1. Implement C# Program to Calculates the Distance Travelled by Reading Speed and Time.

Ans - using System;

class DistanceCalculator

{

static void Main(string[] args)

{

double speed, time, distance;

Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

Console.WriteLine("Distance Calculator");

Console.WriteLine("-------------------");

Console.Write("Enter speed (in miles per hour): ");

speed = Convert.ToDouble(Console.ReadLine());

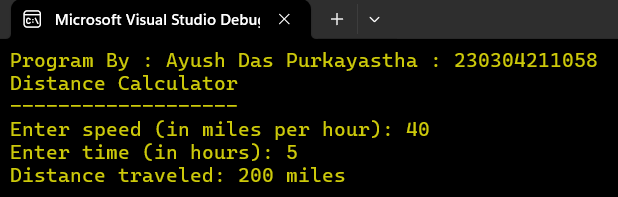
Console.Write("Enter time (in hours): ");

time = Convert.ToDouble(Console.ReadLine());

distance = speed \* time;

Console.WriteLine("Distance traveled: " + distance + " miles");

}

}

1. Implement C# Program to calculate perimeter of circle and rectangle.

Ans - using System;

class Perimeter

{

static void Main(String[] args)

{

Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

Console.WriteLine("Circle");

Console.WriteLine("Enter Radius of the Circle");

double radius = Convert.ToDouble(Console.ReadLine());

double pi = Math.PI;

double perimeterC = 2 \* pi \* radius;

Console.WriteLine("Perimeter of Circle with Radius {0} = {1:0.00}",radius,perimeterC);

Console.WriteLine("Rectangle");

Console.WriteLine("Enter Length of the Rectangle");

double length = Convert.ToDouble(Console.ReadLine());

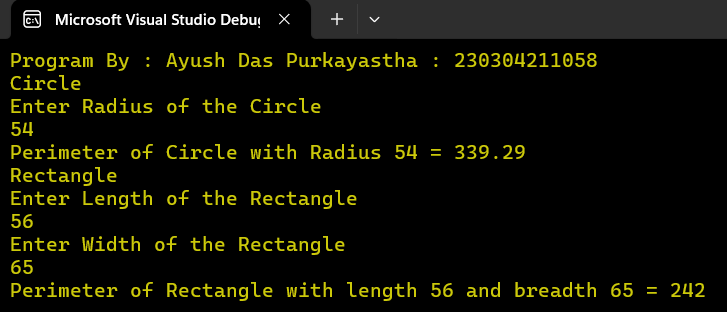
Console.WriteLine("Enter Width of the Rectangle");

double width = Convert.ToDouble(Console.ReadLine());

double perimeterR = 2 \* (length + width);

Console.WriteLine("Perimeter of Rectangle with length {0} and breadth {1} = {2}", length, width, perimeterR);

}}



1. Implement C# Program to check whether the entered number is even or odd.

Ans - using System;

class oddEven

{

static void Main(String[] args)

{

Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

oddEven:

System.Console.WriteLine("Enter a Nunber");

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

if (n % 2 == 0)

{

Console.WriteLine("{0} is Even", n);

}

else

{

Console.WriteLine("{0} is Odd", n);

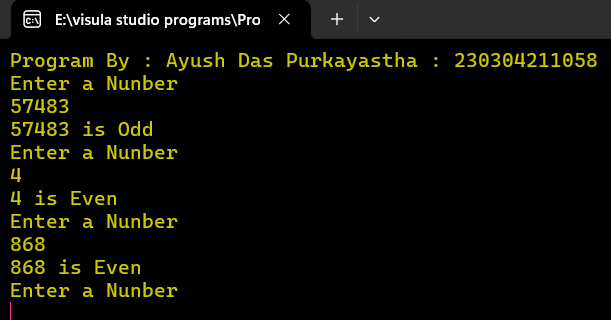
}

while (true)

{

goto oddEven;

}}}



1. Implement C# Program to check whether the entered number is an Armstrong number or not.

Ans - using System;

public class ArmstrongExample

{

public static void Main(string[] args)

{

Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

int n, r, sum = 0, temp;

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

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

temp = n;

Console.Write("Each digit and its cube: ");

while (n > 0)

{r = n % 10;

sum = sum + (r \* r \* r);

Console.Write(r + " \*\* 3 ");

n = n / 10;

if (n > 0)

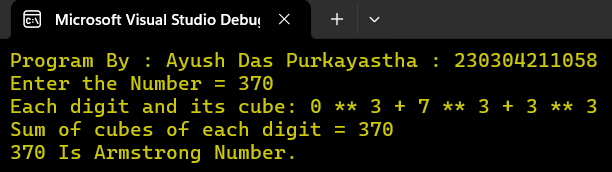
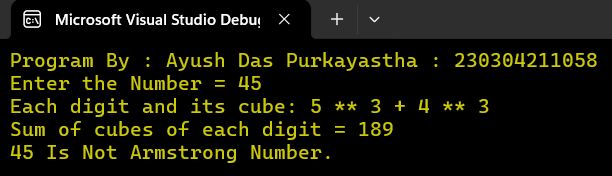
{Console.Write("+ ");}}

Console.WriteLine();

Console.WriteLine("Sum of cubes of each digit = " + sum);

if (temp == sum)

{Console.WriteLine("{0} Is Armstrong Number.", temp);}

 else{Console.WriteLine("{0} Is Not Armstrong Number.", temp);}}}

1. Implement C# Program to Reads a Grade & Display the Equivalent Description. (Switch Case)

Ans - using System;

using System.Reflection.PortableExecutable;

class SwitchCase

{

public static void Main(string[] args)

{

Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

GradeCheck:

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

char grades = char.Parse(Console.ReadLine());

char upper = char.ToUpper(grades);

Console.WriteLine("Grade = {0}", upper);

switch (upper)

{

case 'A':

Console.WriteLine("Very Good");

break;

case 'B':

Console.WriteLine("Good");

break;

case 'C':

Console.WriteLine("Average");

break;

case 'D':

Console.WriteLine("Fail");

break;

default:

Console.WriteLine("Invalid Grades! Grades should be A or B or C or D");

break;

}

while (true)

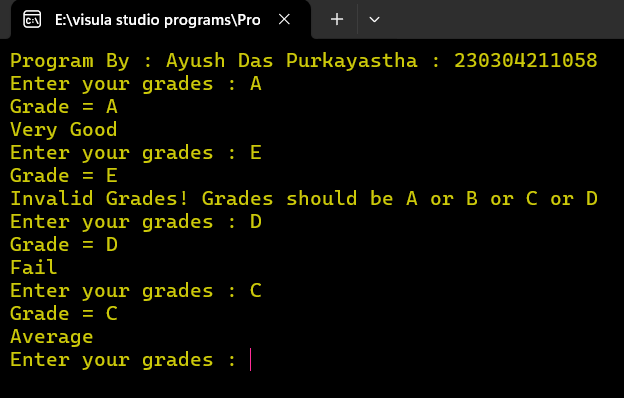
{

goto GradeCheck;

}

}

}



1. Implement C# Program to Displays the ATM Transaction.

Here The types of ATM transaction are

1) Balance checking

2) Cash withdrawal

3) Cash deposition.

You can opt any of the above transaction according to your need of transaction. (Switch Case)

Ans - using System;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

Console.WriteLine("Welcome to the ATM");

Console.WriteLine("Select an option:");

Console.WriteLine("1. Balance Checking");

Console.WriteLine("2. Cash Withdrawal");

Console.WriteLine("3. Cash Deposition");

int choice;

if (int.TryParse(Console.ReadLine(), out choice))

{

switch (choice)

{

case 1:

Console.WriteLine("You selected Balance Checking");

break;

case 2:

Console.WriteLine("You selected Cash Withdrawal");

break;

case 3:

Console.WriteLine("You selected Cash Deposition");

break;

default:

Console.WriteLine("Invalid choice. Please select a valid option.");

break;}

}

else

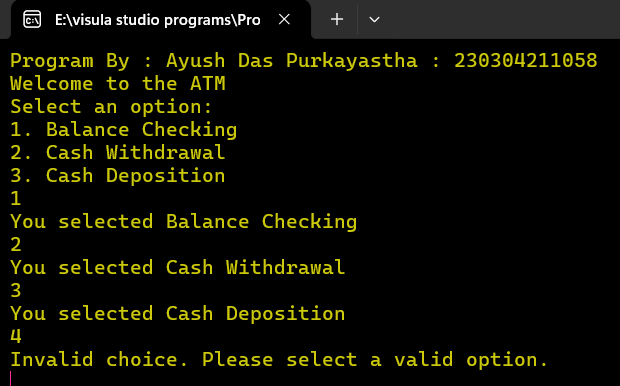
{

Console.WriteLine("Invalid input. Please enter a number.");

}

}

}



1. Implement C# Program Prints a Binary Triangle. (Using For Loop)

Problem Solution: Binary Triangle is a Triangle formed with 1’s and 0’s. Number of rows in the binary triangle is obtained from the user.

Ans - using System;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

Console.WriteLine("Enter the number of rows for the binary triangle:");

for (int i = 1; i <= 5; i++)

{

for (int j = 1; j <= i; j++)

{

if (j % 2 == 0)

{

Console.Write("0");

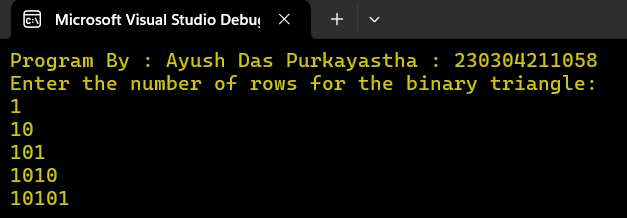
}

else

{

Console.Write("1");

}

 } Console.WriteLine(); }}

1. Implement c# program to demonstrate the use of Continue and goto Statement.

Ans - using System;

class Program

{static void Main(string[] args)

{Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

Console.WriteLine("Demonstrating the use of 'continue' statement:");

for (int i = 1; i <= 5; i++)

{

if (i == 3){

Console.WriteLine("Skipping iteration {0}", i);

continue; }

Console.WriteLine("Iteration {0}", i);}

Console.WriteLine();

Console.WriteLine("Demonstrating the use of 'goto' statement:");

int j = 1;

goto ExampleLabel;

ExampleLabel:

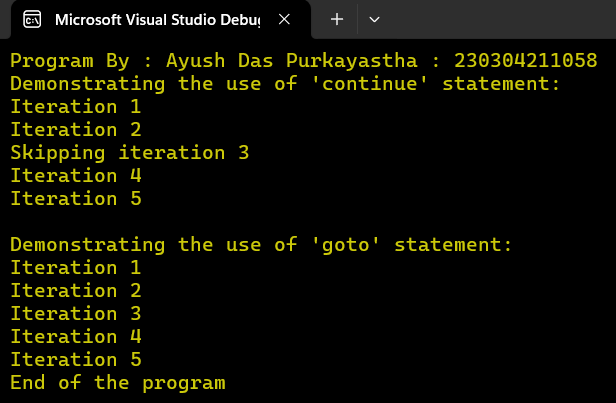
if (j <= 5){

Console.WriteLine("Iteration {0}", j);

j++;

goto ExampleLabel;}

Console.WriteLine("End of the program”)}}



1. Implement C# Program Gets a Number and Display the Number with its Reverse. (Using While Loop)

Ans - using System;

class ReverseNumber

{public static void Main(string[] args)

{Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

Reverse:

Console.Write("Enter a positive number : ");

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

int number = n;

int reverse = 0;

if (n >= 0)

{while (n != 0)

{int digit = n % 10;

reverse = reverse \* 10 + digit;

n = n / 10;}

Console.WriteLine("Reverse of {0} is {1}", number, reverse);

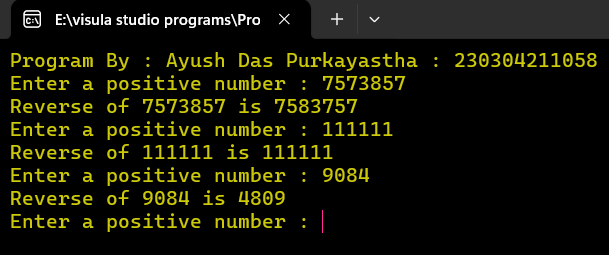
}else

{Console.WriteLine("Invalid Input! Please enter a positive number.");

goto Reverse;}

while (true){

goto Reverse;}}}



1. Implement C# Program to reverse an array. (Using for each Loop)

Ans - using System;

class ReverseArray

{

public static void Main(string[] args)

{

Console.WriteLine("Program By : Ayush Das Purkayastha : 230304211058");

string[] arr = { "Orange", "Mango", "Banana", "Apple" };

var reverseArr = arr.Reverse();

Console.WriteLine("----------------");

Console.WriteLine("Original");

Console.WriteLine("----------------");

foreach (var item in arr)

{

Console.WriteLine(item);

}

Console.WriteLine("----------------");

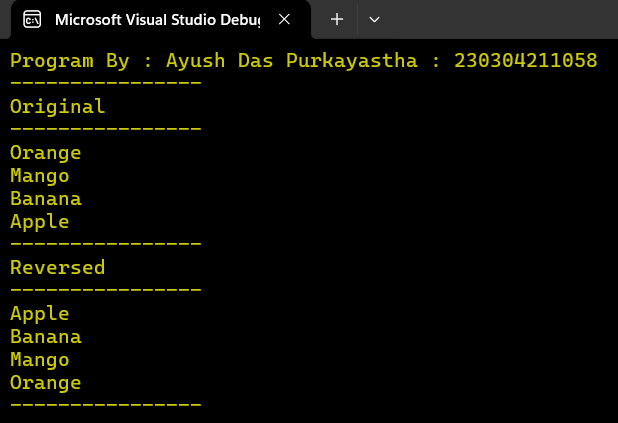
Console.WriteLine("Reversed");

Console.WriteLine("----------------");

foreach (var i in reverseArr)

{Console.WriteLine(i);

} Console.WriteLine("----------------"); }}



1. Implement C# Program to search an element in an array.

Ans - using System;

class Search

{

public static void Main(string[] args)

{

string[] colors = { "Red", "Green", "Blue" };

Console.WriteLine("Enter the element to search");

string element = Console.ReadLine();

string upper = element.ToUpper();

bool found = false;

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

if (colors[i].ToUpper() == upper)

{

Console.WriteLine("Element found at {0} position",i);

found = true;

break;

} }

if (!found)

{

Console.WriteLine("Item not found!"); } }}

1. Implement C# Program to Demonstrates Jagged Arrays.

Ans - using System;

public class JaggedArray

{

public static void Main(string[] args)

{

int[][] jaggedArr = new int[3][];

jaggedArr[0] = new int[] { 1, 2, 3 };

jaggedArr[1] = new int[] { 1, 2, 3, 5, 4 };

jaggedArr[2] = new int[] { 1, 2, 2, 3 };

Console.WriteLine("Jagged Array");

Console.WriteLine("------------");

