Name :- Jagruti Yogesh Patil

Roll No :- 57

Semester :- 7th

Subject :- .Net Advance using C#

Assignment No :- 1st Assignment

Start Date :- 10-9-2024

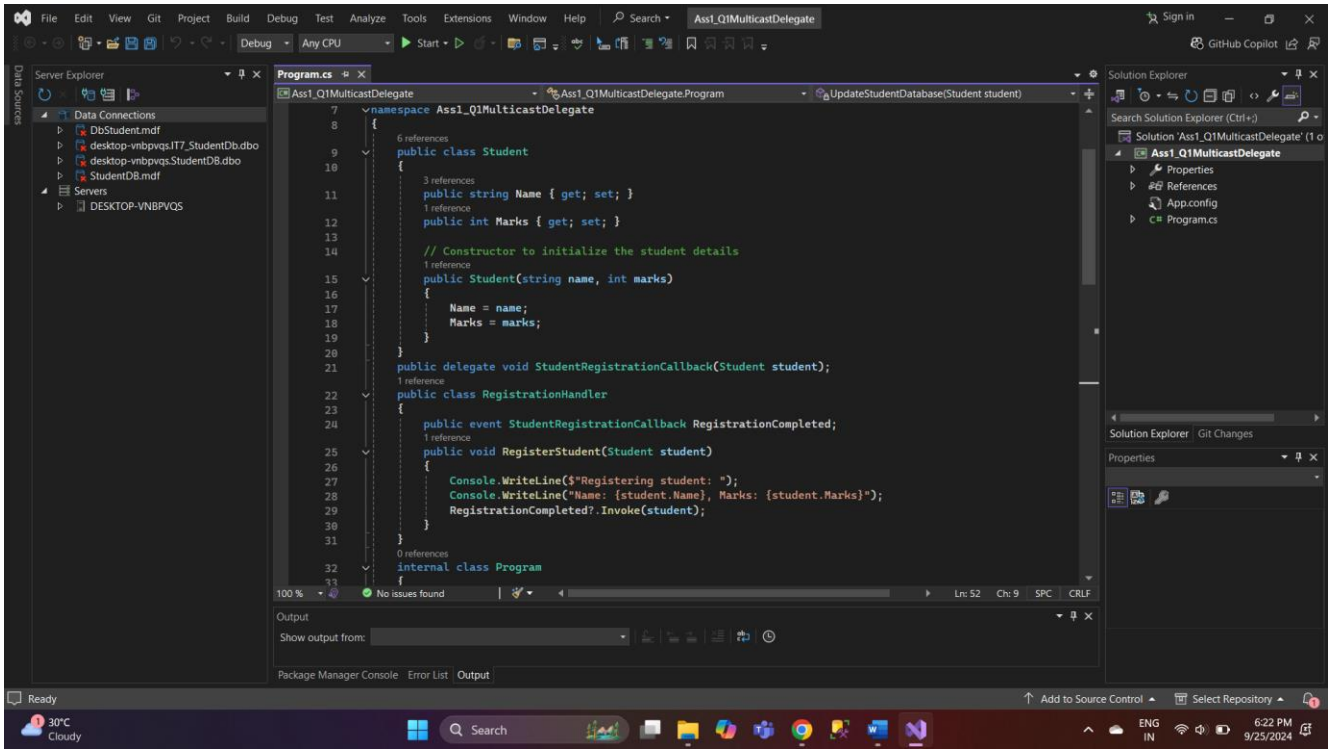
Q.1 Create a C# Application for student class using multicast delegate to demonstrate callback function usage for an application.

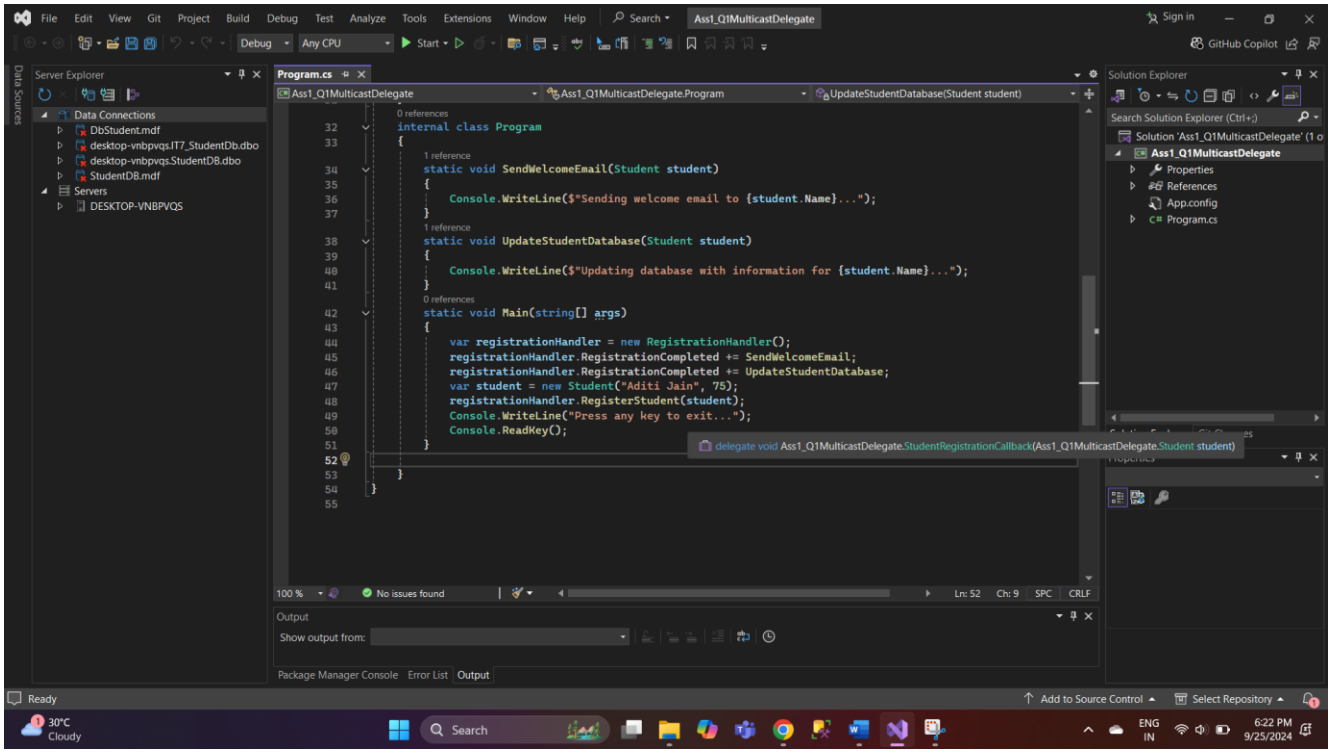
=>

Program.cs

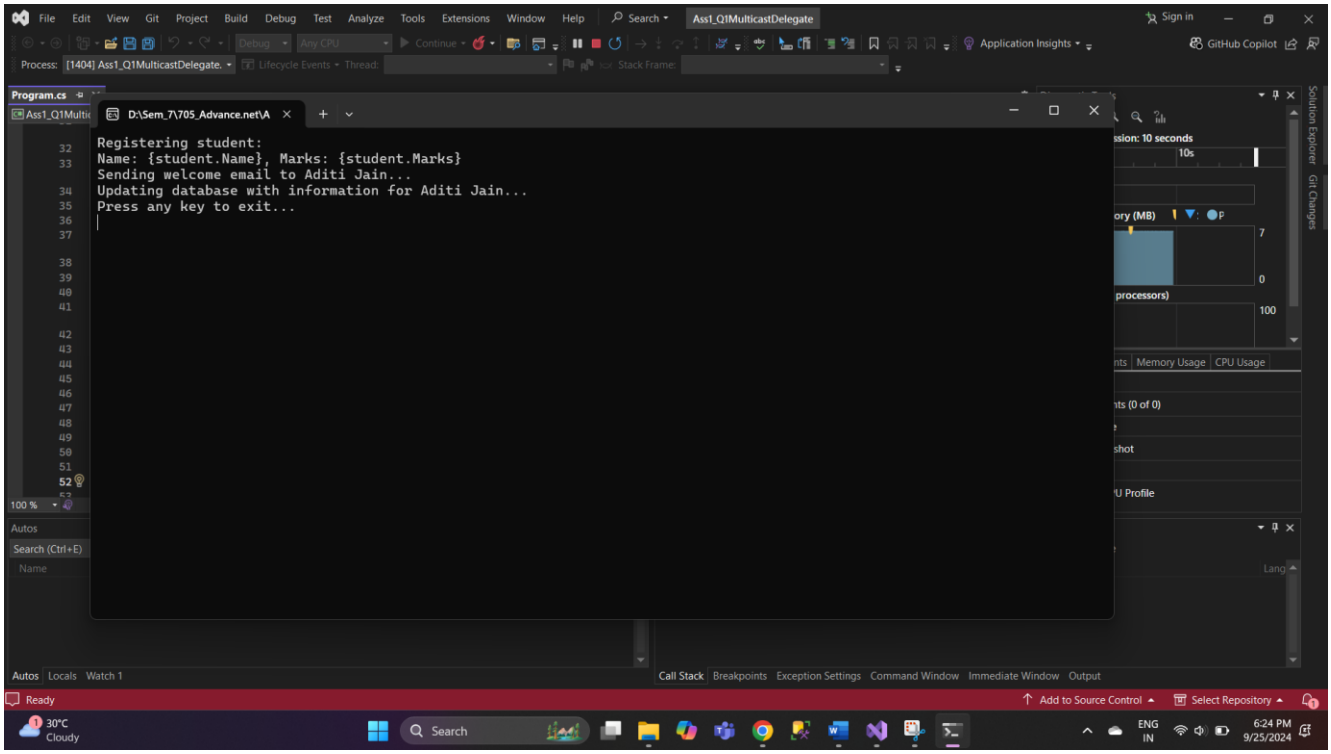
```
namespace Ass1_Q1MulticastDelegate
{    public class Student
   {   public string Name { get; set; }
       public int Marks { get; set; }
       // Constructor to initialize the student details
       public Student(string name, int marks)
       {   Name = name;
           Marks = marks;
       }
   }
   public delegate void StudentRegistrationCallback(Student student);
   public class RegistrationHandler
   {   public event StudentRegistrationCallback RegistrationCompleted;
       public void RegisterStudent(Student student)
       {  Console.WriteLine($"Registering student: ");
           Console.WriteLine("Name: {student.Name}, Marks: {student.Marks}");
           RegistrationCompleted?.Invoke(student);
       }
   }
   internal class Program
   {
       static void SendWelcomeEmail(Student student)
       {
           Console.WriteLine($"Sending welcome email to {student.Name}...");
       }
       static void UpdateStudentDatabase(Student student)
       {
```

```
        Console.WriteLine($"Updating database with information for {student.Name}...");
    }
    static void Main(string[] args)
    {
        var registrationHandler = new RegistrationHandler();
        registrationHandler.RegistrationCompleted += SendWelcomeEmail;
        registrationHandler.RegistrationCompleted += UpdateStudentDatabase;
        var student = new Student("Aditi Jain", 75);
        registrationHandler.RegisterStudent(student);
        Console.WriteLine("Press any key to exit...");
        Console.ReadKey();
    }
  }
}
```




