
         Minutes %= 60;
      }
      Hours %= 24;    }
   public static Time operator +(Time t1, Time t2)
   {
      return new Time(
         t1.Hours + t2.Hours,
         t1.Minutes + t2.Minutes,
         t1.Seconds + t2.Seconds
      );
   }
   public static Time operator -(Time t1, Time t2)
   {
      int totalSeconds1 = t1.Hours * 3600 + t1.Minutes * 60 + t1.Seconds;
      int totalSeconds2 = t2.Hours * 3600 + t2.Minutes * 60 + t2.Seconds;
      int diff = totalSeconds1 - totalSeconds2;
      if (diff < 0) diff += 24 * 3600;
```

```
        int h = diff / 3600;
        int m = (diff % 3600) / 60;
        int s = diff % 60;
        return new Time(h, m, s);
    }
    public static bool operator ==(Time t1, Time t2)
    {
        return t1.Hours == t2.Hours && t1.Minutes == t2.Minutes && t1.Seconds == t2.Seconds;
    }
    public static bool operator !=(Time t1, Time t2)
    {
        return !(t1 == t2);
    }
    public override bool Equals(object obj)
    {
        if (obj is Time t)
            return this == t;
        return false;
    }
    public override int GetHashCode()
    {
        return Hours ^ Minutes ^ Seconds;
    }
}
class Program
{
    static void Main()
    {
        Time t1 = new Time(10, 45, 50);
        Time t2 = new Time(2, 30, 15);
        Console.Write("Time 1: ");
        t1.DisplayTime();
        Console.Write("Time 2: ");
        t2.DisplayTime();
        Time t3 = t1 + t2;
        Console.Write("After Addition: ");
        t3.DisplayTime();
        Time t4 = t1 - t2;
        Console.Write("After Subtraction: ");
        t4.DisplayTime();
        Console.WriteLine("Are t1 and t2 equal? " + (t1 == t2));
    }
}
```

Output :