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

{

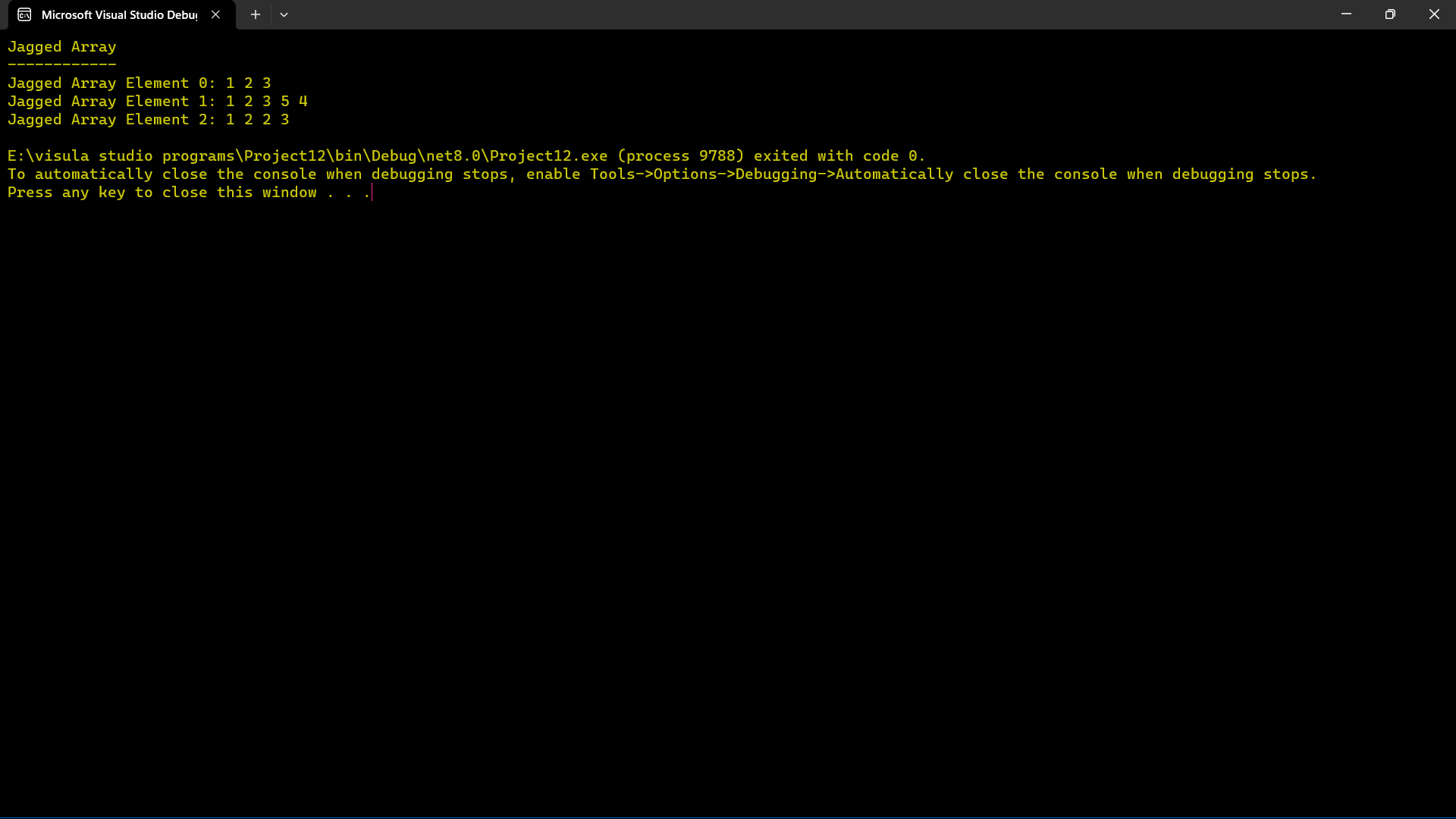
Console.Write("Jagged Array Element {0}: ", i);

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

{

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

}

 Console.WriteLine("");} }}

1. Implement c# program to list all substrings in a given string.

Ans - using System;

class SubstringLister

{

public static void Main(string[] args)

{

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

string input = Console.ReadLine();

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

{

for (int j = i + 1; j <= input.Length; j++)

{

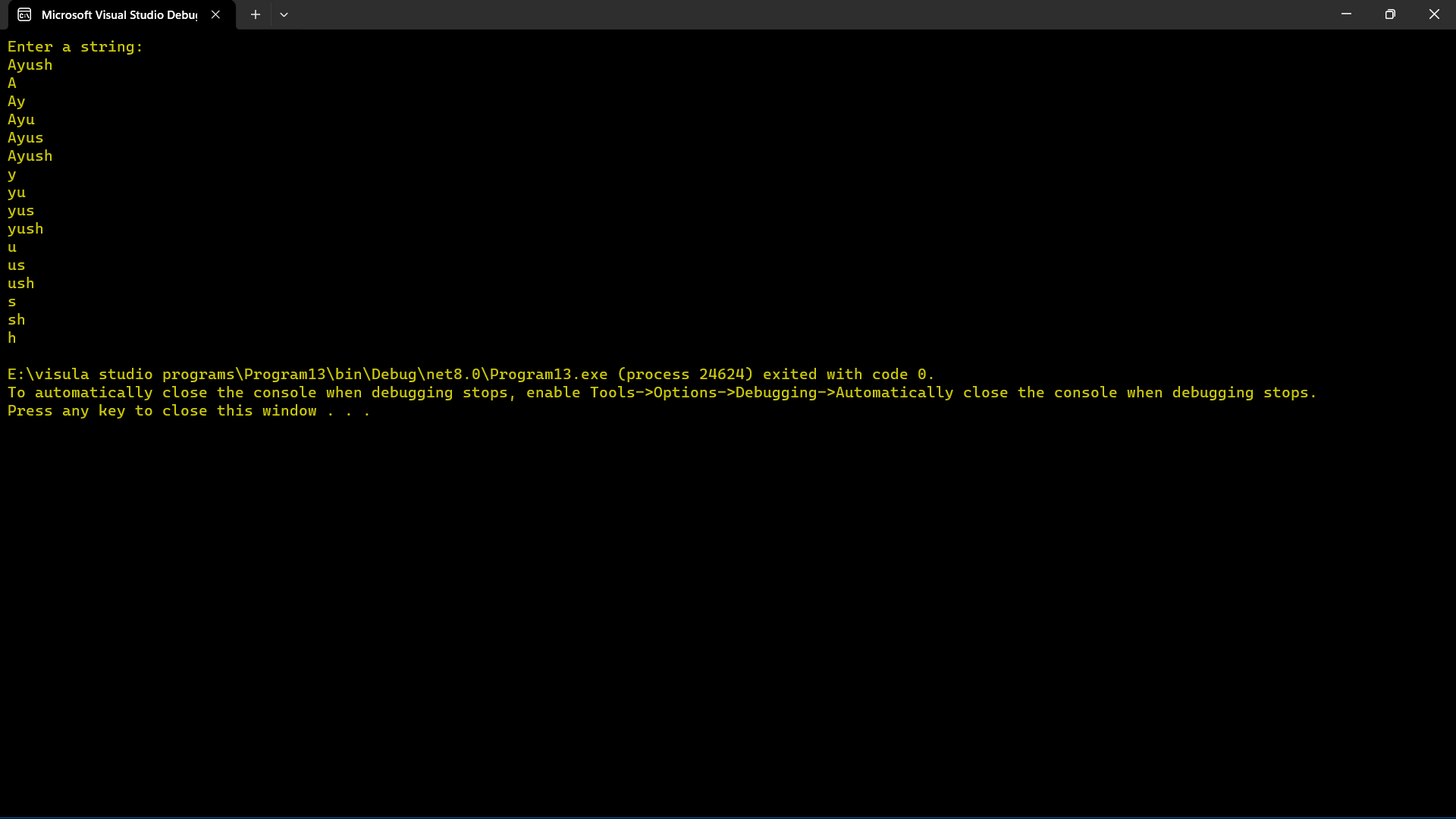
Console.WriteLine(input.Substring(i, j - i));

}

}

}

}



1. Implement c# program to demonstrate the use of following String Methods.

ToUpper(), ToLOwer(), Concat(), IndexOf(), Substring()

Ans - using System;

public class StringMethods

{

public static void Main(string[] args)

{

Methods:

Console.WriteLine("Enter a String");

string str = Console.ReadLine();

Console.WriteLine("Original : " + str);

Console.WriteLine("ToUpper : " + str.ToUpper());

Console.WriteLine("ToLower : " + str.ToLower());

Console.WriteLine("Enter string to concatenate");

string newStr = Console.ReadLine();

Console.WriteLine("Concat : " + string.Concat(str," ",newStr));

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

{

char element = str[i];

Console.WriteLine("Index Of "+element+" : "+str.IndexOf(element));

}

Console.WriteLine("Enter lower bound of substring ");

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

Console.WriteLine("Enter upper bound of substring ");

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

if (lower >= 0 && upper <= str.Length && lower <= upper)

{

Console.WriteLine("Substring : " + str.Substring(lower, upper - lower));

}

else

{

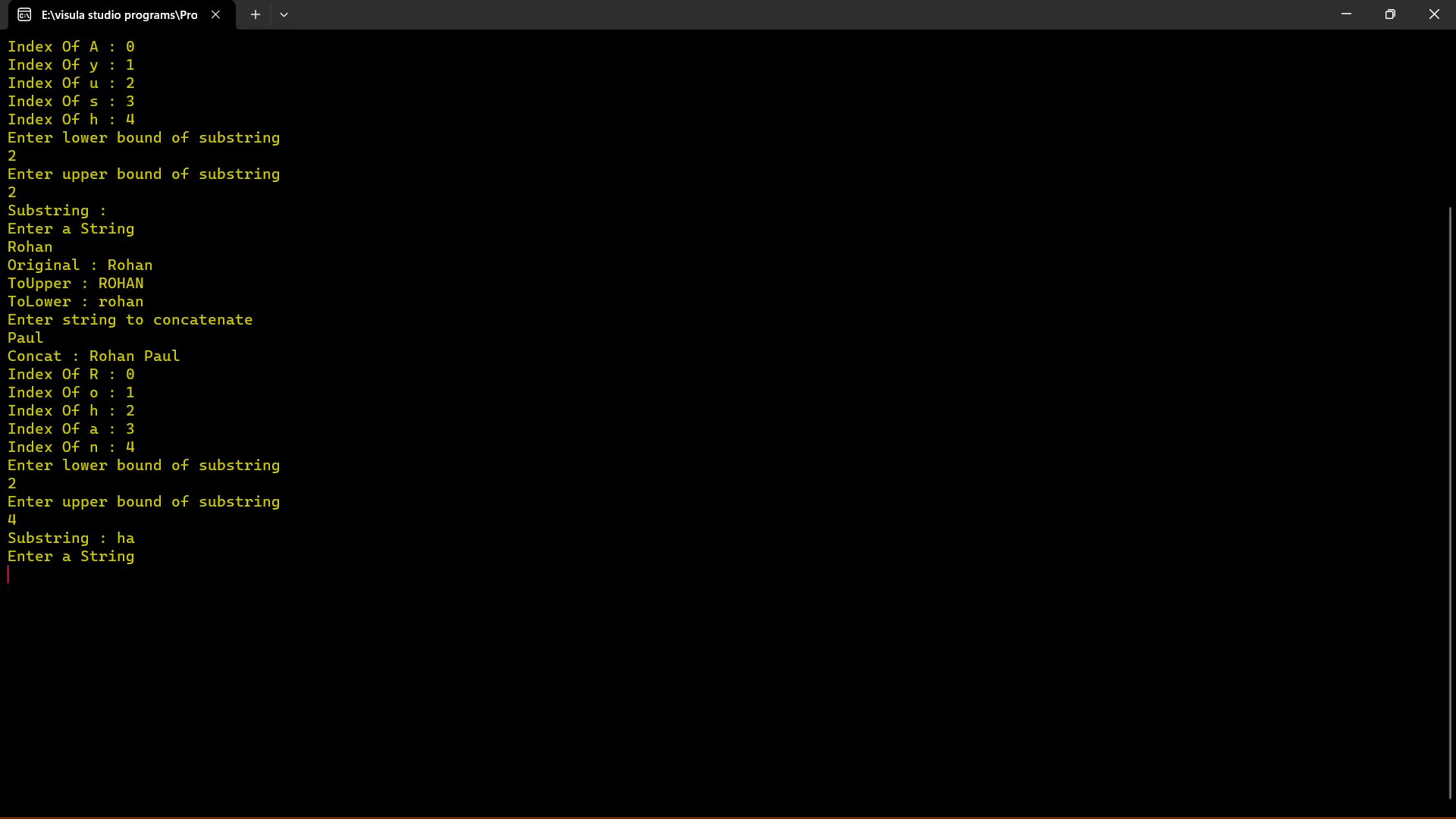
Console.WriteLine("Invalid Lower and Upper Bound!");

}

goto Methods;

}

}



15. Implement c# program to demonstrate the use of Access Modifiers.

Ans - namespace AccessModifiers

{

class AccessMod

{

public int num1;

private int num2;

public int GetNum2()

{

Console.WriteLine("Enter Priavte Value");

num2 = Convert.ToInt32(Console.ReadLine());

return num2;

}

}

class Base

{

protected int num3;

}

class Derive : Base

{

internal int num4;

static void Main()

{

AccessMod obj1 = new AccessMod();

Console.WriteLine("Enter Public Value");

obj1.num1 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Public Value is {0}", obj1.num1);

Derive obj2 = new Derive();

Console.WriteLine("Enter Protected Value");

obj2.num3 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Internal Value");

obj2.num4 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Protected value is {0} ", obj2.num3);

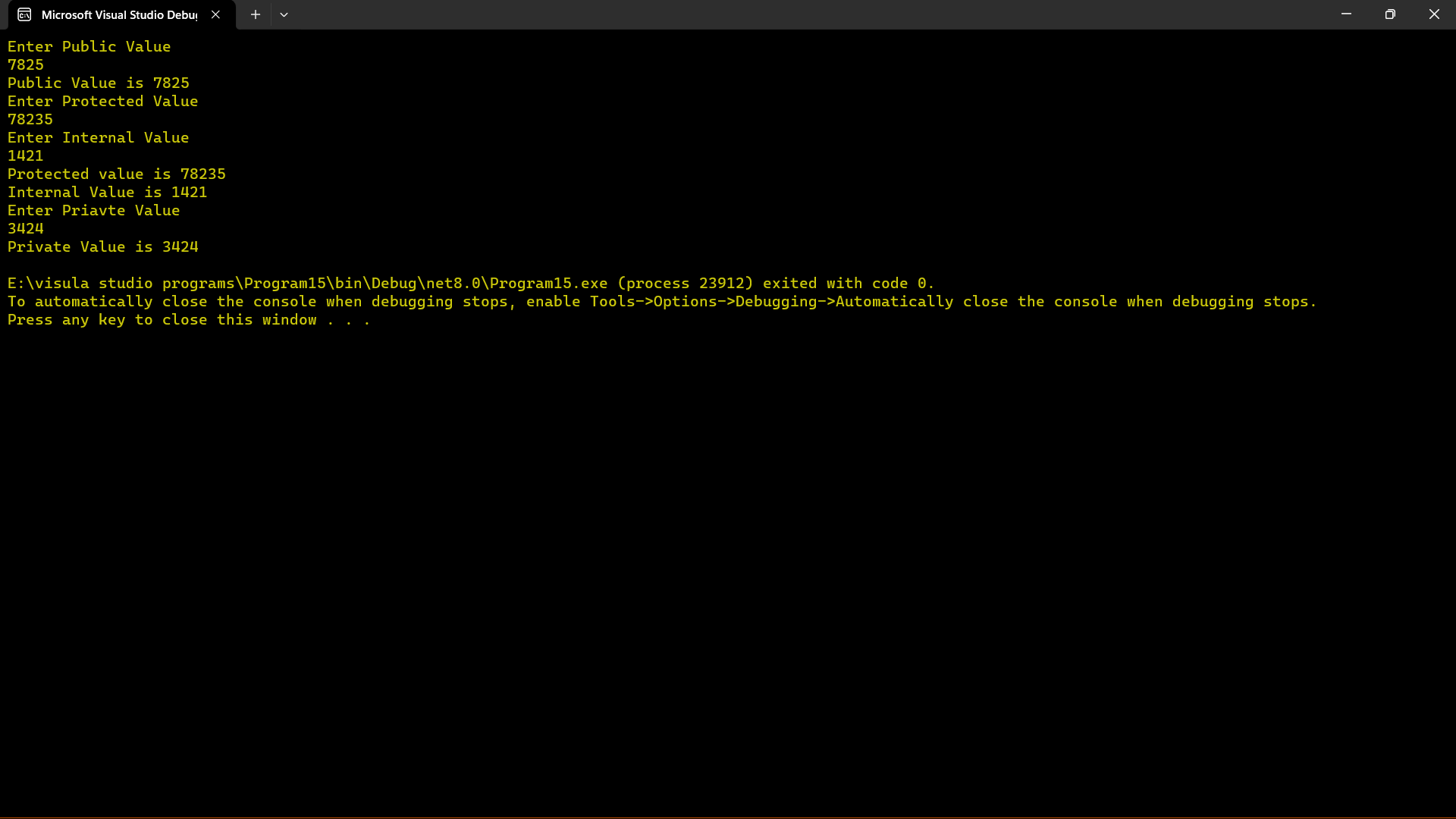
Console.WriteLine("Internal Value is {0} ", obj2.num4);