Ass1_Q1MulticastDelegate
Program.cs
Ass1_Q1MulticastDelegate.Program
UpdateStudentDatabase(Student student)
internal class Program
static void SendWelcomeEmail(Student student)
Console.WriteLine($"Sending welcome email to {student.Name}...");
static void UpdateStudentDatabase(Student student)
Console.WriteLine($"Updating database with information for {student.Name}...");
static void Main(string[] args)
var registrationHandler = new RegistrationHandler();
registrationHandler.RegistrationCompleted += SendWelcomeEmail;
registrationHandler.RegistrationCompleted += UpdateStudentDatabase;
var student = new Student("Aditi Jain", 75);
registrationHandler.RegisterStudent(student);
Console.WriteLine("Press any key to exit...");
Console.ReadKey();
delegate void Ass1_Q1MulticastDelegate.StudentRegistrationCallback(Ass1_Q1MulticastDelegate.Student student)
Solution Explorer
Ass1_Q1MulticastDelegate
Properties
References
App.config
Program.cs
Server Explorer
Data Connections
DbStudent.mdf
StudentDB.mdf
Servers
DESKTOP-VNBPVQS
Output
Package Manager Console
Error List
Output
Ready
Add to Source Control
Select Repository

Output :-


D:\Sem_7\705_Advance.net\A
Registering student:
Name: {student.Name}, Marks: {student.Marks}
Sending welcome email to Aditi Jain...
Updating database with information for Aditi Jain...
Press any key to exit...
Program.cs
Ass1_Q1MulticastDelegate
Memory Usage
CPU Usage
Autos
Locals
Watch 1
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Ready
Add to Source Control
Select Repository

Q.2 Create a C# application using generics class Matrix to perform addition subtraction and multiplication of two matrix as per user choice of datatype.

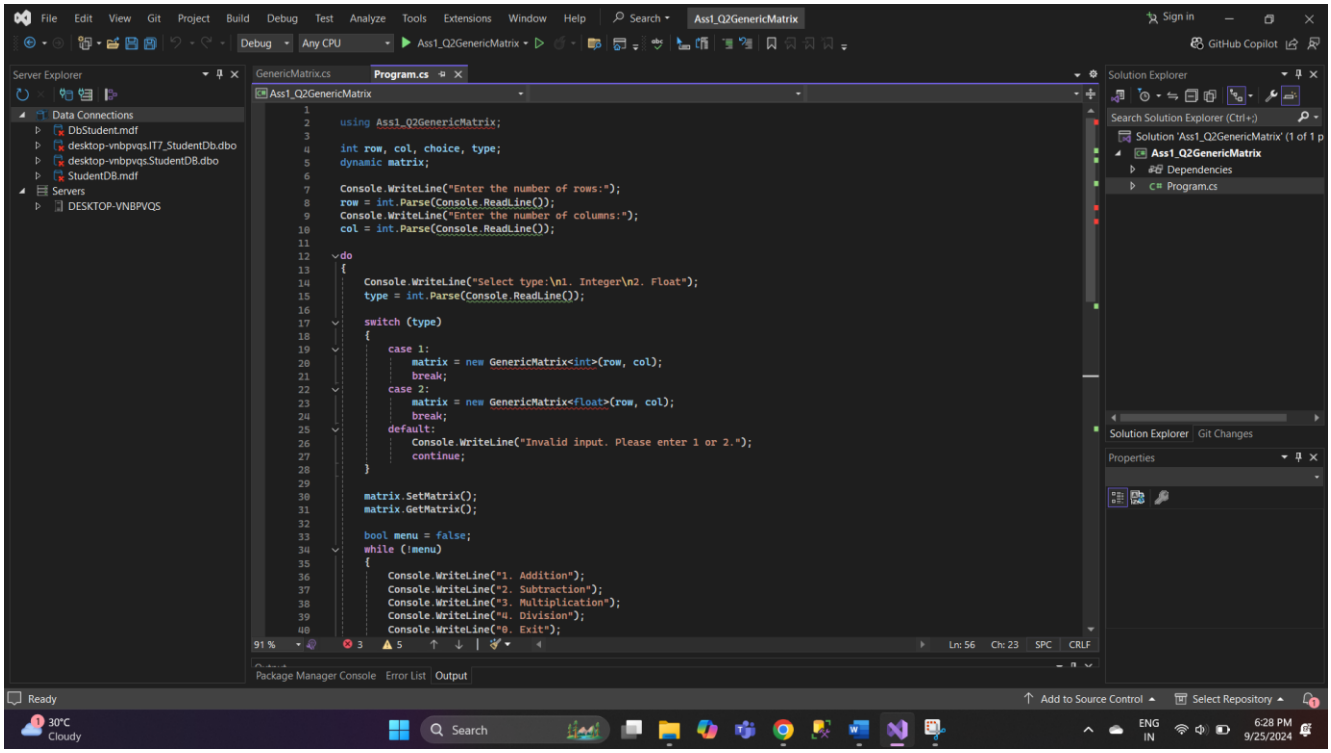
=>

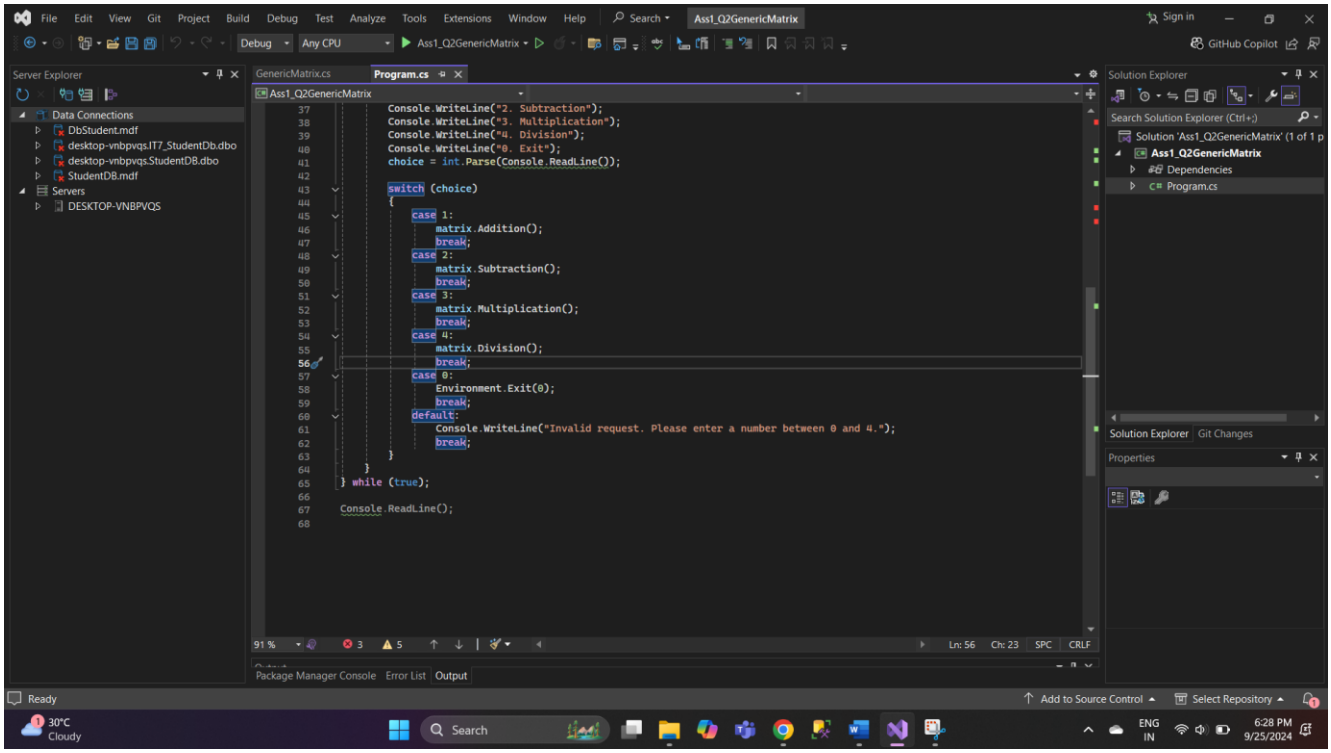
Program.cs

```
using Ass1_Q2GenericMatrix;
int row, col, choice, type;
dynamic matrix;
Console.WriteLine("Enter the number of rows:");
row = int.Parse(Console.ReadLine());
Console.WriteLine("Enter the number of columns:");
col = int.Parse(Console.ReadLine());
do
{
   Console.WriteLine("Select type:\n1. Integer\n2. Float");
   type = int.Parse(Console.ReadLine());
   switch (type)
   {
      case 1:
         matrix = new GenericMatrix<int>(row, col);
         break;
      case 2:
         matrix = new GenericMatrix<float>(row, col);
         break;
      default:
         Console.WriteLine("Invalid input. Please enter 1 or 2.");
         continue;
   }
```

```
matrix.SetMatrix();
matrix.GetMatrix();

bool menu = false;
while (!menu)
{
   Console.WriteLine("1. Addition");
   Console.WriteLine("2. Subtraction");
   Console.WriteLine("3. Multiplication");
   Console.WriteLine("4. Division");
   Console.WriteLine("0. Exit");
   choice = int.Parse(Console.ReadLine());

   switch (choice)
   {
      case 1:
         matrix.Addition();
         break;
      case 2:
         matrix.Subtraction();
         break;
      case 3:
         matrix.Multiplication();
         break;
      case 4:
         matrix.Division();
         break;
      case 0:
         Environment.Exit(0);
         break;
      default:
```

```
            Console.WriteLine("Invalid request. Please enter a number between 0 and 4.");
            break;
        }
    }
} while (true);
Console.ReadLine();
```





GenericMatrix.cs

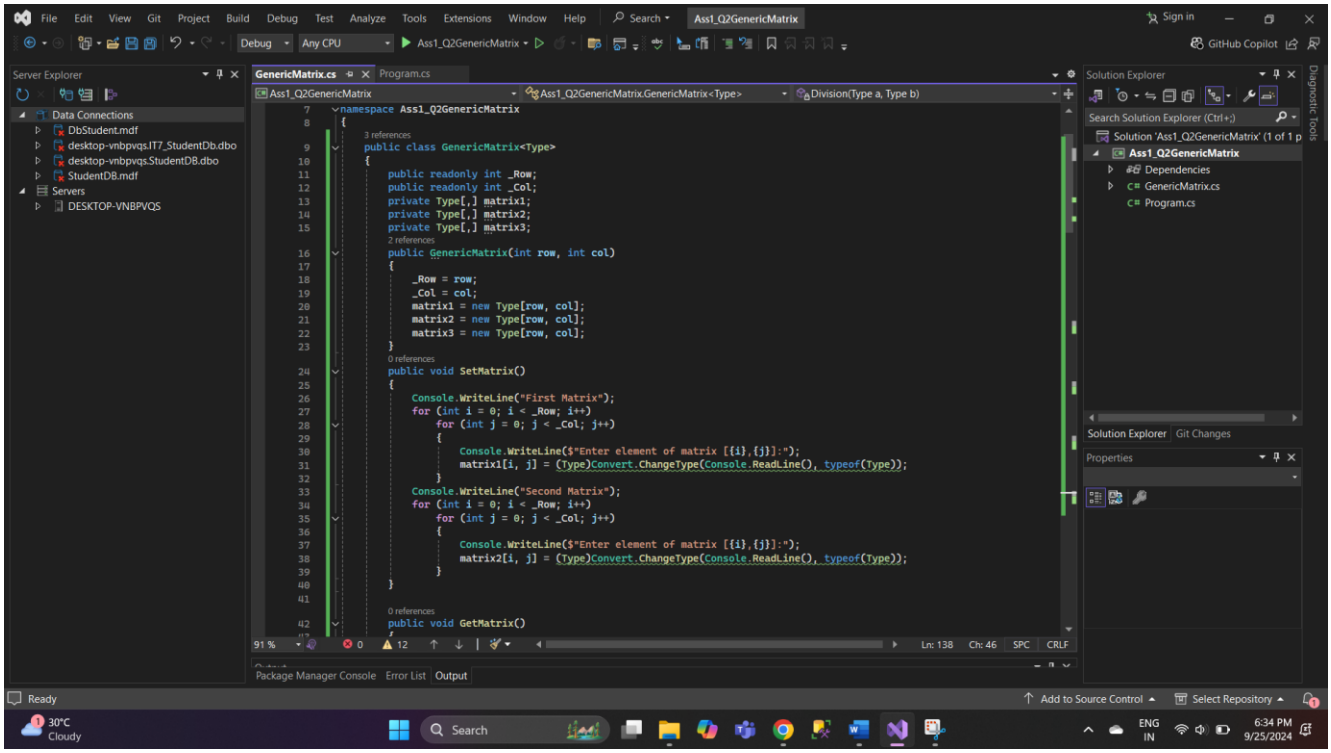
```
namespace Ass1_Q2GenericMatrix
{   public class GenericMatrix<Type>
    {
        public readonly int _Row;
        public readonly int _Col;
        private Type[,] matrix1;
        private Type[,] matrix2;
        private Type[,] matrix3;
        public GenericMatrix(int row, int col)
        {
            _Row = row;
            _Col = col;
            matrix1 = new Type[row, col];
            matrix2 = new Type[row, col];
            matrix3 = new Type[row, col];
        }
        public void SetMatrix()
```

```
{
    Console.WriteLine("First Matrix");
    for (int i = 0; i < _Row; i++)
        for (int j = 0; j < _Col; j++)
        {
            Console.WriteLine($"Enter element of matrix [{i},{j}]:");
            matrix1[i, j] = (Type)Convert.ChangeType(Console.ReadLine(), typeof(Type));
        }
    Console.WriteLine("Second Matrix");
    for (int i = 0; i < _Row; i++)
        for (int j = 0; j < _Col; j++)
        {
            Console.WriteLine($"Enter element of matrix [{i},{j}]:");
            matrix2[i, j] = (Type)Convert.ChangeType(Console.ReadLine(), typeof(Type));
        }
}

public void GetMatrix()
{
    Console.WriteLine("First Matrix");
    for (int i = 0; i < _Row; i++)
    {
        for (int j = 0; j < _Col; j++)
            Console.Write(matrix1[i, j] + "\t");
        Console.WriteLine("\n");
    }
    Console.WriteLine("Second Matrix");
    for (int i = 0; i < _Row; i++)
    {
        for (int j = 0; j < _Col; j++)
            Console.Write(matrix2[i, j] + "\t");
```