Console.WriteLine("Private Value is {0} ", obj1.GetNum2());

}

}

}



1. Implement c# program to demonstrate the use of Constructor and Destructor.

Ans - using System;

class MyClass

{

public MyClass()

{

Console.WriteLine("Constructor called. Object created.");

}

~MyClass()

{

Console.WriteLine("Destructor called. Object destroyed.");

}}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Creating object...");

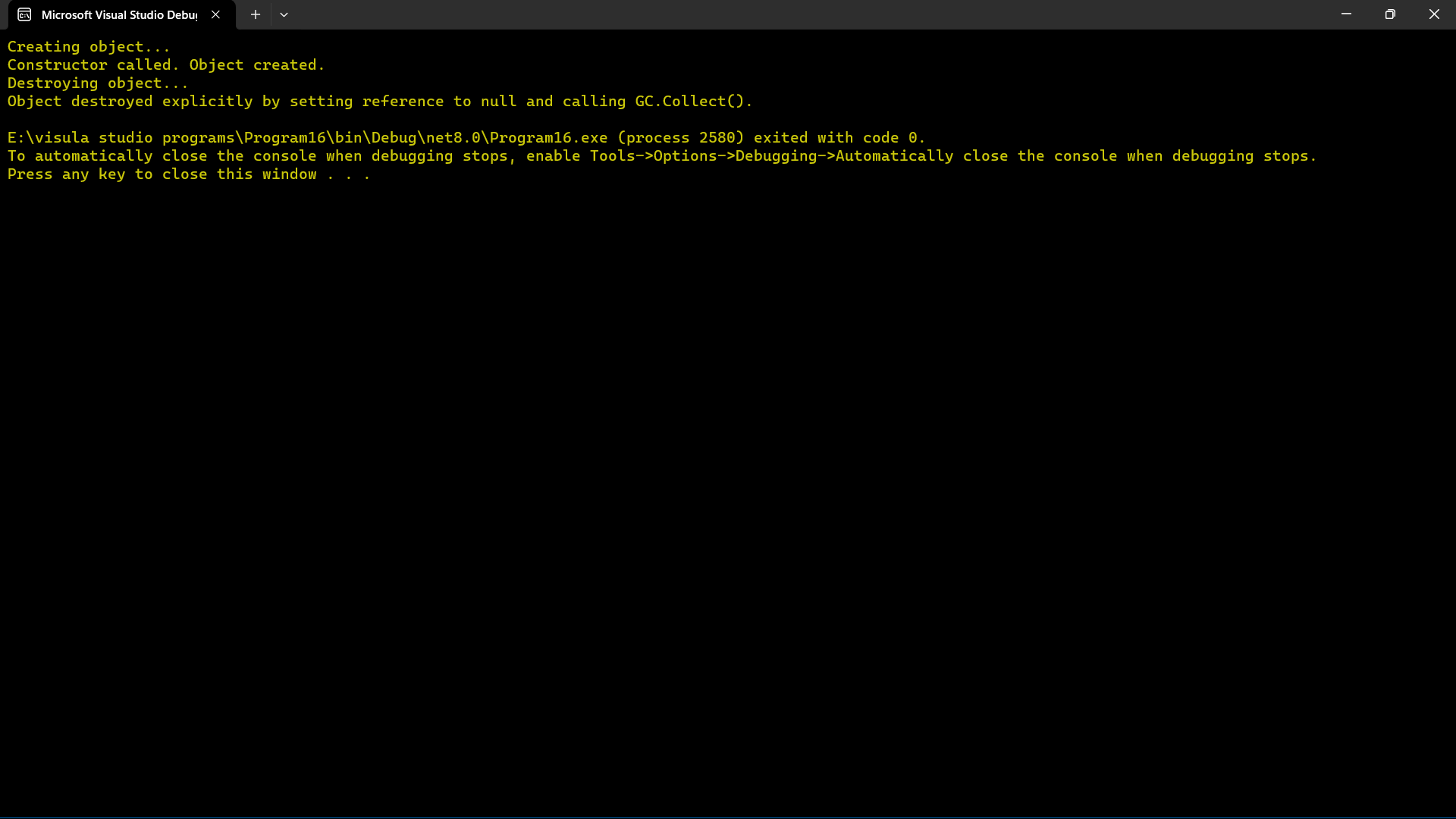
MyClass obj = new MyClass();

Console.WriteLine("Destroying object...");

obj = null; // Set object reference to null to destroy it explicitly

GC.Collect(); // Trigger garbage collection to call destructor

Console.WriteLine("Object destroyed explicitly by setting reference to null and calling GC.Collect().");

 }}

1. Implement c# program to demonstrate the use of Inheritance (Single Inheritance, Multi-Level Inheritance)

Ans - using System;

class Animal

{

public void Eat()

{

Console.WriteLine("Animal is eating...");

}

public void Sleep()

{

Console.WriteLine("Animal is sleeping...");

}}

class Dog : Animal

{

public void Bark()

{

Console.WriteLine("Dog is barking...");

}

}

class Labrador : Dog

{

public void DisplayInfo()

{

Console.WriteLine("Labrador is a type of dog.");

}}

class Program

{

static void Main(string[] args)

{

// Single inheritance example

Dog myDog = new Dog();

myDog.Eat(); // Inherited from Animal

myDog.Sleep(); // Inherited from Animal

myDog.Bark(); // Defined in Dog

Console.WriteLine();

Labrador myLabrador = new Labrador();

myLabrador.Eat(); // Inherited from Animal

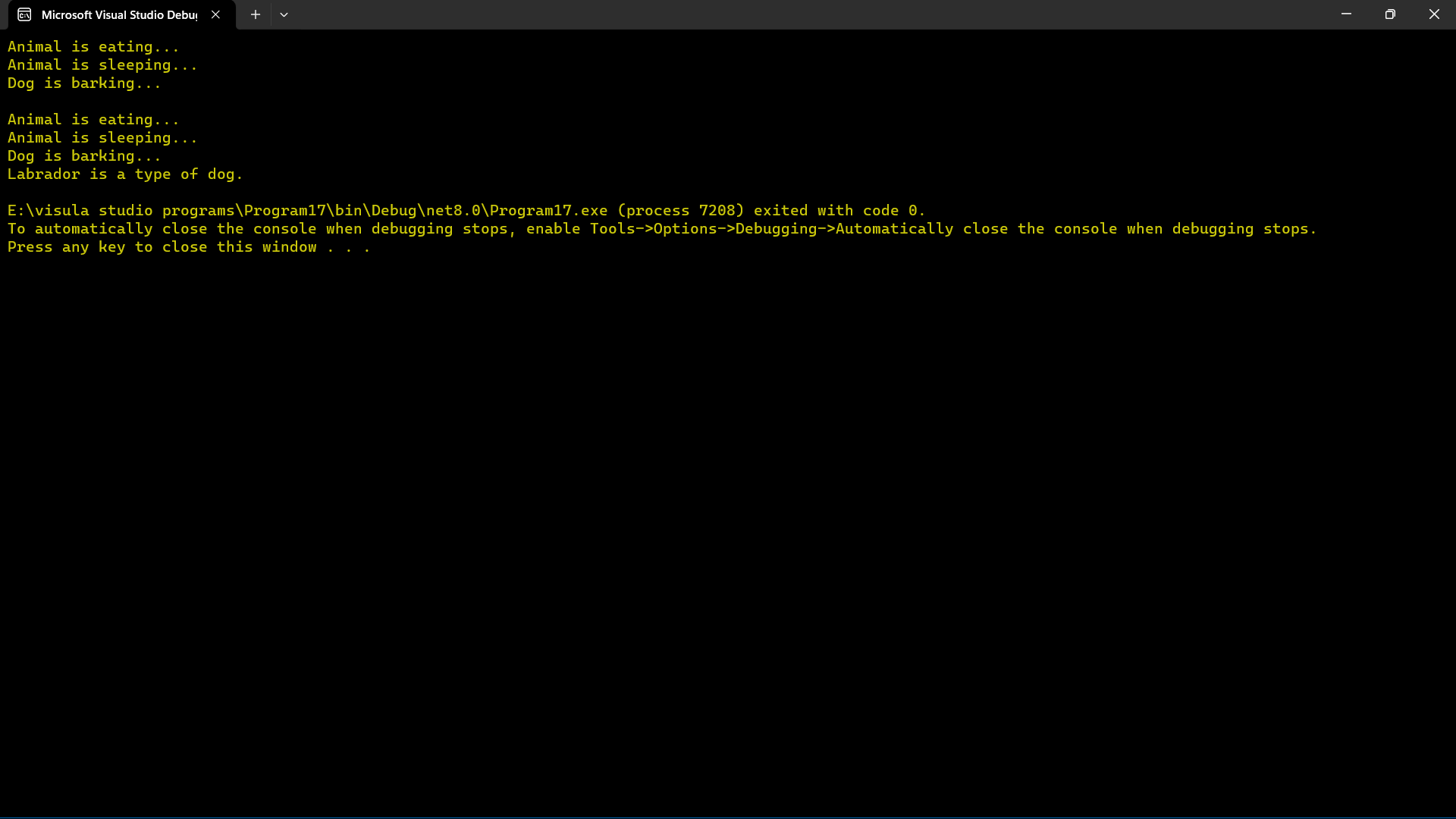
myLabrador.Sleep(); // Inherited from Animal

myLabrador.Bark(); // Inherited from Dog

myLabrador.DisplayInfo(); // Defined in Labrador

}

}



1. Implement c# program to demonstrate the use of Polymorphism (Method Overloading and Method Overriding)

Ans - using System;

class Sum

{

public void add()

{

Console.WriteLine("Enter a number");

int n1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter another number");

int n2 = int.Parse(Console.ReadLine());

int sum = n1 + n2;

Console.WriteLine("Sum of {0} and {1} is {2}", n1, n2, sum);

Console.WriteLine("------------");

}

public void add(int n1, int n2)

{

int sum = n1 + n2;

Console.WriteLine("Sum of {0} and {1} is {2}", n1, n2, sum);

Console.WriteLine("------------");

}

}

class Multiply:Sum

{

public void add(int n1, int n2)

{

int product = n1 \* n2;

Console.WriteLine("Product of {0} and {1} is {2}", n1, n2, product);

Console.WriteLine("------------");

}

}

class Calling {

public static void Main(string[] args)

{

Console.WriteLine("Overloading");

Console.WriteLine("------------");

Sum s = new Sum();

s.add();

s.add(5, 4);

Console.WriteLine("------------");

Console.WriteLine("Overriding");

Console.WriteLine("------------");

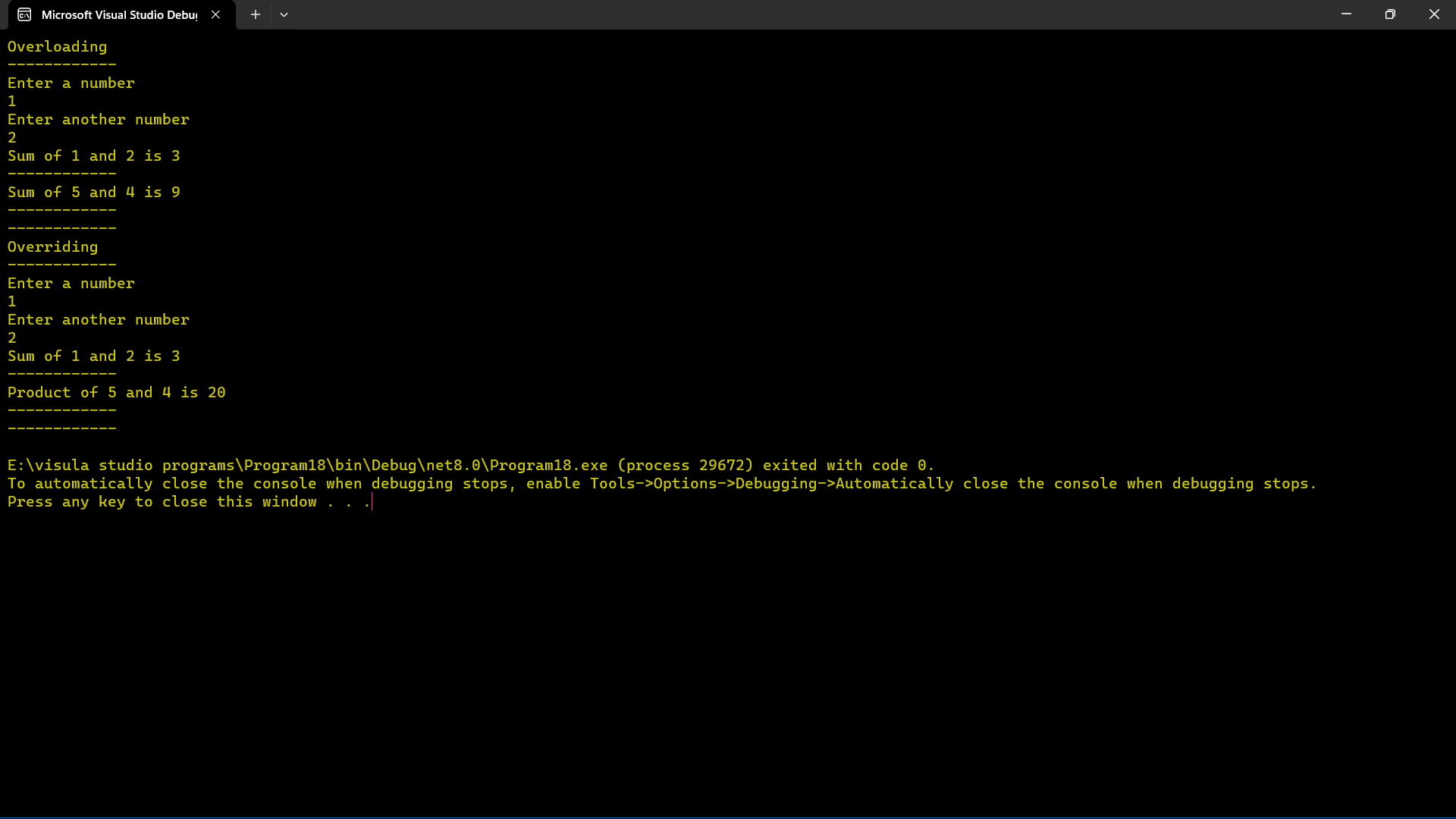
Multiply m = new Multiply();

m.add();

m.add(5,4);

Console.WriteLine("------------");

}

}

1. Implement c# program to demonstrate the use of the following methods of ArrayList Class.

Add(), Insert(), Remove(), RemoveAt(), Sort(), Reverse(), Contains(), IndexOf()

Ans - using System;

using System.Collections;

class Program

{

static void Main(string[] args)

{

ArrayList list = new ArrayList();

list.Add(10);

list.Add(20);

list.Add(30);

list.Add(40);

list.Add(50);

Console.WriteLine("ArrayList elements after adding :");

DisplayArrayList(list);

list.Insert(2, 25);

Console.WriteLine("ArrayList elements after adding and inserting:");

DisplayArrayList(list);

list.Remove(40);

Console.WriteLine("\nArrayList elements after removing 40:");

DisplayArrayList(list);

list.RemoveAt(0);

Console.WriteLine("\nArrayList elements after removing index 0:");

DisplayArrayList(list);

list.Sort();

Console.WriteLine("\nArrayList elements after sorting:");

DisplayArrayList(list);

list.Reverse();

Console.WriteLine("\nArrayList elements after reversing:");

DisplayArrayList(list);

Console.WriteLine("\nIs 25 present in the ArrayList? " + list.Contains(25));

Console.WriteLine("Index of 30 in the ArrayList: " + list.IndexOf(30));

}

static void DisplayArrayList(ArrayList list)

{

foreach (var item in list)

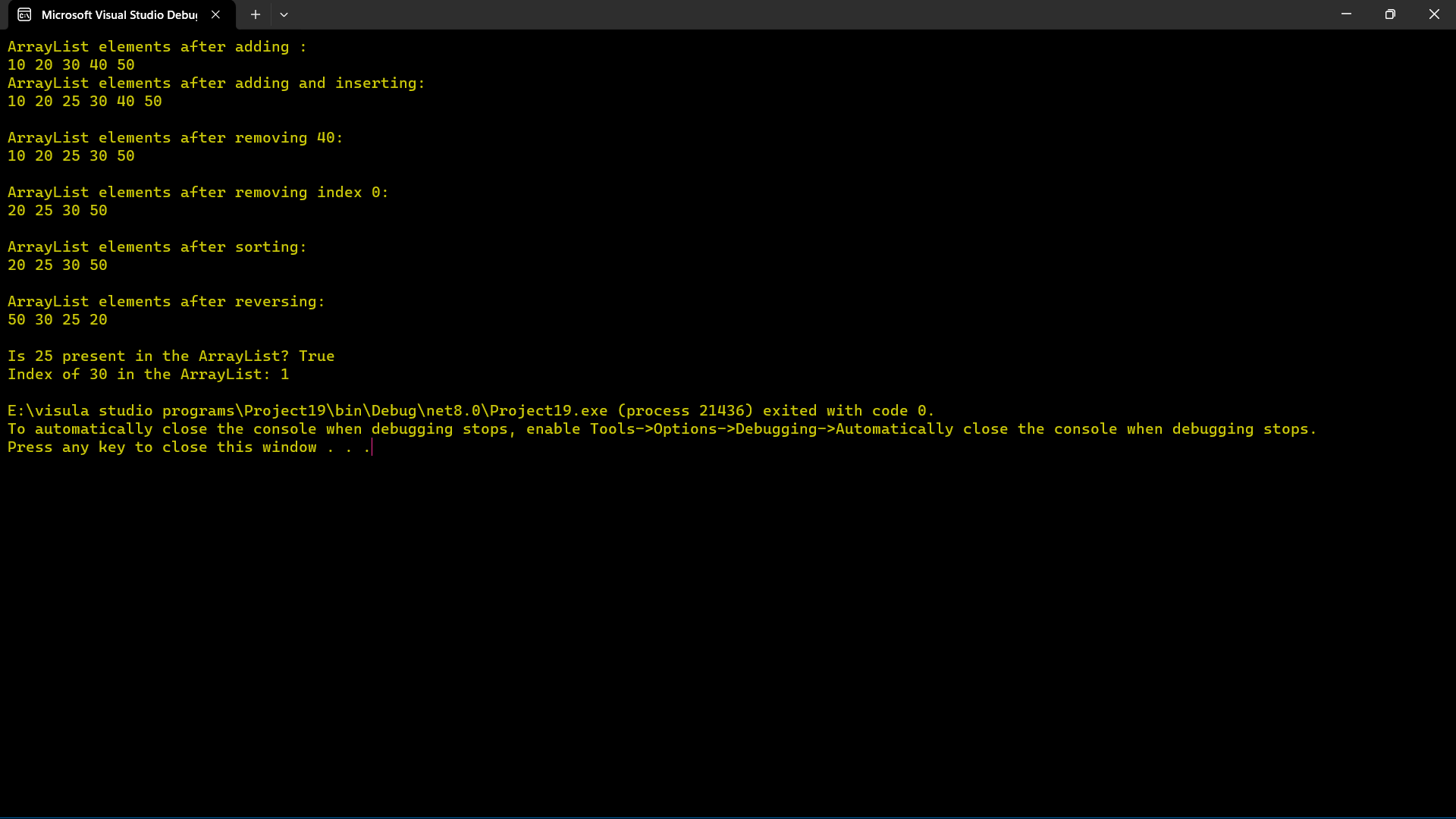
{

Console.Write(item + " ");

}

Console.WriteLine();

}

}

1. Implement c# program to demonstrate the use of the following methods of Hashtable Class.

Add(), Remove(), ContainsKey(), ContainsValue()

Ans - using System;

using System.Collections;

class Program

{

static void Main(string[] args)

{

Hashtable hashtable = new Hashtable();

hashtable.Add("John", 25);

hashtable.Add("Alice", 30);

hashtable.Add("Bob", 35);

hashtable.Add("Emma", 40);

Console.WriteLine("Hashtable elements:");

DisplayHashtable(hashtable);

hashtable.Remove("Alice");

Console.WriteLine("\nHashtable elements after removal of Alice:");

DisplayHashtable(hashtable);

Console.WriteLine("\nDoes the Hashtable contain the key 'Bob'? " + hashtable.ContainsKey("Bob"));

Console.WriteLine("Does the Hashtable contain the value 404? " + hashtable.ContainsValue(404));

}

static void DisplayHashtable(Hashtable hashtable)

{

foreach (DictionaryEntry entry in hashtable)

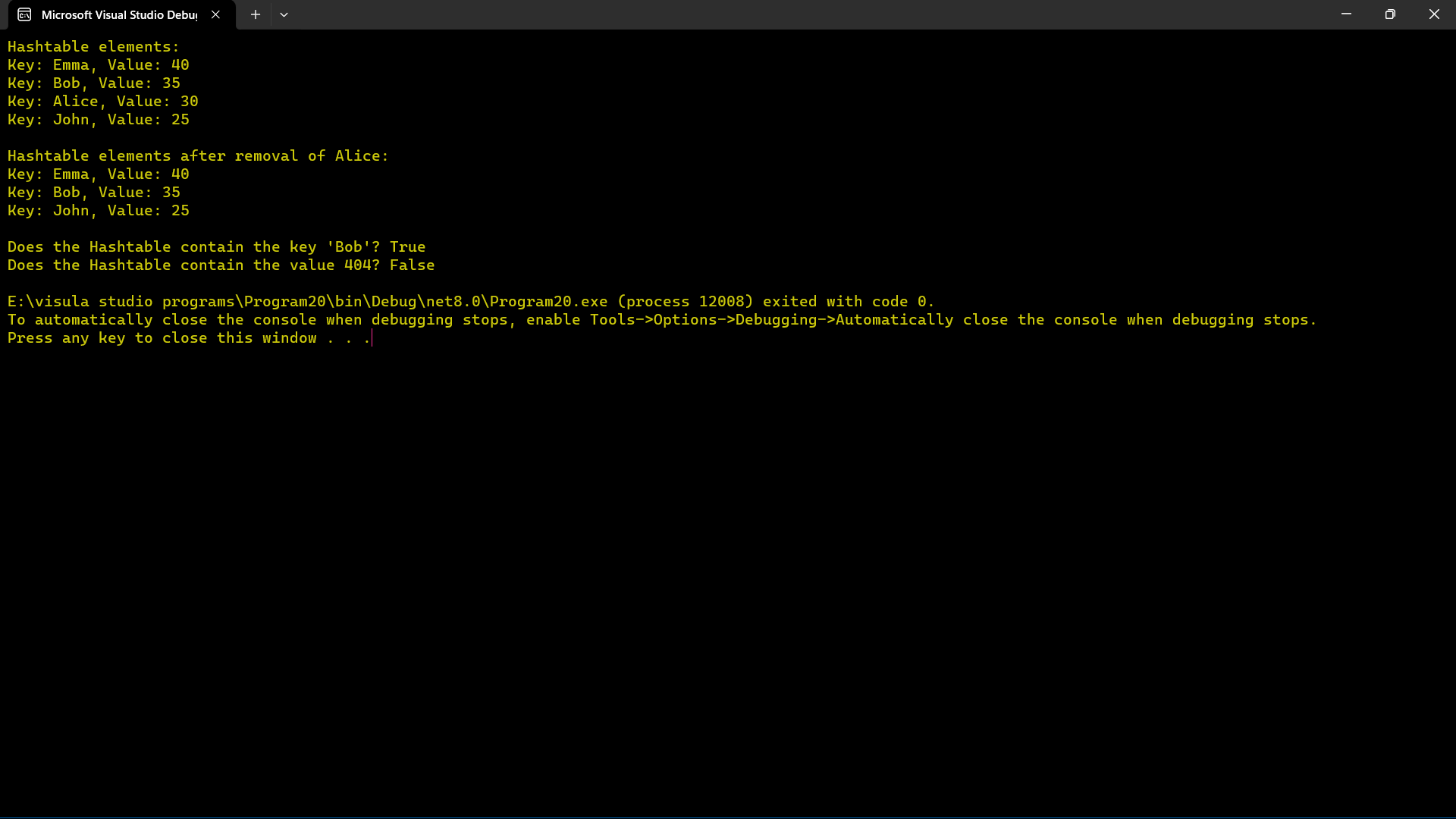
{

Console.WriteLine("Key: " + entry.Key + ", Value: " + entry.Value);

}

}

}



1. Implement c# program to demonstrate the use of Exception Handling. (try, catch, finally)

Ans - using System;

class TryCatch

{

public static void Main(string[] args)

{

try

{

Console.WriteLine("Enter a number");

int n1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter another number");

int n2 = int.Parse(Console.ReadLine());

int div = n1 / n2;

}

catch (DivideByZeroException ex)

{

Console.WriteLine("Error: " + ex.Message);

}

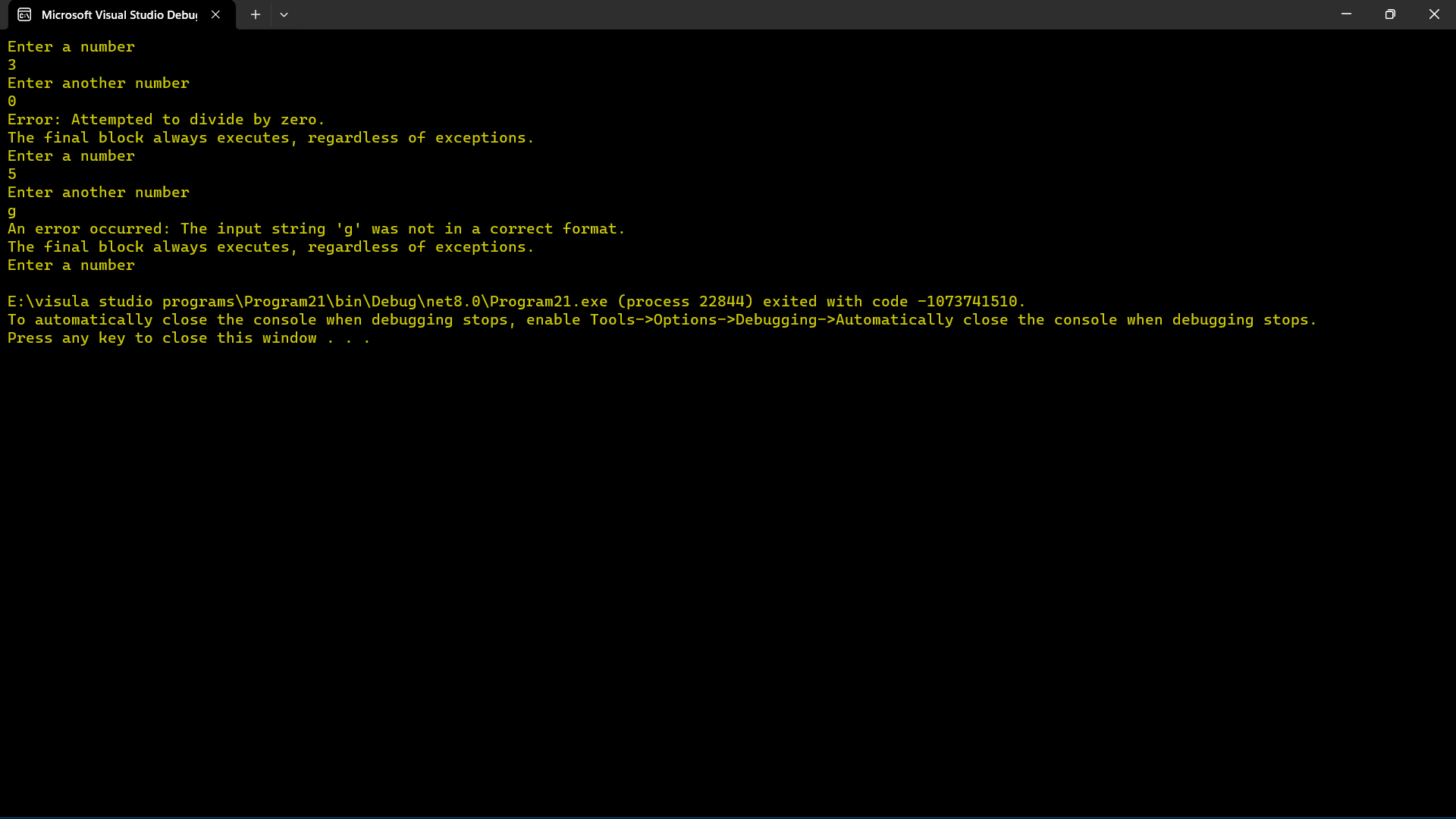
catch (Exception ex)

{

Console.WriteLine("An error occurred: " + ex.Message);

}

finally

 {Console.WriteLine("The final block always executes, regardless of exceptions.");}}}

1. Implement c# program to demonstrate the use of LINQ Query Syntax and LINQ Method Syntax.

Ans - using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

static void Main(string[] args)