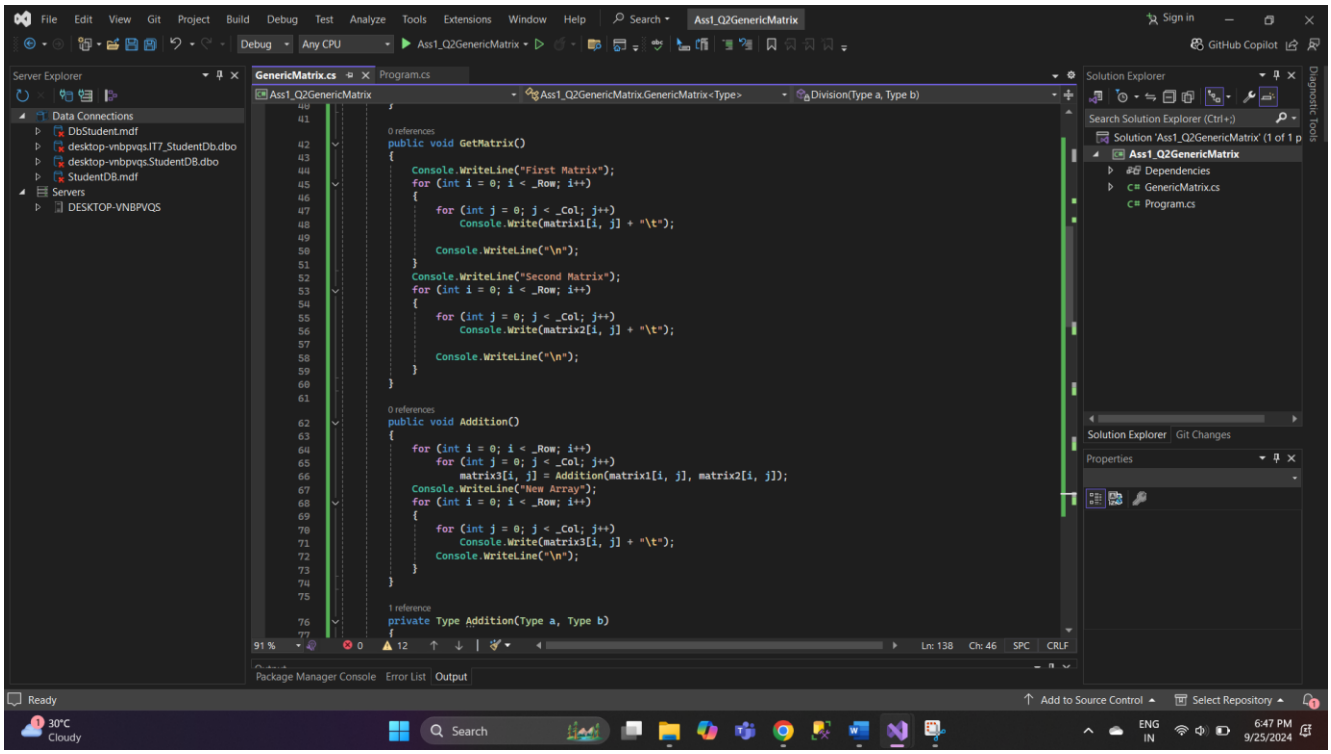
```
        Console.WriteLine("\n");
    }
}

public void Addition()
{
    for (int i = 0; i < _Row; i++)
        for (int j = 0; j < _Col; j++)
            matrix3[i, j] = Addition(matrix1[i, j], matrix2[i, j]);
    Console.WriteLine("New Array");
    for (int i = 0; i < _Row; i++)
    {
        for (int j = 0; j < _Col; j++)
            Console.Write(matrix3[i, j] + "\t");
        Console.WriteLine("\n");
    }
}

private Type Addition(Type a, Type b)
{
    dynamic x = a;
    dynamic y = b;
    return x + y;
}

public void Subtraction()
{
    for (int i = 0; i < _Row; i++)
        for (int j = 0; j < _Col; j++)
            matrix3[i, j] = Subtraction(matrix1[i, j], matrix2[i, j]);
```

```
    Console.WriteLine("New Array");
    for (int i = 0; i < _Row; i++)
    {
        for (int j = 0; j < _Col; j++)
            Console.Write(matrix3[i, j] + "\t");
        Console.WriteLine("\n");
    }
}

private Type Subtraction(Type a, Type b)
{
    dynamic x = a;
    dynamic y = b;
    return x - y;
}
public void Multiplication()
{
    for (int i = 0; i < _Row; i++)
        for (int j = 0; j < _Col; j++)
            matrix3[i, j] = Multiplication(matrix1[i, j], matrix2[i, j]);

    Console.WriteLine("New Array");
    for (int i = 0; i < _Row; i++)
    {
        for (int j = 0; j < _Col; j++)
            Console.Write(matrix3[i, j] + "\t");
        Console.WriteLine("\n");
    }
}
private Type Multiplication(Type a, Type b)
```

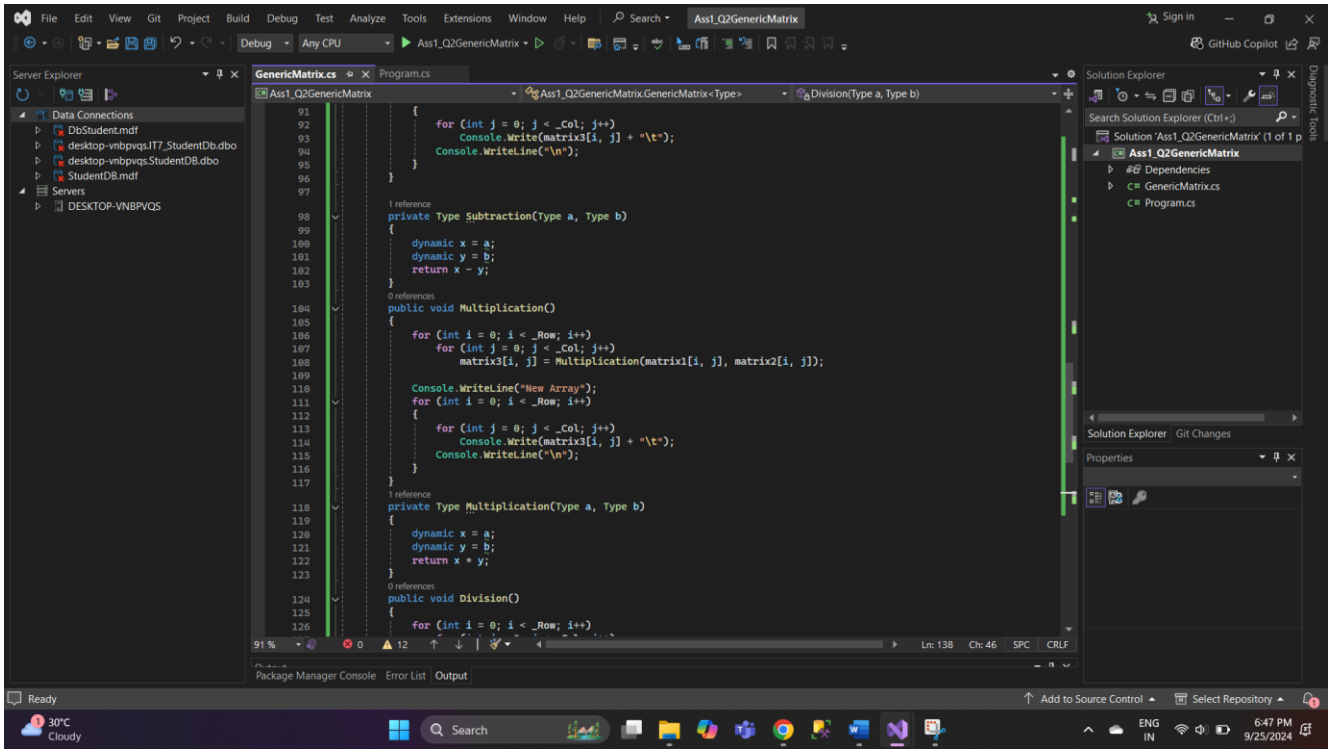
```
    {
        dynamic x = a;
        dynamic y = b;
        return x * y;
    }
    public void Division()
    {
        for (int i = 0; i < _Row; i++)
            for (int j = 0; j < _Col; j++)
                matrix3[i, j] = Division(matrix1[i, j], matrix2[i, j]);
        Console.WriteLine("New Array");
        for (int i = 0; i < _Row; i++)
        {
            for (int j = 0; j < _Col; j++)
                Console.Write(matrix3[i, j] + "\t");
            Console.WriteLine("\n");
        }
    }
    private Type Division(Type a, Type b)
    {
        dynamic x = a;
        dynamic y = b;
        return x / y;
    }
  }
}
```