{

List<string> fruits = new List<string> { "apple", "banana", "orange", "kiwi", "grape" };

var querySyntaxResult = from fruit in fruits

where fruit.Length > 5

orderby fruit

select fruit;

Console.WriteLine("LINQ Query Syntax:");

foreach (var fruit in querySyntaxResult)

{

Console.WriteLine(fruit);

}

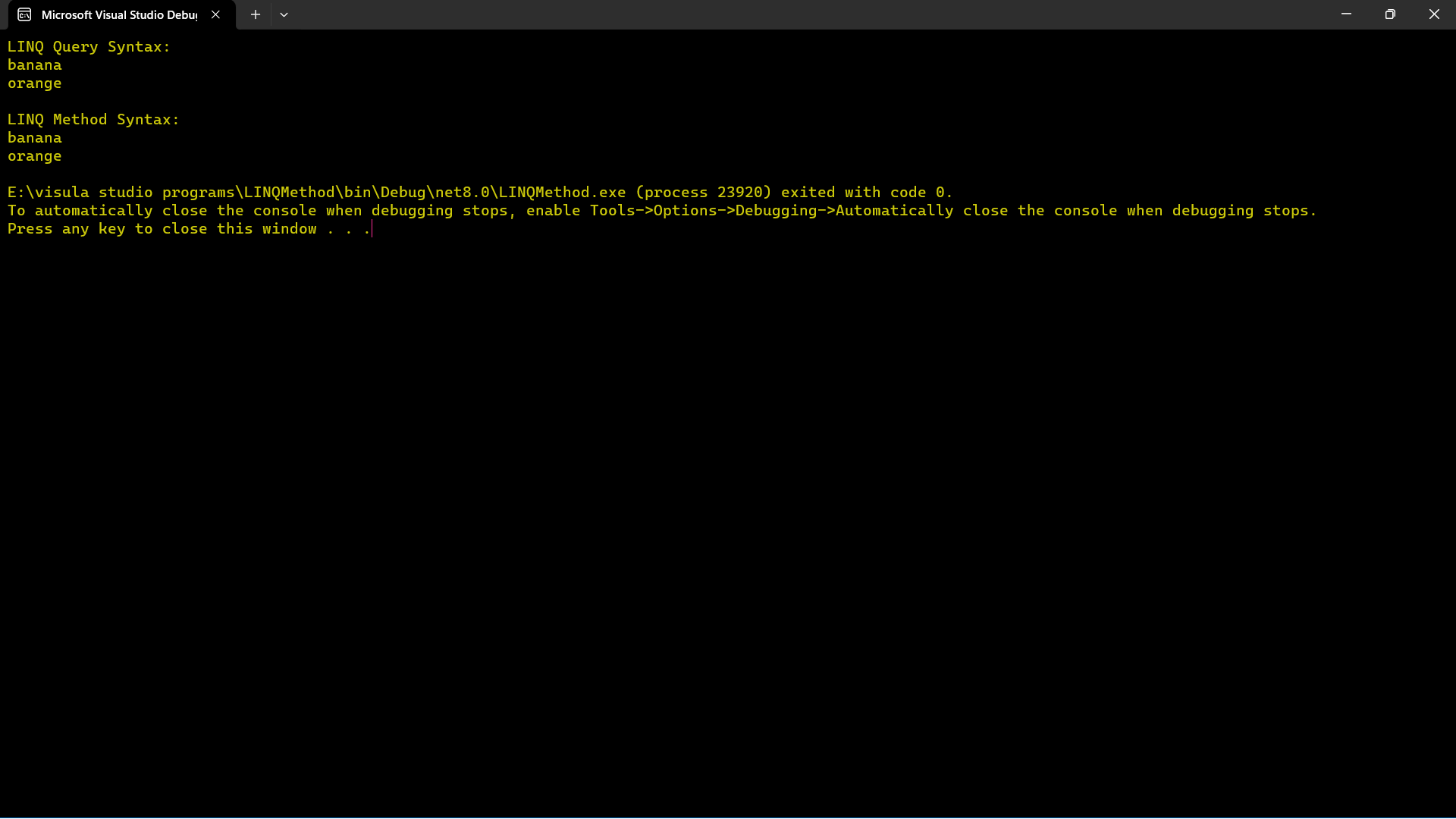
var methodSyntaxResult = fruits.Where(fruit => fruit.Length > 5).OrderBy(fruit => fruit);

Console.WriteLine("\nLINQ Method Syntax:");

foreach (var fruit in methodSyntaxResult)

{

Console.WriteLine(fruit);

 }}}

1. Implement c# program to demonstrate the use of LINQ Query Operators (select, where, OrderBy, Count, Max, Min, Sum, All, Any, Contains, etc.)

Ans - using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

static void Main(string[] args)

{

List<int> numbers = new List<int> { 10, 20, 30, 40, 50 };

var selectResult = numbers.Select(num => num \* 2);

Console.WriteLine("Select:");

foreach (var num in selectResult)

{Console.WriteLine(num);}

var whereResult = numbers.Where(num => num > 20);

Console.WriteLine("\nWhere:");

foreach (var num in whereResult)

{Console.WriteLine(num);}

var orderByResult = numbers.OrderBy(num => num);

Console.WriteLine("\nOrderBy:");

foreach (var num in orderByResult)

{Console.WriteLine(num); }

var countResult = numbers.Count();

Console.WriteLine("\nCount: " + countResult);

var maxResult = numbers.Max();

Console.WriteLine("Max: " + maxResult);

var minResult = numbers.Min();

Console.WriteLine("Min: " + minResult);

var sumResult = numbers.Sum();

Console.WriteLine("Sum: " + sumResult);

var allResult = numbers.All(num => num < 100);

Console.WriteLine("All numbers are less than 100: " + allResult);

var anyResult = numbers.Any(num => num % 2 == 0);

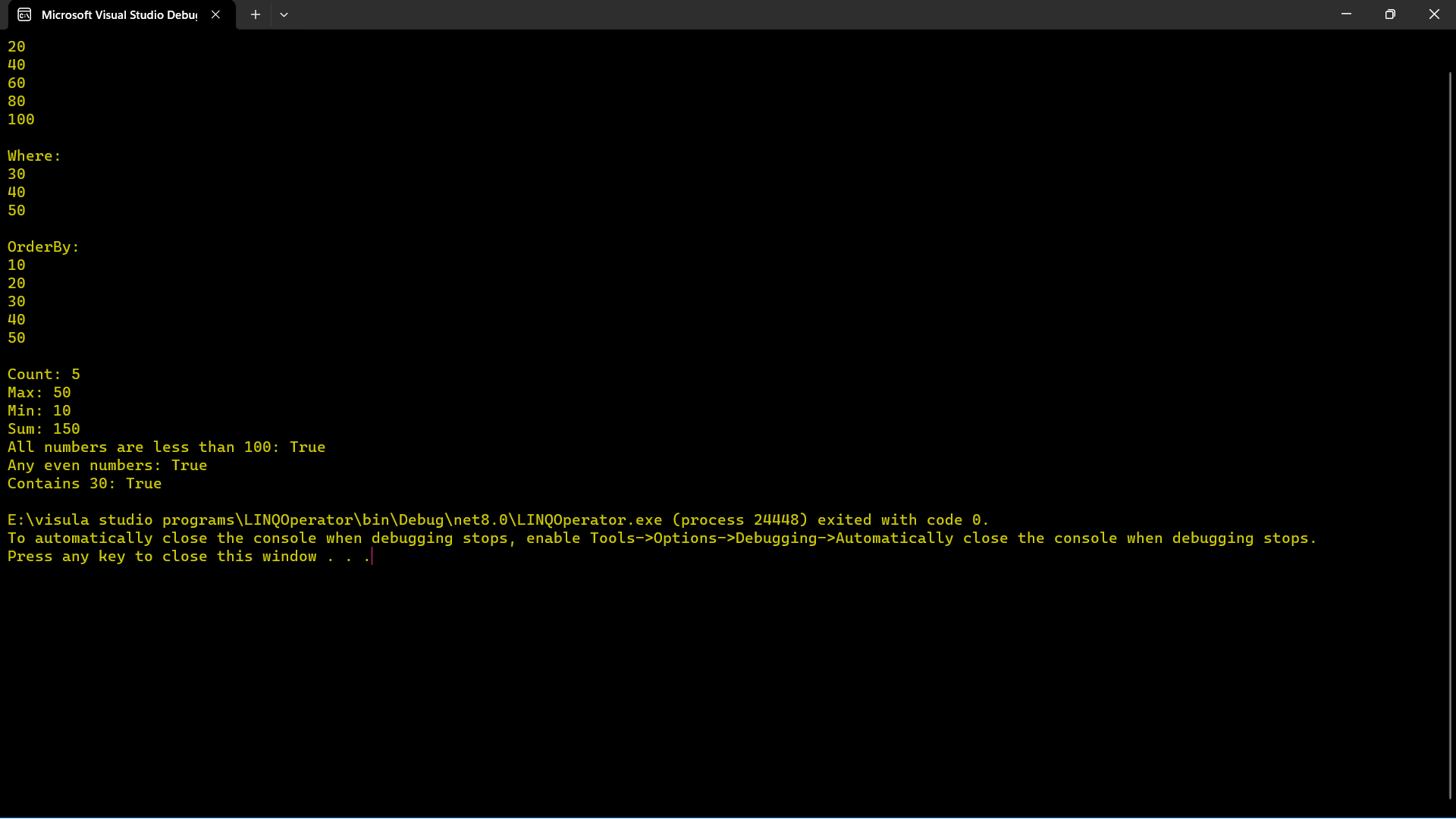
Console.WriteLine("Any even numbers: " + anyResult);

var containsResult = numbers.Contains(30);

Console.WriteLine("Contains 30: " + containsResult);

}

}



1. Develop ASP.NET Core MVC web application to demonstrate the use of ViewData, Viewbag and TempData.

Ans - index.cshtml

@{

ViewData["Title"] = "Home Page";

}

<h1>@ViewData["Sandesh"]</h1>

<h2>@ViewBag.Message</h2>

<h3>@TempData["Message"]</h3>

HomeController.cs

using Microsoft.AspNetCore.Mvc;

using Program24.Models;

using System.Diagnostics;

namespace Program24.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

ViewData["Sandesh"] = "This is a message from ViewData";

ViewBag.Message = "This is a message from ViewBag";

TempData["Message"] = "This is a message from TempData";

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

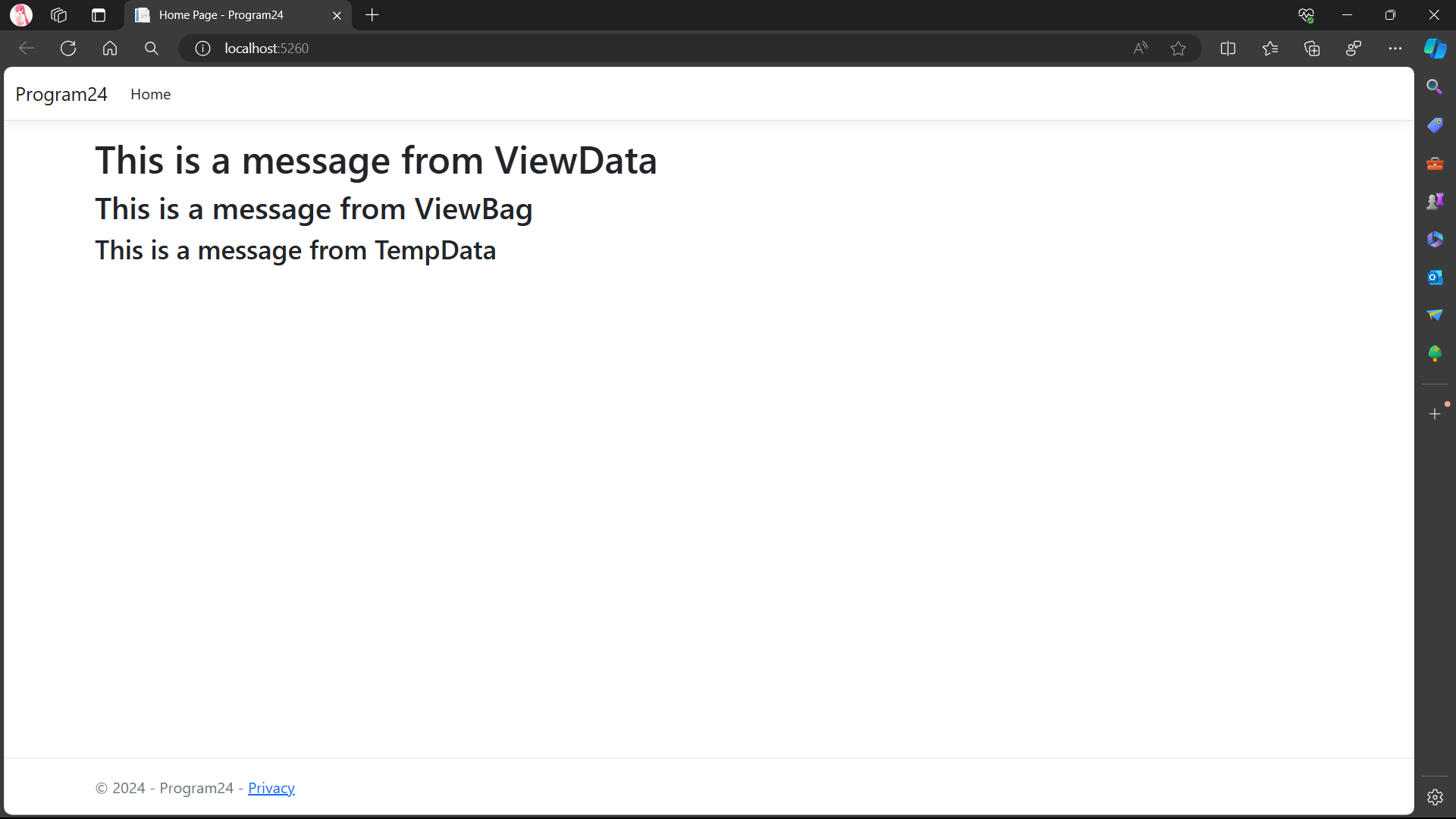
public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

1. Develop ASP.NET Core MVC web application to Create a Layout View in ASP.NET Core MVC.

Ans – index.cshtml

@{

ViewBag.Title = "Home Page";

Layout = "~/Views/Shared/\_Layout1.cshtml";

}

<center><h1 style="font-size: 60px;">LOGIN</h1></center>

Layout.cshtml

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login Page</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

width: 300px;

background-color: #fff;

border-radius: 5px;

padding: 20px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

h2 {

text-align: center;

}

input[type="text"],

input[type="password"],

input[type="submit"] {

width: 100%;

padding: 10px;

margin-top: 10px;

margin-bottom: 20px;

border: 1px solid #ccc;

border-radius: 3px;

box-sizing: border-box;

}

input[type="submit"] {

background-color: #4caf50;

color: white;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #45a049;

}

.error-message {

color: red;

font-size: 14px;

margin-top: 5px;

}

</style>

</head>

<body>

<div class="container">

@RenderBody()

<form id="loginForm">

<input type="text" id="username" placeholder="Username">

<input type="password" id="password" placeholder="Password">

<input type="submit" value="Login">

<div class="error-message" id="errorMessage"></div>

</form>

</div>

<script>

document.getElementById("loginForm").addEventListener("submit", function(event) {

event.preventDefault();

var username = document.getElementById("username").value;

var password = document.getElementById("password").value;

if (username === "" || password === "") {

document.getElementById("errorMessage").innerText = "Please enter both username and password.";

return;

}

if (username === "user" && password === "password") {

alert("Login successful!");

} else {

document.getElementById("errorMessage").innerText = "Invalid username or password.";

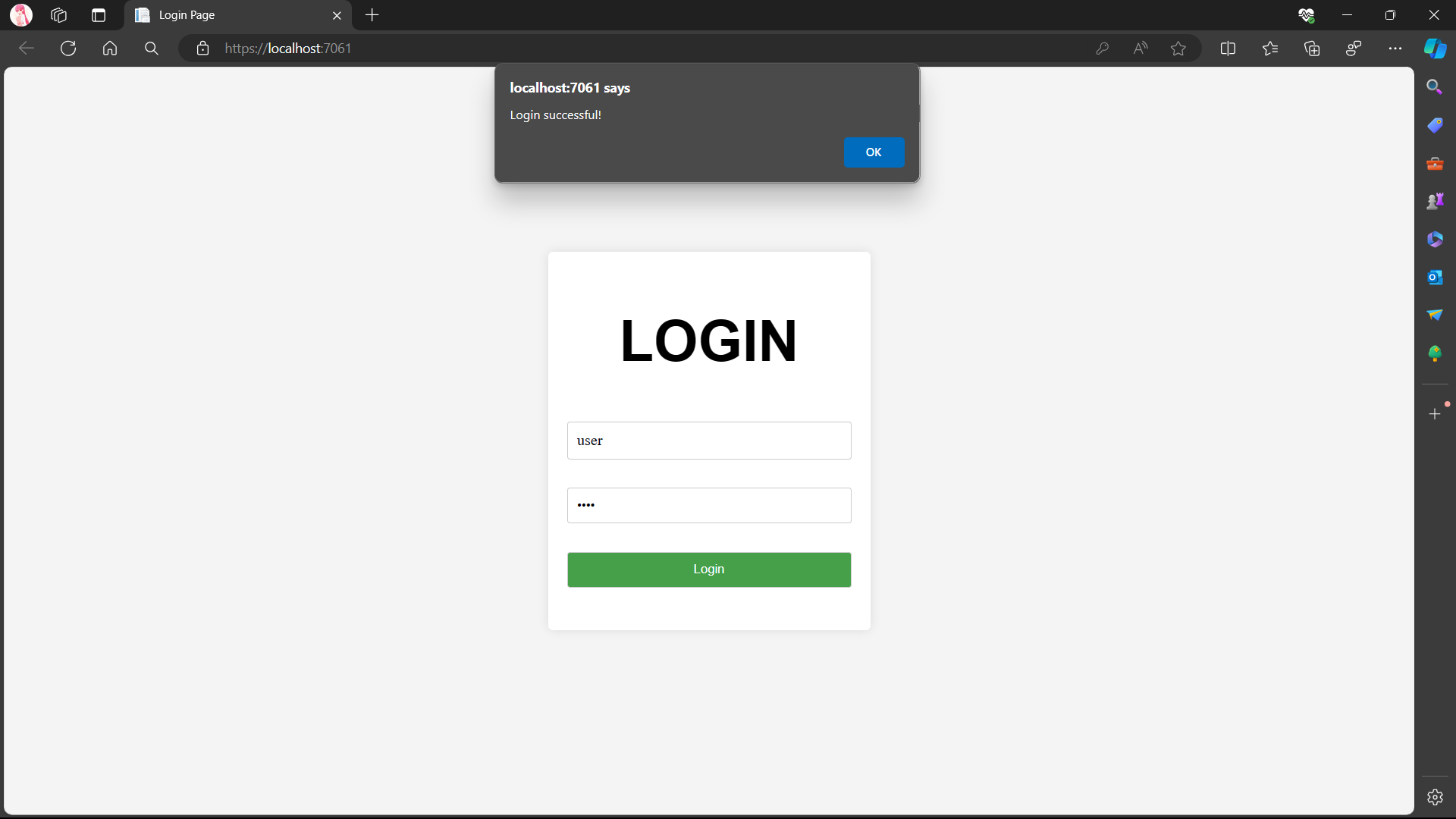
}

});

</script>

</body>

</html>



1. Develop ASP.NET Core MVC web application to demonstrate the use of Razor Syntax (If …else, Switch, For Loop, For each Loop, While Loop)

Ans – index.cshtml

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewData["Title"]="Program 26"</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

color: #333;

background-color: #f9f9f9;

}

section {

padding: 25px 0;

text-align: center;

background-color: #fff;

border-bottom: 1px solid #ddd;

}

h1 {

font-size: 25px;

margin: 0;

}

ul {

list-style: none;

padding: 0;

}

li {

margin-bottom: 5px;

}

p {

margin: 0;

}

. special-day {

color: #e74c3c;

font-weight: bold;

}

</style>

</head>

<body>

<section style="margin-top: 10px;">

<div class="container">

<center><h1>IF...ELSE</h1></center>

<h1>-------------------</h1>

@{

int num1 = 10, num2 = 3;

String num3;

if (num1 % num2 == 0)

{

num3 = "The number is Even!";

}

else

{

num3 = "The number is Odd!";

}

}

<h1>The difference between <b>@num1</b> and <b>@num2</b> is <b>@num3</b></h1>

</div>

</section>

<section>

<div class="container">

<center><h1>FOR LOOP</h1></center>

<h1>-------------------</h1>

<ul>

@for (int i = 0; i <= 2; i++)

{

<li><h1>@i</h1></li>

}

</ul>

</div>

</section>

<section>

<div class="container">

<center><h1>WHILE LOOP</h1></center>

<h1>-------------------</h1>

<ul>

@{

var n = 0;

while (n < 2)

{

n += 1;

<p><h1>Line @n</h1></p>

}

}

</ul>

</div>

</section>

<section>

<div class="container">

<center><h1>FOREACH</h1></center>

<h1>-------------------</h1>

@{

string[] members = { "Apple", "Banana","Grapes"};

foreach (var person in members)

{

<p><h1>@person</h1></p>

}

}

</div>

</section>

<section>

<div class="container">

<center><h1>SWITCH CASE</h1></center>

<h1>-------------------</h1>

@switch (DateTime.Now.DayOfWeek)

{

case DayOfWeek.Monday:

<span class="special-day">Uh-oh...</span>

break;

case DayOfWeek.Friday:

<span class="special-day">Weekend coming up!</span>

break;

case DayOfWeek.Saturday:

<span class="special-day">Saturday AA Gaya!</span>

break;

case DayOfWeek.Sunday:

<span class="special-day">Finally weekend!</span>

break;

default:

<h1><span>Nothing special about this day...</span></h1>

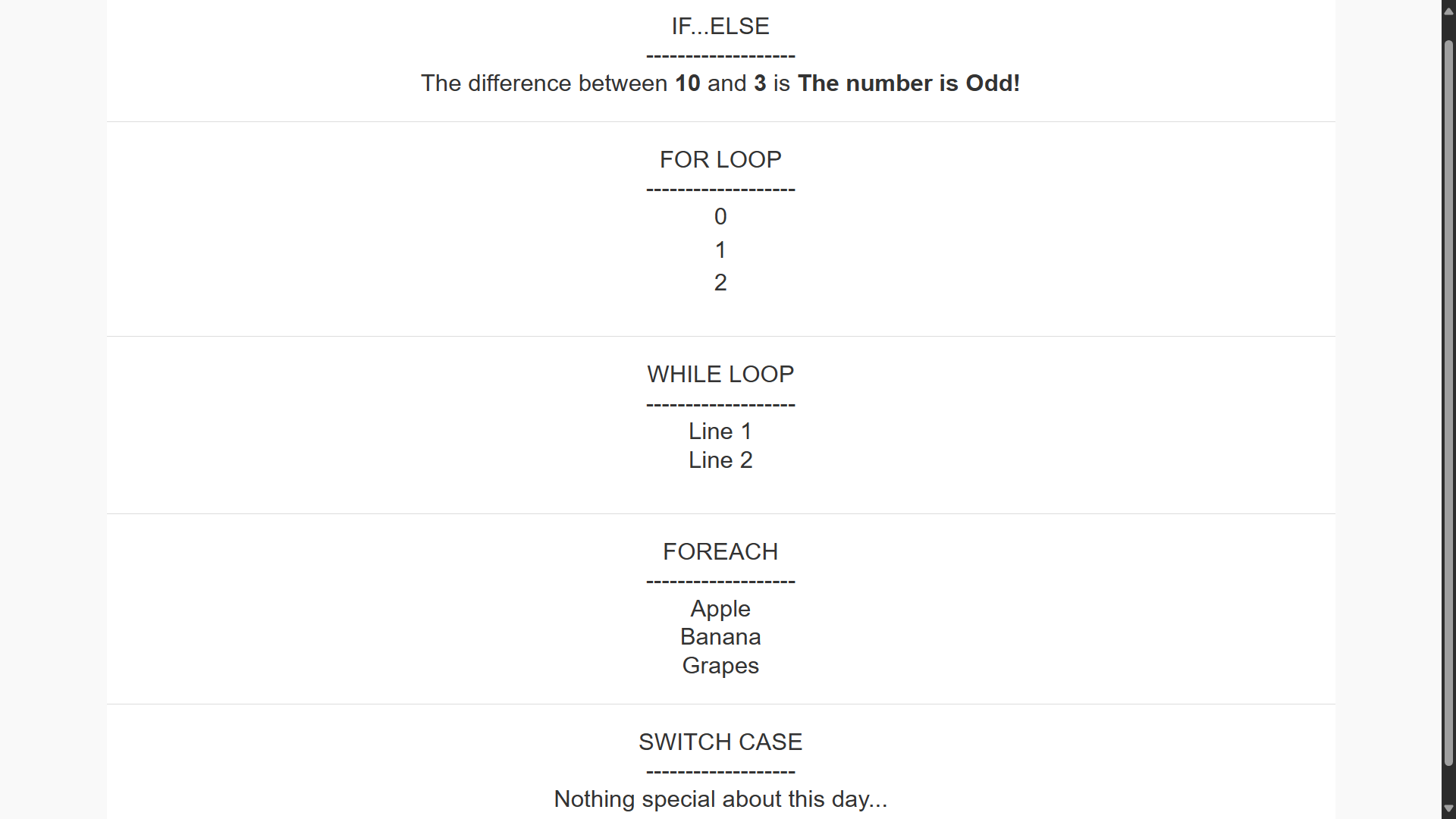
break;

}

</div>

</section>

</body>

</html>

1. Create ASP.NET Core MVC registration page using Razor Syntax and Html Helper methods to register new student.

First name (Textbox)

Last name or initial (Textbox)

Gender (Radio Button)

Age (Take numeric input only)

Existing Qualification (Dropdown List)

Email address (Regular Expression)

Choose a password (min. 8 characters)

Ans – HomeController.cs

using Microsoft.AspNetCore.Mvc;

using Register. Models;

using System. Diagnostics;

namespace Register. Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

public IActionResult Index(Registration registration)

{

return View("Congratulation", registration);

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

} }}

Index.cshtml

@model Register.Models.Registration

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<title>RsvpForm</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f5f5f5;

margin: 0;

padding: 0;

box-sizing: border-box;

}

.container {

width: 90%; /\* Relative width \*/

max-width: 600px;

margin: 4% auto;

padding: 20px;

background-color: #fff;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

input[type="text"],

input[type="email"],

input[type="password"],

select,

input[type="number"] {

width: calc(100% - 22px); /\* Adjusted width \*/

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

box-sizing: border-box;

font-size: 16px;

}

input[type="submit"] {

width: 100%; /\* Relative width \*/

padding: 10px;

border: none;

border-radius: 5px;

background-color: #4CAF50;

color: #fff;

font-size: 16px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #45a049;

}

p {

margin-bottom: 10px;

}

</style>

<script>

function validateForm() {

var firstName = document.getElementById("FirstName").value;

var lastName = document.getElementById("LastName").value;

var gender = document.querySelector('input[name="Gender"]:checked');

var age = document.getElementById("Age").value;

var qualification = document.getElementById("Qualification").value;

var email = document.getElementById("Email").value;

var password = document.getElementById("Password").value;

if (firstName.trim() == "") {

alert("Enter your first name");

return false;

}

if (lastName.trim() == "") {

alert("Enter your last name");

return false;

}

if (!gender) {

alert("Select your gender");

return false;

}

if (age.trim() == "") {

alert("Enter your age");

return false;

}

if (qualification.trim() == "") {

alert("Enter your qualification");

return false;

}

if (email.trim() == "") {

alert("Enter your email");

return false;

}

if (password.trim() == "") {

alert("Enter your password");

return false;

}

return true;

}

</script>

</head>

<body>

<div class="container">

<center><h1>Registration Form</h1></center>

@using (Html.BeginForm("Index", "Home", FormMethod.Post, new { onsubmit = "return validateForm();" }))

{

<p>

Your First Name : @Html.TextBoxFor(x => x.FirstName, new { id = "FirstName" })

</p>

<p>

Your Last Name: @Html.TextBoxFor(x => x.LastName, new { id = "LastName" })

</p>

<p>

Your Gender:

@Html.RadioButtonFor(x => x.Gender, "Male", new { id = "GenderMale" }) Male

@Html.RadioButtonFor(x => x.Gender, "Female", new { id = "GenderFemale" }) Female

</p>

<p>

Your Age: @Html.TextBoxFor(x => x.Age, new { id = "Age", type = "number", min = "11", max = "24" })

</p>

<p>

Your Qualification: @Html.DropDownListFor(x => x.Qualification, new SelectList(new List<string> { "High School", "College", "Bachelor's Degree", "Master's Degree", "PhD" }), "Select", new { id = "Qualification" })

</p>

<p>

Your email: @Html.TextBoxFor(x => x.Email, new { id = "Email", type = "email" })

</p>

<p>

Your password: @Html.PasswordFor(x => x.Password, new { id = "Password", minlength = "3", maxlength = "8" })

</p>

<input type="submit" value="Register Me" />

}

</div>

</body>

</html>

Registration.cs

using System.ComponentModel.DataAnnotations;

namespace Register.Models

{

public class Registration

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string Gender { get; set; }

public int Age { get; set; }

public string Qualification { get; set; }

public string Email { get; set; }

public string Password { get; set; }

}}

Congradulation.cshtml

@model Register.Models.Registration

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Congratulations</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

}

.container {

max-width: 600px;

margin: 20px auto;

padding: 20px;

background-color: #fff;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

h1, h2 {

text-align: center;

}

h1 {

color: #333;

}

h2 {

color: #666;

}

.details {

margin-top: 20px;

}

.details p {

margin: 10px 0;