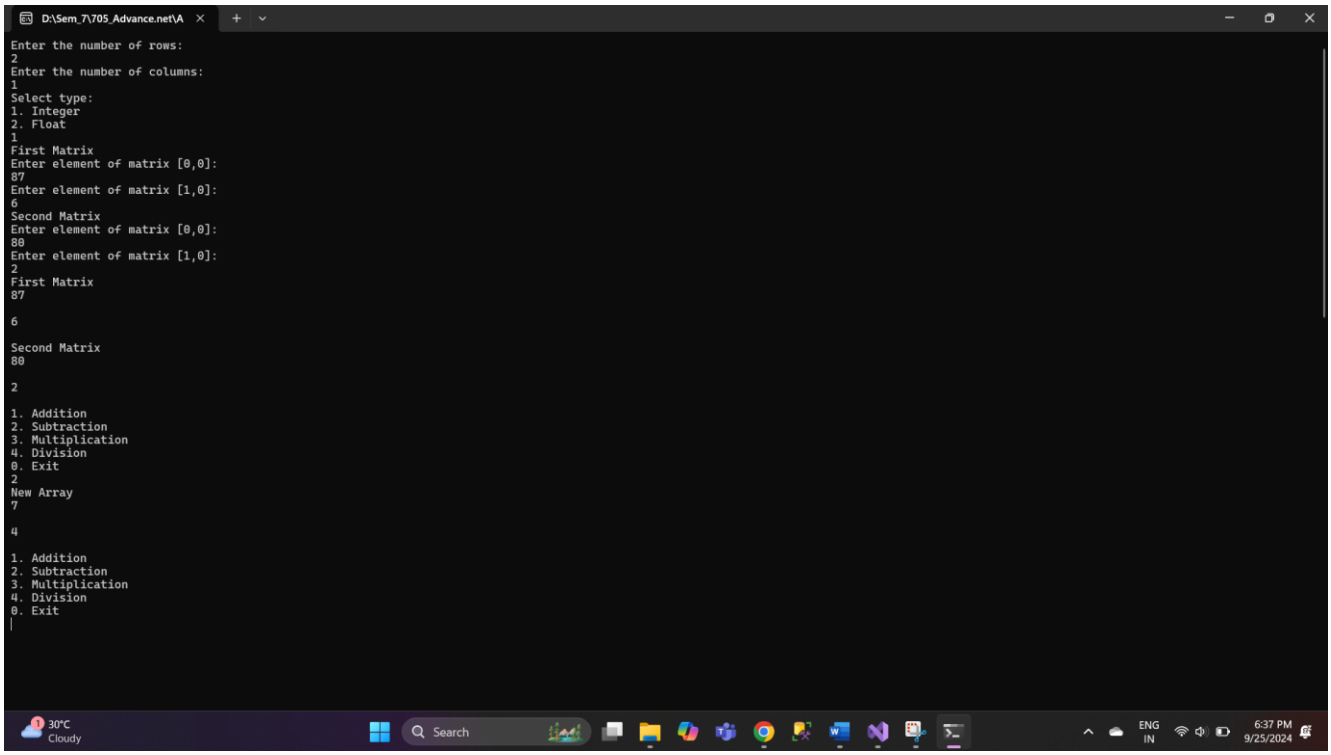
```
namespace Ass1_Q2GenericMatrix
{
    3 references
    public class GenericMatrix<Type>
    {
        public readonly int _Row;
        public readonly int _Col;
        private Type[,] matrix1;
        private Type[,] matrix2;
        private Type[,] matrix3;
        2 references
        public GenericMatrix(int row, int col)
        {
            _Row = row;
            _Col = col;
            matrix1 = new Type[row, col];
            matrix2 = new Type[row, col];
            matrix3 = new Type[row, col];
        }
        0 references
        public void SetMatrix()
        {
            Console.WriteLine("First Matrix");
            for (int i = 0; i < _Row; i++)
                for (int j = 0; j < _Col; j++)
                {
                    Console.WriteLine($"Enter element of matrix [{i},{j}]:");
                    matrix1[i, j] = (Type)Convert.ChangeType(Console.ReadLine(), typeof(Type));
                }
            Console.WriteLine("Second Matrix");
            for (int i = 0; i < _Row; i++)
                for (int j = 0; j < _Col; j++)
                {
                    Console.WriteLine($"Enter element of matrix [{i},{j}]:");
                    matrix2[i, j] = (Type)Convert.ChangeType(Console.ReadLine(), typeof(Type));
                }
        }

        0 references
        public void GetMatrix()
```



```
        0 references
        public void GetMatrix()
        {
            Console.WriteLine("First Matrix");
            for (int i = 0; i < _Row; i++)
            {
                for (int j = 0; j < _Col; j++)
                    Console.Write(matrix1[i, j] + "\t");

                Console.WriteLine("\n");
            }
            Console.WriteLine("Second Matrix");
            for (int i = 0; i < _Row; i++)
            {
                for (int j = 0; j < _Col; j++)
                    Console.Write(matrix2[i, j] + "\t");

                Console.WriteLine("\n");
            }
        }

        0 references
        public void Addition()
        {
            for (int i = 0; i < _Row; i++)
                for (int j = 0; j < _Col; j++)
                    matrix3[i, j] = Addition(matrix1[i, j], matrix2[i, j]);
            Console.WriteLine("New Array");
            for (int i = 0; i < _Row; i++)
            {
                for (int j = 0; j < _Col; j++)
                    Console.Write(matrix3[i, j] + "\t");
                Console.WriteLine("\n");
            }
        }

        1 reference
        private Type Addition(Type a, Type b)
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Ass1_Q2GenericMatrix
Debug
Any CPU
GenericMatrix.cs
Program.cs
Server Explorer
Data Connections
DbStudent.mdf
desktop-vnbpvqs.IT7_StudentDb.dbo
desktop-vnbpvqs.StudentDB.dbo
StudentDB.mdf
Servers
DESKTOP-VNBPVQS
Solution Explorer
Solution 'Ass1_Q2GenericMatrix' (1 of 1 p
Dependencies
GenericMatrix.cs
Program.cs
Properties
Package Manager Console
Error List
Output
Ready
Add to Source Control
Select Repository
30°C
Cloudy
Search
ENG
IN
6:47 PM
9/25/2024

Output:-


D:\Sem_7\705_Advance.net\A
Enter the number of rows:
2
Enter the number of columns:
1
Select type:
1. Integer
2. Float
1
First Matrix
Enter element of matrix [0,0]:
87
Enter element of matrix [1,0]:
6
Second Matrix
Enter element of matrix [0,0]:
80
Enter element of matrix [1,0]:
2
First Matrix
87
6
Second Matrix
80
2
1. Addition
2. Subtraction
3. Multiplication
4. Division
0. Exit
2
New Array
7
4
1. Addition
2. Subtraction
3. Multiplication
4. Division
0. Exit
30°C
Cloudy
Search
ENG
IN
6:37 PM
9/25/2024

Q.3 Create a C# menu driven console application to manage employee data for following requirements:

A. User should enter employee using generic collection.
B. User must be able to update and delete employee details
C. User should be able to filter employee data as per their choice.

=>

Program.cs

```
namespace Ass1_Q3EmployeeDetail
{
   // Define the Employee class
   public class Employee
   {
      public int Id { get; set; }
      public string Name { get; set; }
      public int Age { get; set; }
      public string Department { get; set; }

      public Employee(int id, string name, int age, string department)
      {
         Id = id;
         Name = name;
         Age = age;
         Department = department;
      }

      public override string ToString()
      {
         return $"ID: {Id}, Name: {Name}, Age: {Age}, Department: {Department}";
      }
   }
```

```
class Program
{
    // Generic collection to store employee data
    static List<Employee> employees = new List<Employee>();
    static int employeeIdCounter = 1; // Auto-increment for employee IDs
    static void Main(string[] args)
    {
        bool exit = false;
        while (!exit)
        {
            Console.Clear();
            Console.WriteLine("Employee Management System");
            Console.WriteLine("1. Add Employee");
            Console.WriteLine("2. Update Employee");
            Console.WriteLine("3. Delete Employee");
            Console.WriteLine("4. Filter Employees");
            Console.WriteLine("5. View All Employees");
            Console.WriteLine("6. Exit");
            Console.Write("Choose an option: ");
            string choice = Console.ReadLine();
            switch (choice)
            {
                case "1":
                    AddEmployee();
                    break;
                case "2":
                    UpdateEmployee();
                    break;
                case "3":
                    DeleteEmployee();
                    break;
```

```
            case "4":
                FilterEmployees();
                break;
            case "5":
                ViewAllEmployees();
                break;
            case "6":
                exit = true;
                break;
            default:
                Console.WriteLine("Invalid option! Please try again.");
                break;
        }
    }
}

static void AddEmployee()
{
    Console.Write("Enter Employee Name: ");
    string name = Console.ReadLine();
    Console.Write("Enter Employee Age: ");
    int age = int.Parse(Console.ReadLine());
    Console.Write("Enter Employee Department: ");
    string department = Console.ReadLine();

    // Add new employee to the collection
    employees.Add(new Employee(employeeIdCounter++, name, age, department));

    Console.WriteLine("Employee added successfully!");
    Pause();
}
```

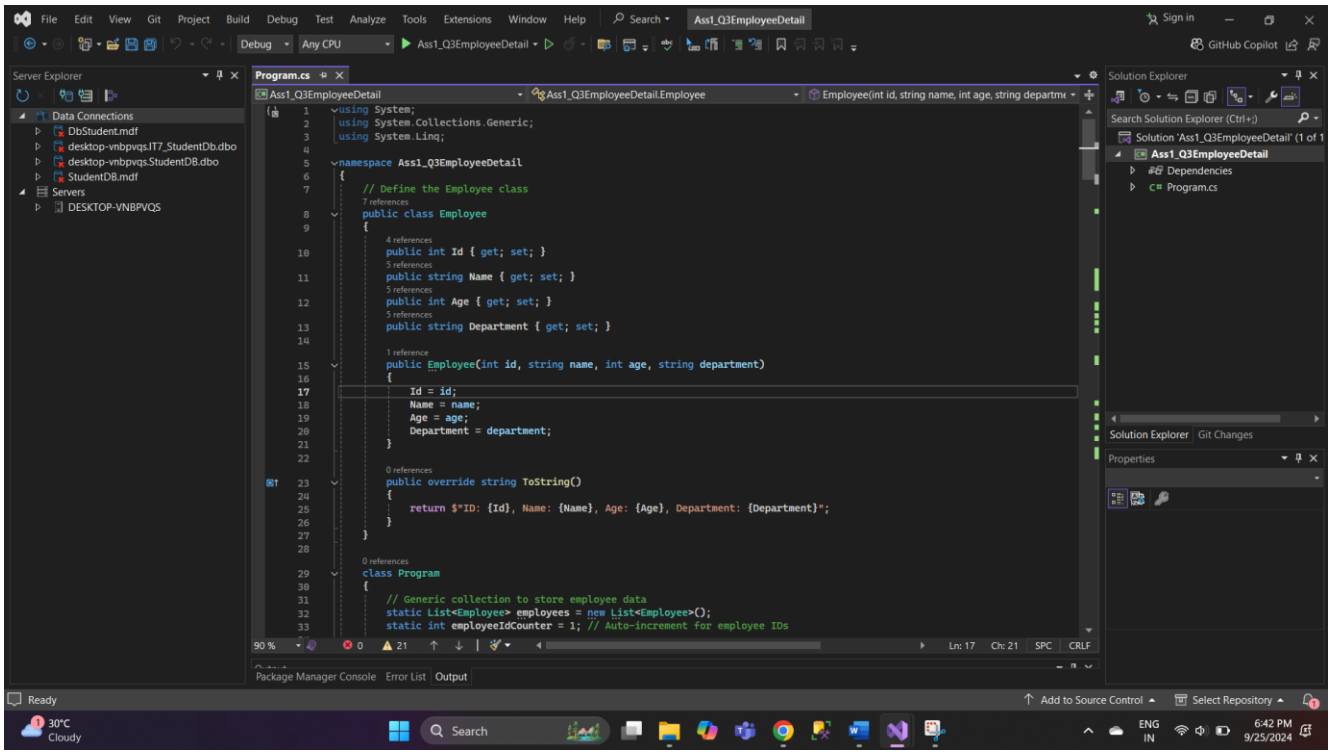
```
static void UpdateEmployee()
{
    Console.Write("Enter Employee ID to update: ");
    int id = int.Parse(Console.ReadLine());