}

</style>

</head>

<body>

<div class="container">

<h1>Congratulations, @Model.FirstName!</h1>

<h2>Your Registration has been Successfully completed.</h2>

<div class="details">

<p><strong>Your full name:</strong> @Model.FirstName @Model.LastName</p>

<p><strong>Your age:</strong> @Model.Age</p>

<p><strong>Your qualification:</strong> @Model.Qualification</p>

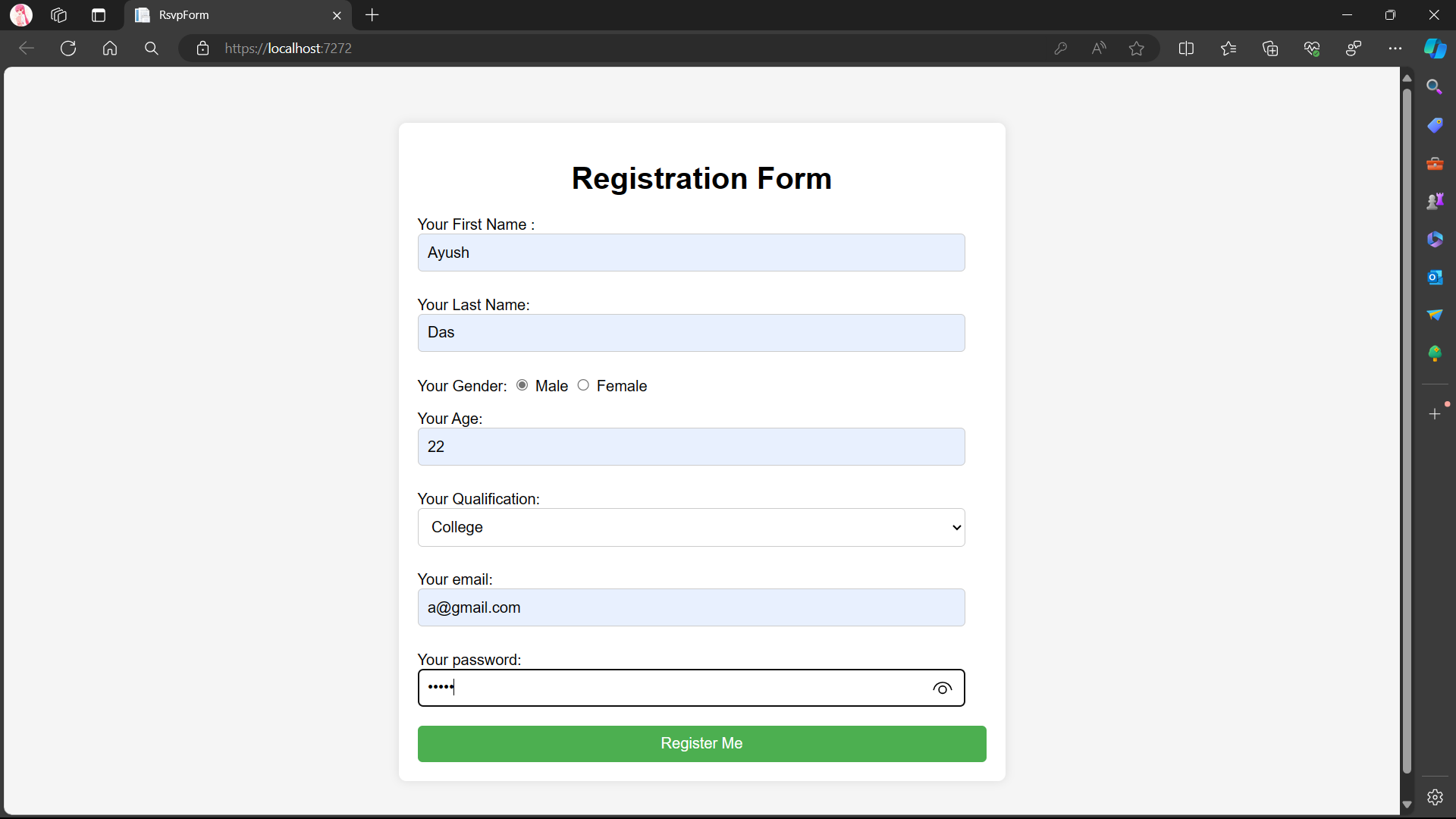
<p><strong>Your email:</strong> @Model.Email</p>

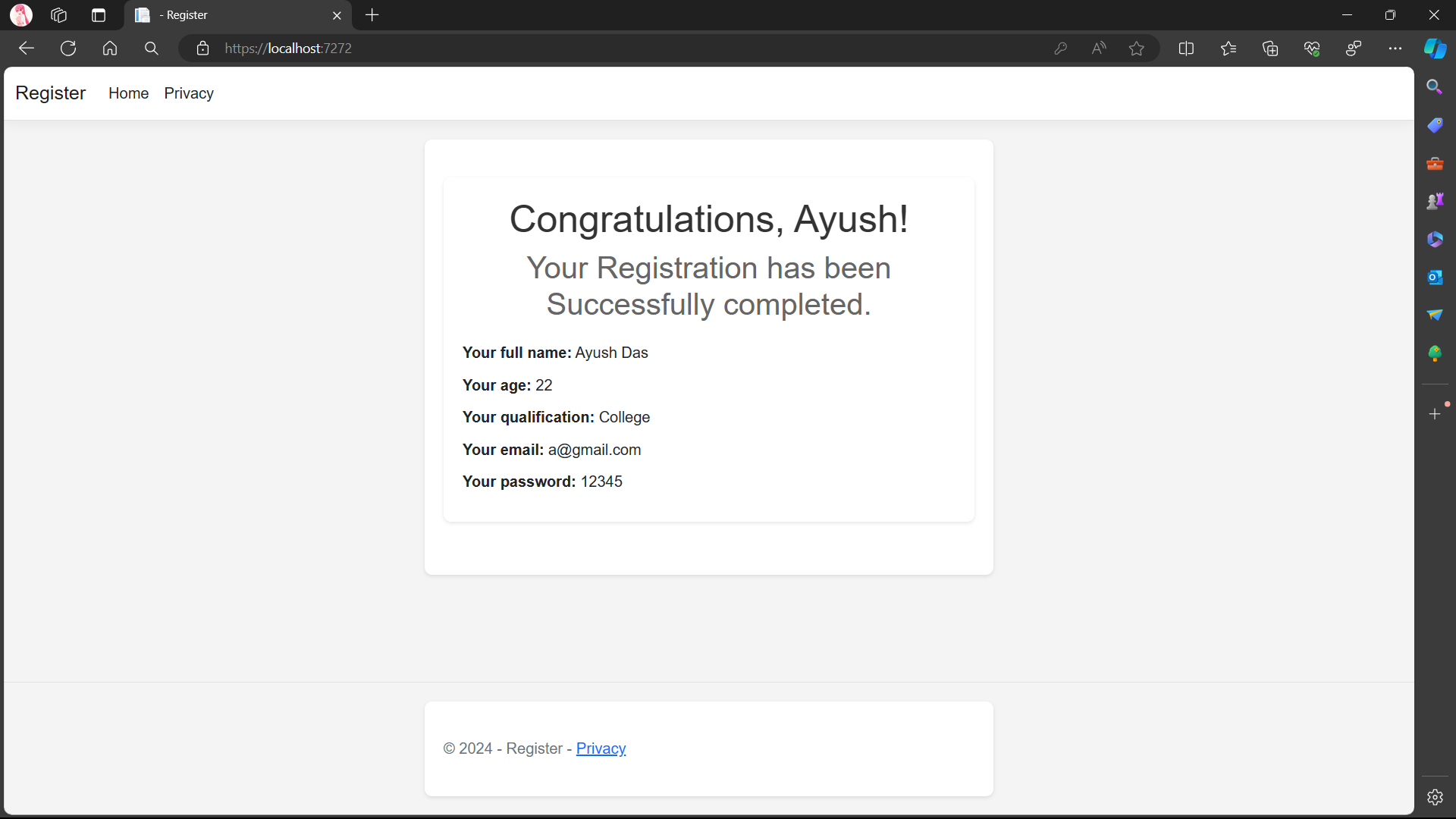
<p><strong>Your password:</strong> @Model.Password</p>

</div>

</div>

</body>

</html>



1. Develop ASP.NET core MVC web application to perform CRUD operation by using entity framework with validation on Book model.

Book (BookId, BookTitle, AuthorName, Publication and Price)

Ans - Book.cs

using System.ComponentModel.DataAnnotations;

namespace Program28.Models

{

public class Book

{

[Key]

public int BookId { get; set; }

[Required(ErrorMessage = "Title is required")]

public string BookTitle { get; set; }

[Required(ErrorMessage = "Author name is required")]

public string AuthorName { get; set; }

[Required(ErrorMessage = "Publication is required")]

public string Publication { get; set; }

[Required(ErrorMessage = "Price is required")]

[Range(0, double.MaxValue, ErrorMessage = "Price must be a positive number")]

public decimal Price { get; set; }

}}

Program28Context.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using Program28.Models;

namespace Program28.Data

{

public class Program28Context : DbContext

{

public Program28Context (DbContextOptions<Program28Context> options)

: base(options)

{

}

public DbSet<Program28.Models.Book> Book { get; set; } = default!;

}}

InitialMigration.cs

using Microsoft.EntityFrameworkCore.Migrations;

namespace Program28.Migrations

{

public partial class InitialMigration : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Book",

columns: table => new

{

BookId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

BookTitle = table.Column<string>(type: "nvarchar(max)", nullable: false),

AuthorName = table.Column<string>(type: "nvarchar(max)", nullable: false),

Publication = table.Column<string>(type: "nvarchar(max)", nullable: false),

Price = table.Column<decimal>(type: "decimal(18,2)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Book", x => x.BookId);

});

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Book");

} }}

BookController.cs

public IActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("BookId,BookTitle,AuthorName,Publication,Price")] Book book)

{

if (ModelState.IsValid)

{

\_context.Add(book);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(book);

}

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var book = await \_context.Book.FindAsync(id);

if (book == null)

{

return NotFound();

}

return View(book);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("BookId,BookTitle,AuthorName,Publication,Price")] Book book)

{

if (id != book.BookId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(book);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!BookExists(book.BookId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(book);

}

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var book = await \_context.Book

.FirstOrDefaultAsync(m => m.BookId == id);

if (book == null)

{

return NotFound();

}

return View(book);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var book = await \_context.Book.FindAsync(id);

if (book != null)

{

\_context.Book.Remove(book);

}

await \_context.SaveChangesAsync();

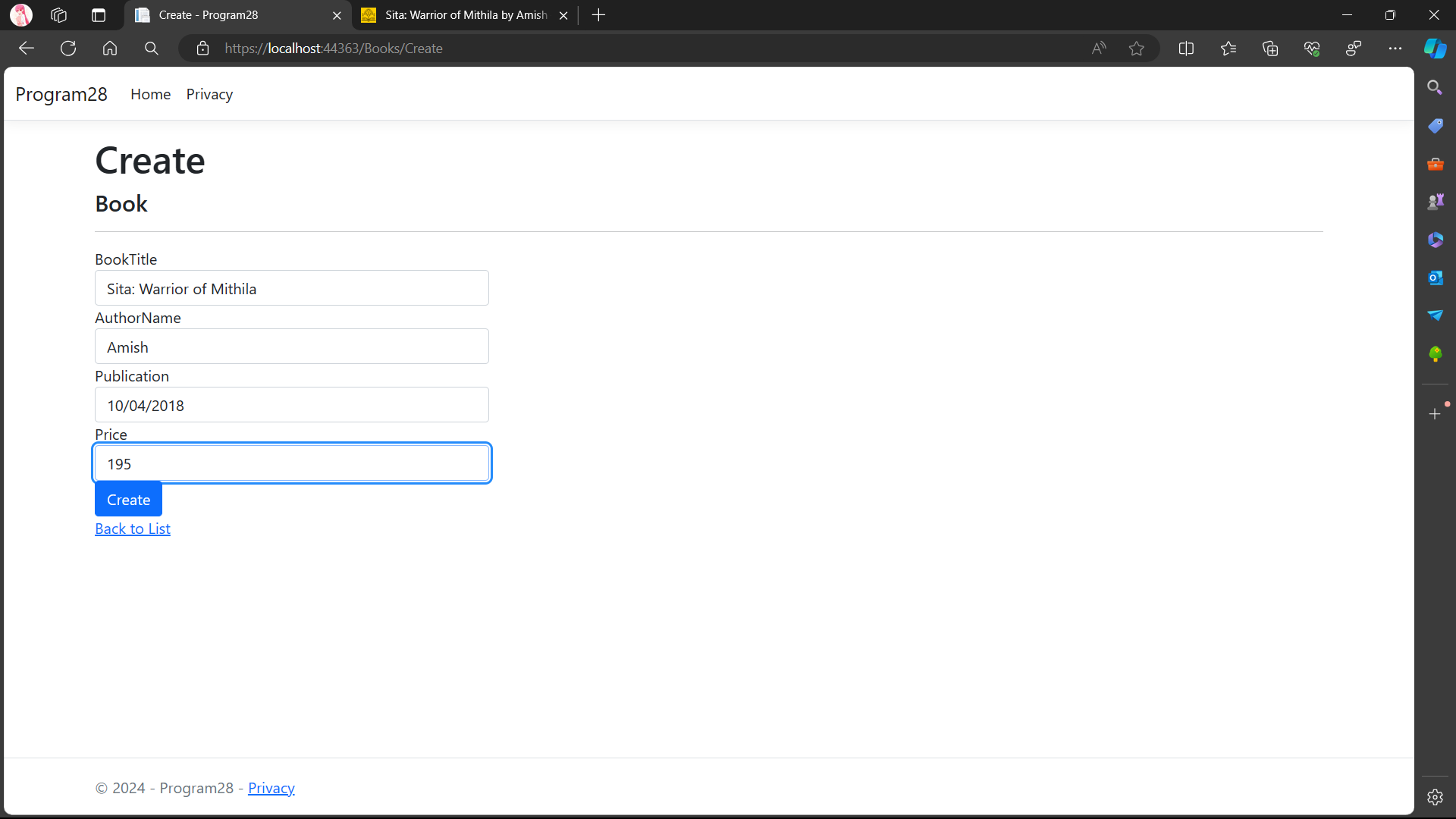
return RedirectToAction(nameof(Index));