    Employee employee = employees.FirstOrDefault(e => e.Id == id);
    if (employee != null)
    {
        Console.Write("Enter new Name (leave blank to keep current): ");
        string name = Console.ReadLine();
        if (!string.IsNullOrEmpty(name)) employee.Name = name;

        Console.Write("Enter new Age (leave blank to keep current): ");
        string ageInput = Console.ReadLine();
        if (!string.IsNullOrEmpty(ageInput)) employee.Age = int.Parse(ageInput);

        Console.Write("Enter new Department (leave blank to keep current): ");
        string department = Console.ReadLine();
        if (!string.IsNullOrEmpty(department)) employee.Department = department;

        Console.WriteLine("Employee updated successfully!");
    }
    else
    {
        Console.WriteLine("Employee not found.");
    }
    Pause();
}

static void DeleteEmployee()
```

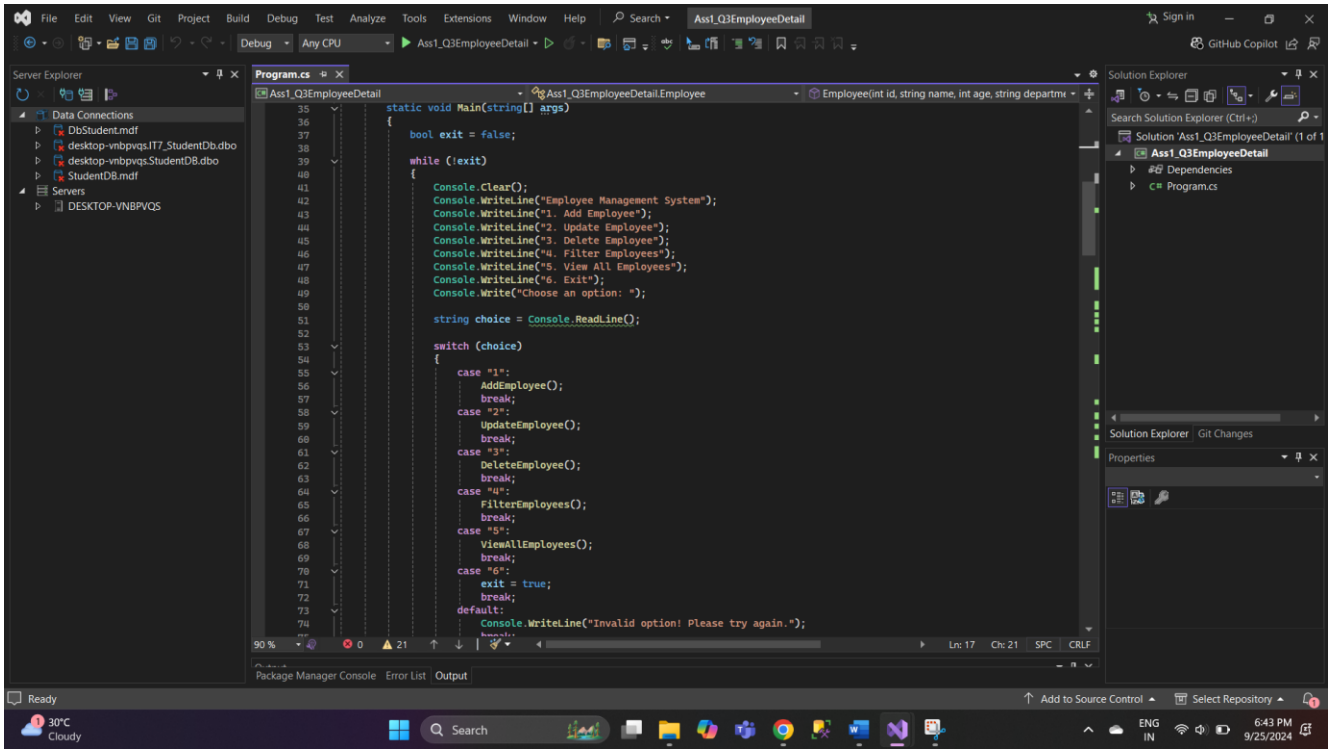
```
{
    Console.Write("Enter Employee ID to delete: ");
    int id = int.Parse(Console.ReadLine());

    Employee employee = employees.FirstOrDefault(e => e.Id == id);
    if (employee != null)
    {
        employees.Remove(employee);
        Console.WriteLine("Employee deleted successfully!");
    }
    else
    {
        Console.WriteLine("Employee not found.");
    }
    Pause();
}

static void FilterEmployees()
{
    Console.WriteLine("Filter Options:");
    Console.WriteLine("1. By Name");
    Console.WriteLine("2. By Age");
    Console.WriteLine("3. By Department");
    Console.WriteLine("4. Multiple Criteria");
    Console.Write("Choose a filter option: ");

    string choice = Console.ReadLine();
    IEnumerable<Employee> filteredEmployees = employees;

    switch (choice)
    {
```

```
            case "1":
                Console.Write("Enter Name to filter by: ");
                string name = Console.ReadLine();
                filteredEmployees = employees.Where(e => e.Name.Contains(name,
StringComparison.OrdinalIgnoreCase));
                break;

            case "2":
                Console.Write("Enter Age to filter by: ");
                int age = int.Parse(Console.ReadLine());
                filteredEmployees = employees.Where(e => e.Age == age);
                break;

            case "3":
                Console.Write("Enter Department to filter by: ");
                string department = Console.ReadLine();
                filteredEmployees = employees.Where(e => e.Department.Equals(department,
StringComparison.OrdinalIgnoreCase));
                break;

            case "4":
                Console.Write("Enter Name (optional): ");
                string multiName = Console.ReadLine();
                Console.Write("Enter Age (optional): ");
                string multiAgeInput = Console.ReadLine();
                Console.Write("Enter Department (optional): ");
                string multiDept = Console.ReadLine();

                if (!string.IsNullOrEmpty(multiName))
                {
                    filteredEmployees = filteredEmployees.Where(e => e.Name.Contains(multiName,
StringComparison.OrdinalIgnoreCase));
```

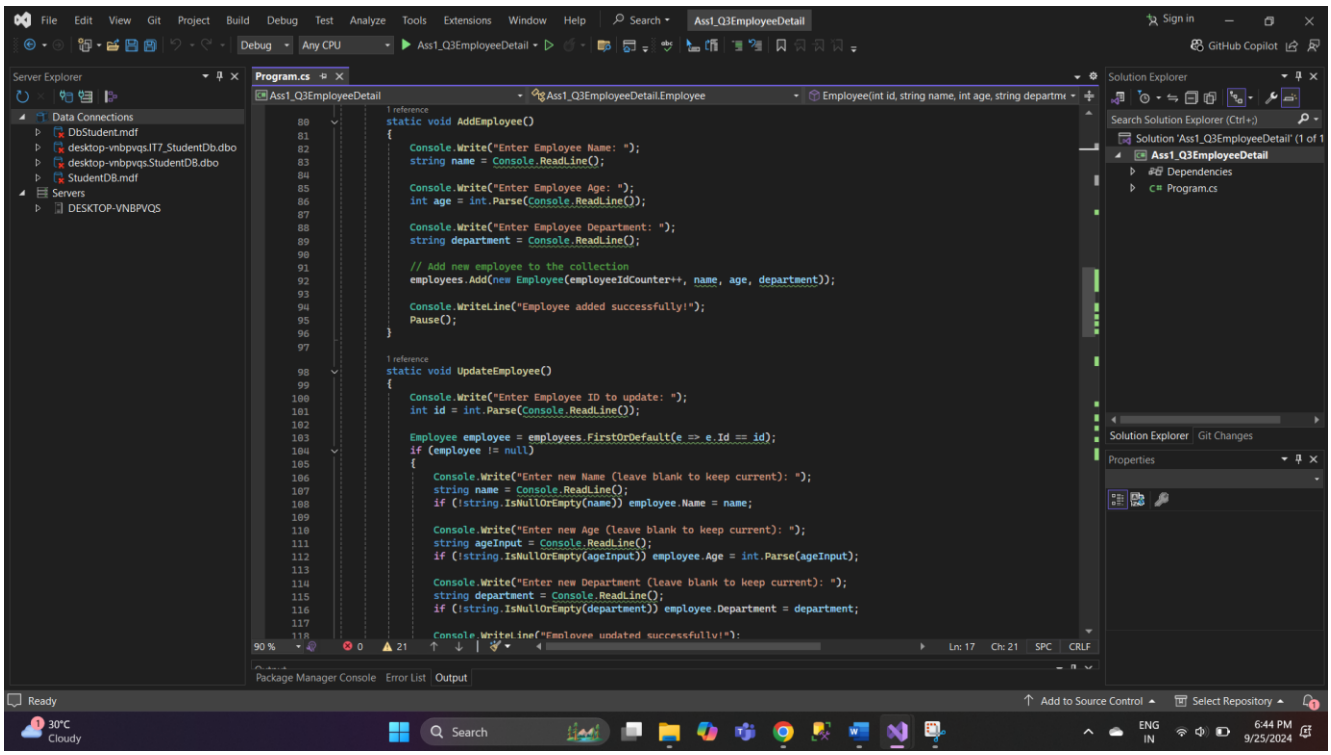
```
            }

            if (!string.IsNullOrEmpty(multiAgeInput) && int.TryParse(multiAgeInput, out int multiAge))
            {
                filteredEmployees = filteredEmployees.Where(e => e.Age == multiAge);
            }

            if (!string.IsNullOrEmpty(multiDept))
            {
                filteredEmployees = filteredEmployees.Where(e => e.Department.Equals(multiDept,
StringComparison.OrdinalIgnoreCase));
            }
            break;

        default:
            Console.WriteLine("Invalid option.");
            break;
    }

    Console.WriteLine("\nFiltered Employees:");
    foreach (var emp in filteredEmployees)
    {
        Console.WriteLine(emp);
    }

    Pause();
}

static void ViewAllEmployees()
{
    Console.WriteLine("\nAll Employees:");
```

```
        foreach (var employee in employees)
        {
            Console.WriteLine(employee);
        }
        Pause();
    }

    static void Pause()
    {
        Console.WriteLine("\nPress any key to continue...");
        Console.ReadKey();
    }
  }
}
```

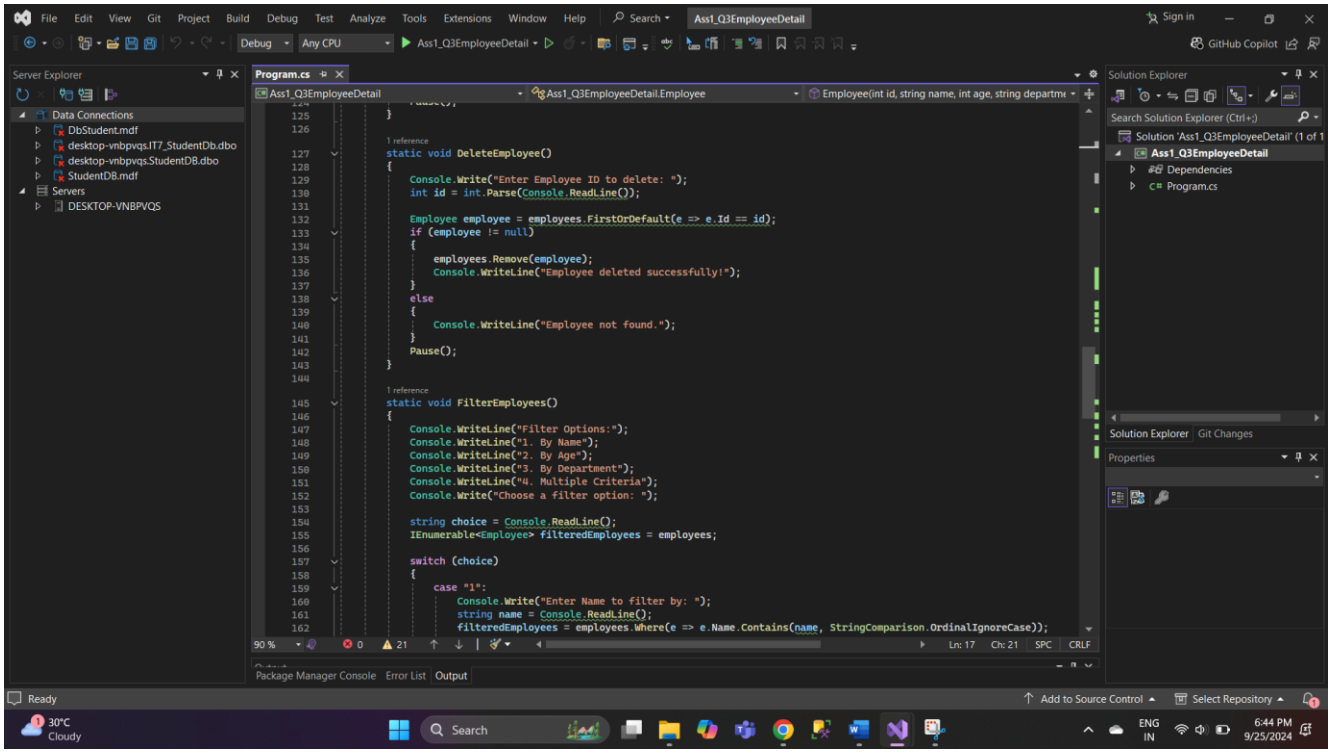




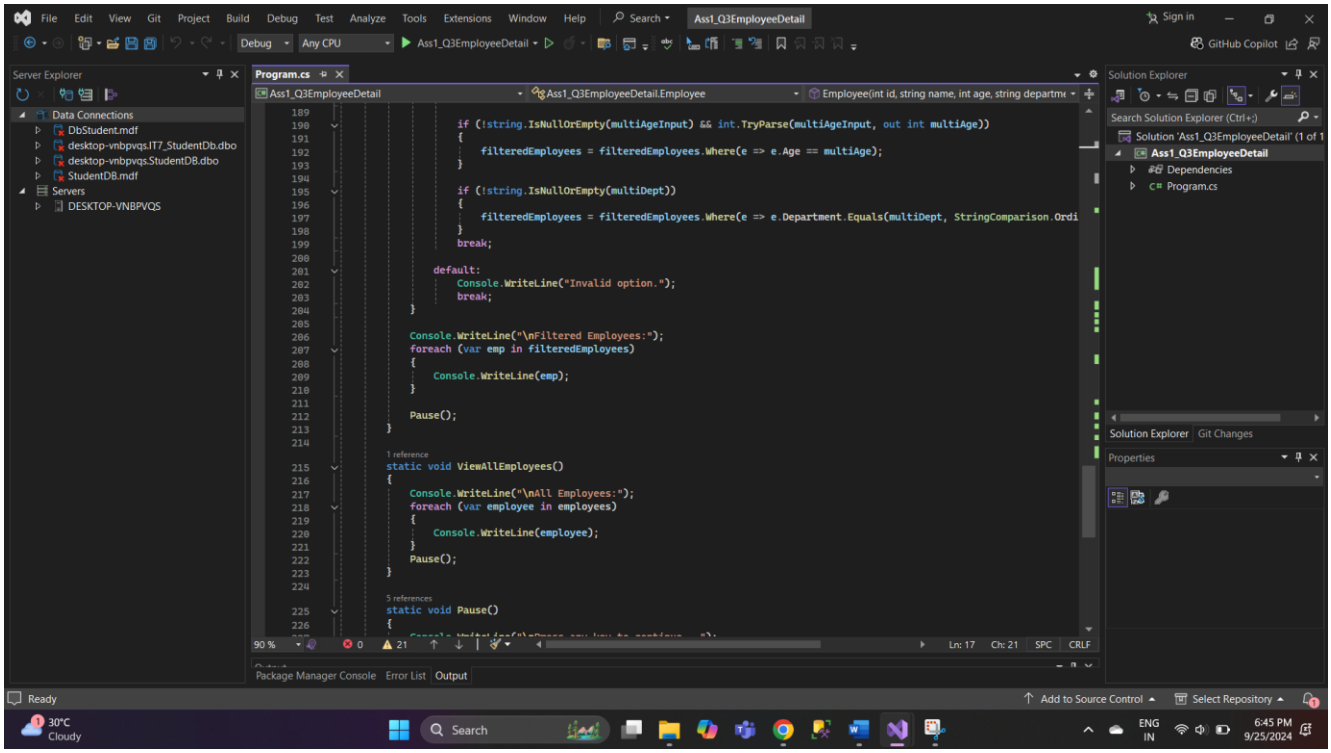
```
static void Main(string[] args)
{
    bool exit = false;

    while (!exit)
    {
        Console.Clear();
        Console.WriteLine("Employee Management System");
        Console.WriteLine("1. Add Employee");
        Console.WriteLine("2. Update Employee");
        Console.WriteLine("3. Delete Employee");
        Console.WriteLine("4. Filter Employees");
        Console.WriteLine("5. View All Employees");
        Console.WriteLine("6. Exit");
        Console.Write("Choose an option: ");

        string choice = Console.ReadLine();

        switch (choice)
        {
            case "1":
                AddEmployee();
                break;
            case "2":
                UpdateEmployee();
                break;
            case "3":
                DeleteEmployee();
                break;
            case "4":
                FilterEmployees();
                break;
            case "5":
                ViewAllEmployees();
                break;
            case "6":
                exit = true;
                break;
            default:
                Console.WriteLine("Invalid option! Please try again.");
```