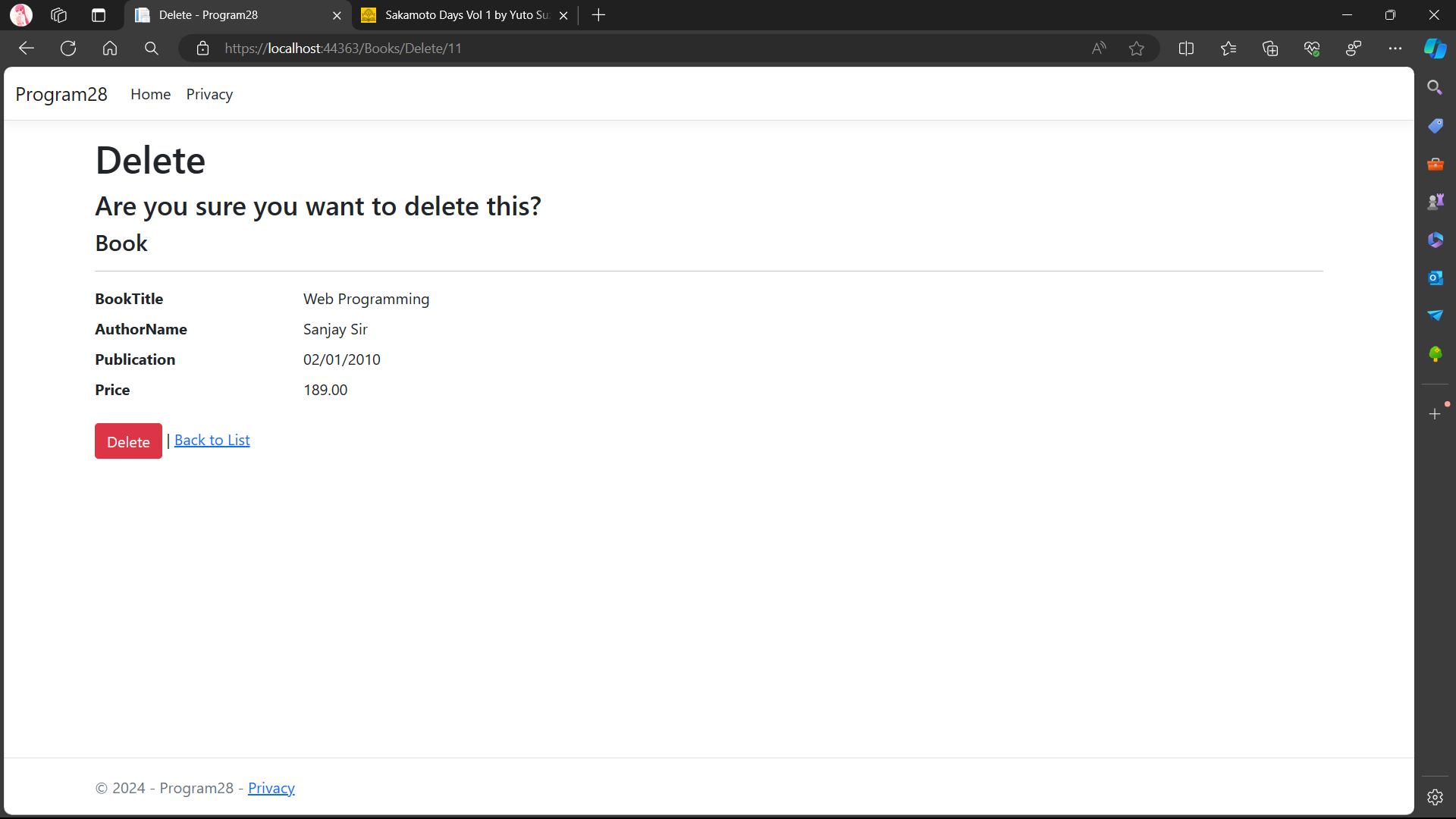
}

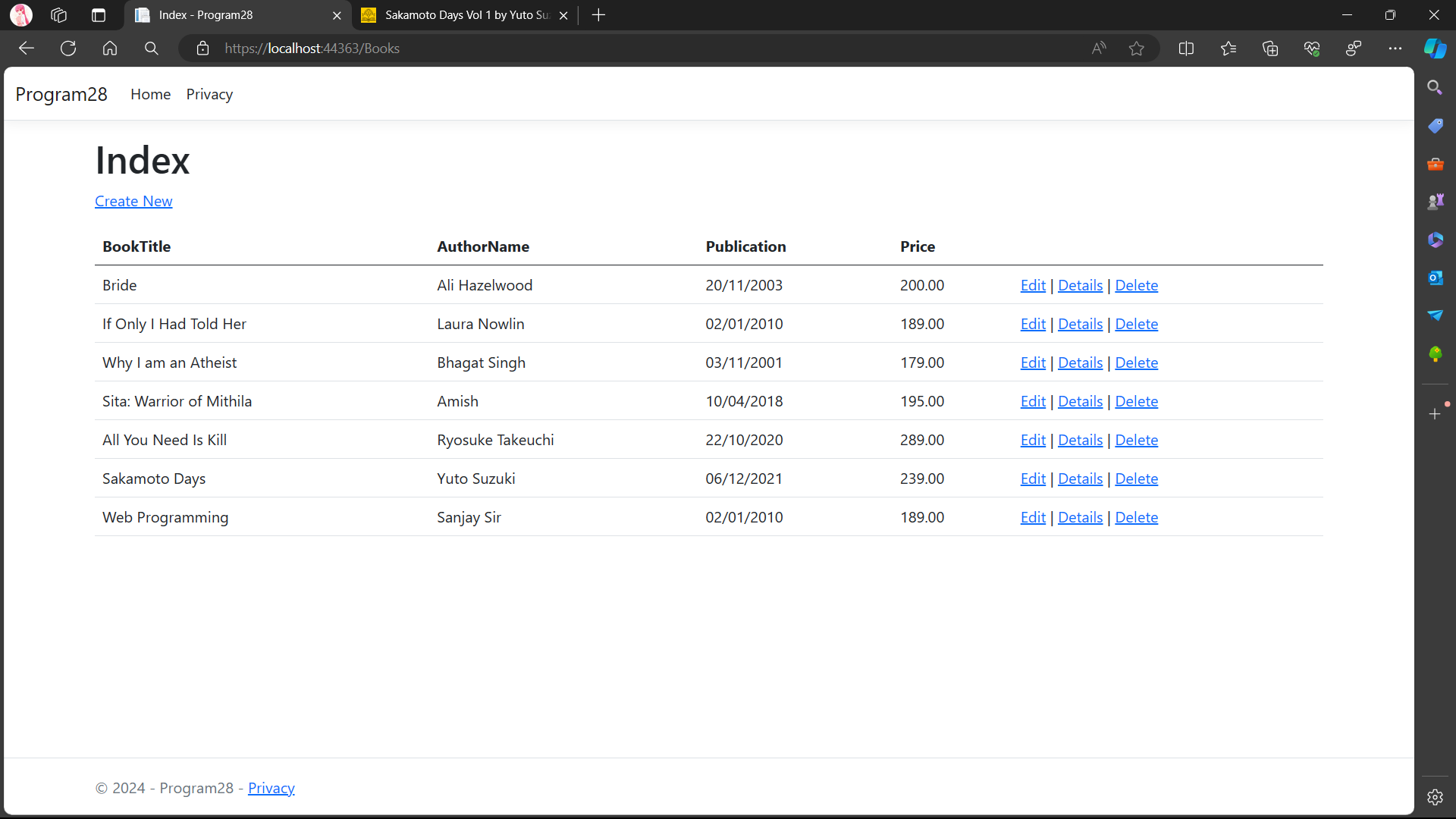
private bool BookExists(int id)

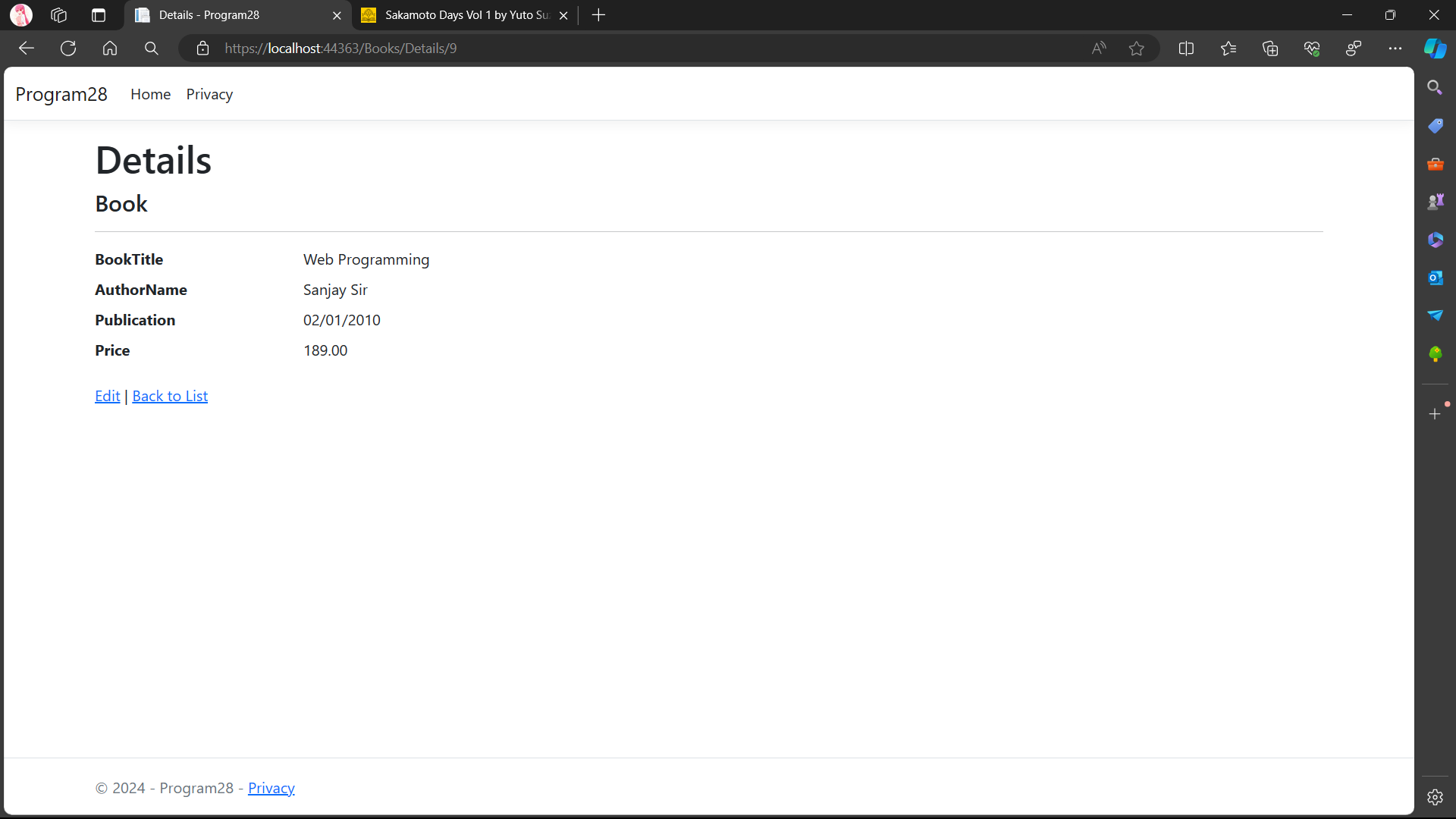
{

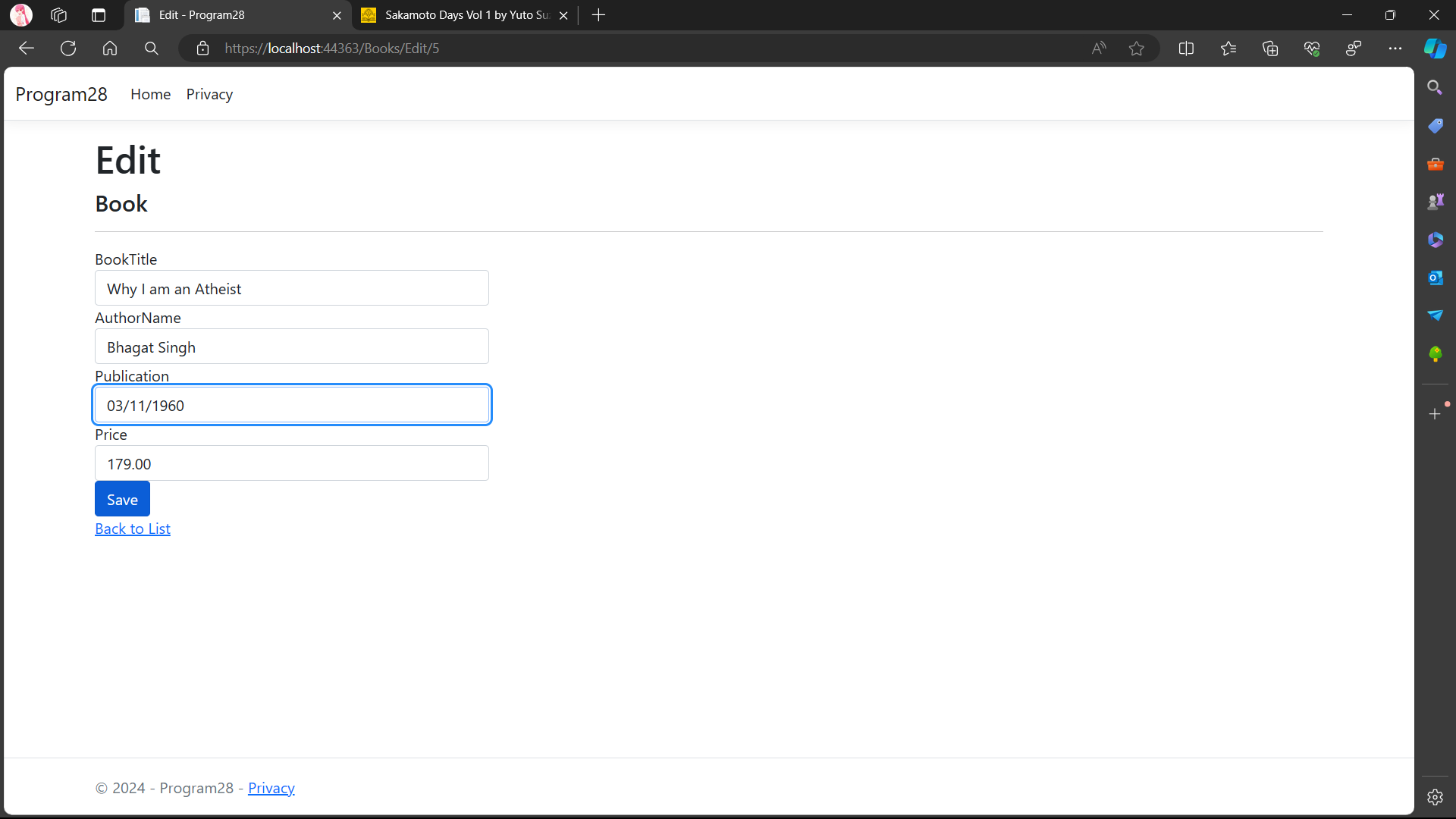
return \_context.Book.Any(e => e.BookId == id);

 } }}









1. Develop ASP.NET Core MVC application to perform CRUD operation by using entity framework with validation on Student, Course and Enrollment models.

Ans - Student.cs

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace CRUD.Models

{

public class Student

{

[Key]

public int StudentID { get; set; }

[Required(ErrorMessage = "First Name is required")]

[StringLength(50)]

public string FirstName { get; set; }

[Required(ErrorMessage = "Last Name is required")]

[StringLength(50)]

public string LastMidName { get; set; }

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

public DateTime EnrollmentDate { get; set; }

public ICollection<Enrollment> Enrollments { get; set; }

public Student()

{

Enrollments = new List<Enrollment>();

}

}

}

Course.cs

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace CRUD.Models

{

public class Course

{

[Key]

public int CourseID { get; set; }

[Required(ErrorMessage = "Course Title is required")]

[StringLength(100, ErrorMessage = "Title cannot exceed 100 characters")]

public string CourseTitle { get; set; }

[Required(ErrorMessage = "Credits are required")]

[Range(1, int.MaxValue, ErrorMessage = "Credits must be a positive number")]

public int CourseCredits { get; set; }

public ICollection<Enrollment> Enrollments { get; set; }

public Course()

{

Enrollments = new List<Enrollment>();

} }}

Enrollment.cs

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace CRUD.Models

{

public class Enrollment

{

[Key]

public int EnrollmentID { get; set; }

[Required(ErrorMessage = "CourseID is required")]

public int CourseID { get; set; }

[Required(ErrorMessage = "StudentID is required")]

public int StudentID { get; set; }

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

public DateTime EnrollmentDate { get; set; }

[ForeignKey("CourseID")]

public Course? Course { get; set; }

[ForeignKey("StudentID")]

public Student? Student { get; set; }

}}