```
1 reference
static void AddEmployee()
{
    Console.Write("Enter Employee Name: ");
    string name = Console.ReadLine();

    Console.Write("Enter Employee Age: ");
    int age = int.Parse(Console.ReadLine());

    Console.Write("Enter Employee Department: ");
    string department = Console.ReadLine();

    // Add new employee to the collection
    employees.Add(new Employee(employeeIdCounter++, name, age, department));

    Console.WriteLine("Employee added successfully!");
    Pause();
}

1 reference
static void UpdateEmployee()
{
    Console.Write("Enter Employee ID to update: ");
    int id = int.Parse(Console.ReadLine());

    Employee employee = employees.FirstOrDefault(e => e.Id == id);
    if (employee != null)
    {
        Console.Write("Enter new Name (leave blank to keep current): ");
        string name = Console.ReadLine();
        if (!string.IsNullOrEmpty(name)) employee.Name = name;

        Console.Write("Enter new Age (leave blank to keep current): ");
        string ageInput = Console.ReadLine();
        if (!string.IsNullOrEmpty(ageInput)) employee.Age = int.Parse(ageInput);

        Console.Write("Enter new Department (leave blank to keep current): ");
        string department = Console.ReadLine();
        if (!string.IsNullOrEmpty(department)) employee.Department = department;

        Console.WriteLine("Employee updated successfully!");
```

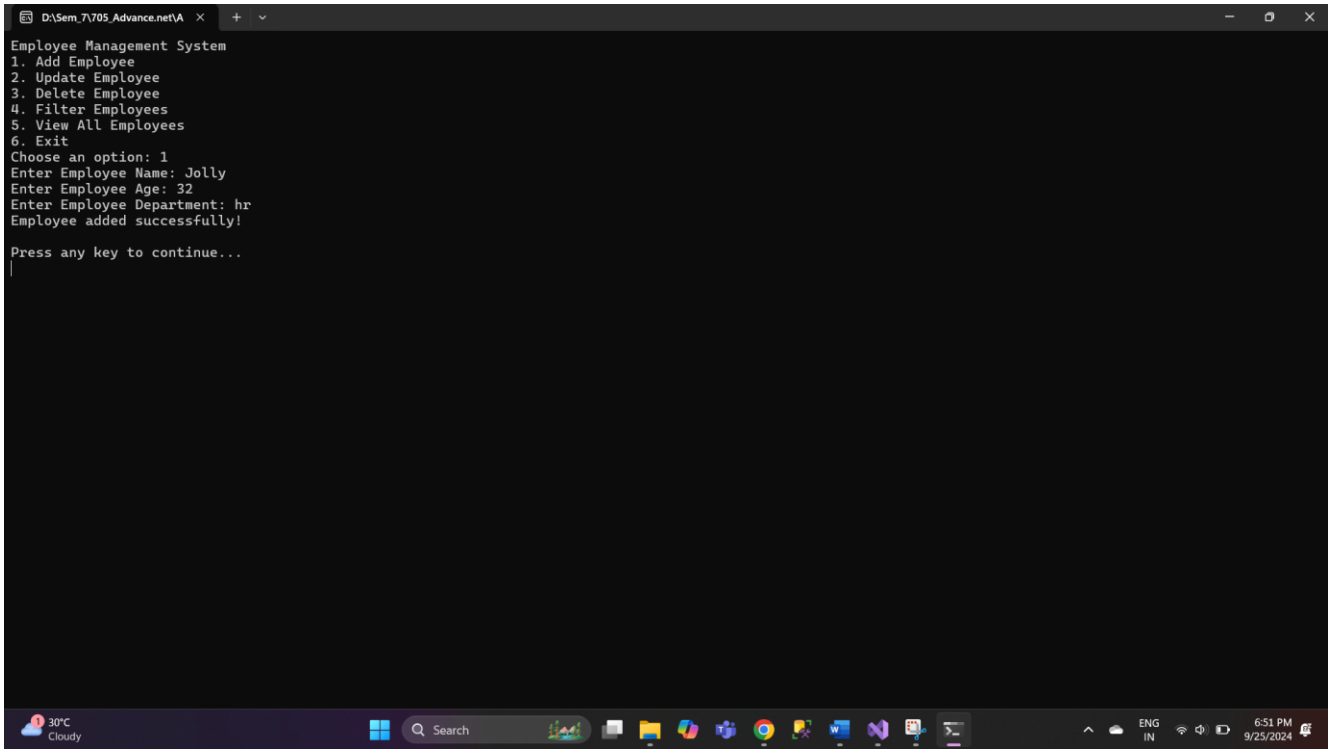


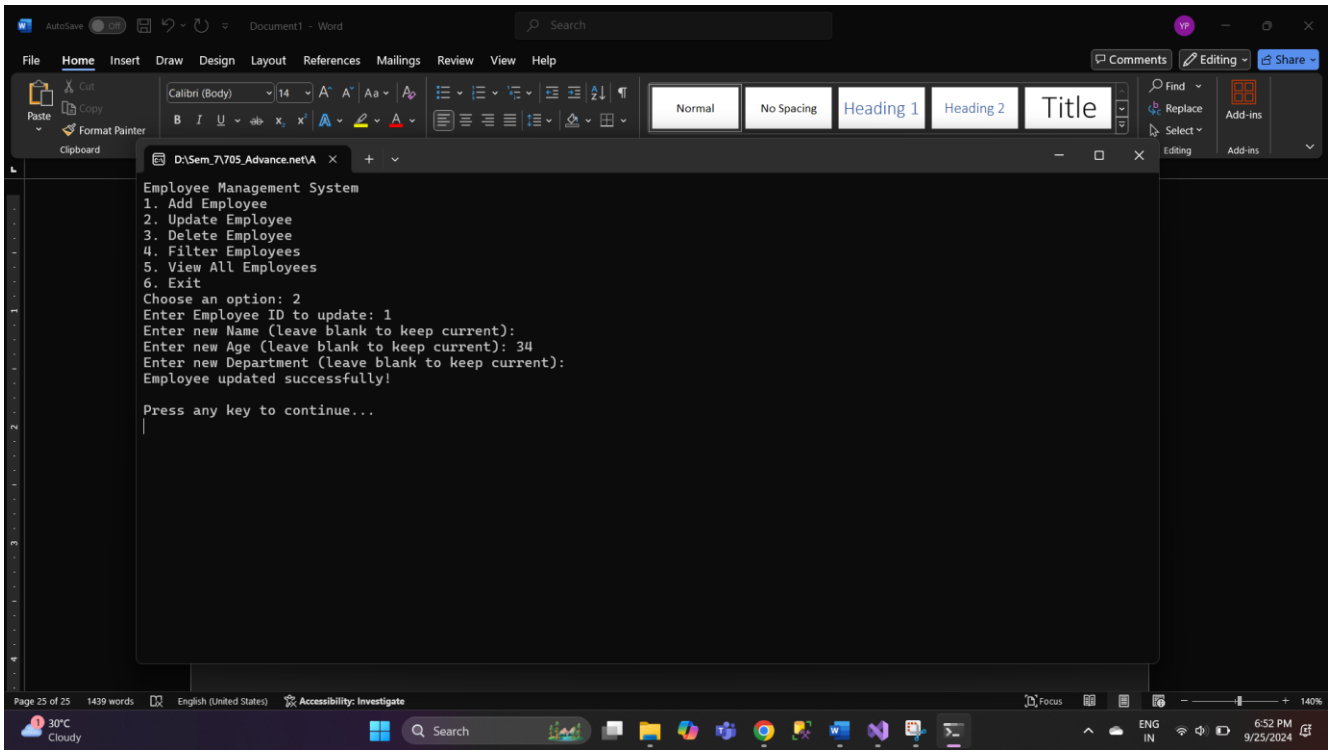
```
1 reference
static void DeleteEmployee()
{
    Console.Write("Enter Employee ID to delete: ");
    int id = int.Parse(Console.ReadLine());

    Employee employee = employees.FirstOrDefault(e => e.Id == id);
    if (employee != null)
    {
        employees.Remove(employee);
        Console.WriteLine("Employee deleted successfully!");
    }
    else
    {
        Console.WriteLine("Employee not found.");
    }
    Pause();
}

1 reference
static void FilterEmployees()
{
    Console.WriteLine("Filter Options:");
    Console.WriteLine("1. By Name");
    Console.WriteLine("2. By Age");
    Console.WriteLine("3. By Department");
    Console.WriteLine("4. Multiple Criteria");
    Console.Write("Choose a filter option: ");

    string choice = Console.ReadLine();
    IEnumerable<Employee> filteredEmployees = employees;

    switch (choice)
    {
        case "1":
            Console.Write("Enter Name to filter by: ");
            string name = Console.ReadLine();
            filteredEmployees = employees.Where(e => e.Name.Contains(name, StringComparison.OrdinalIgnoreCase));
```

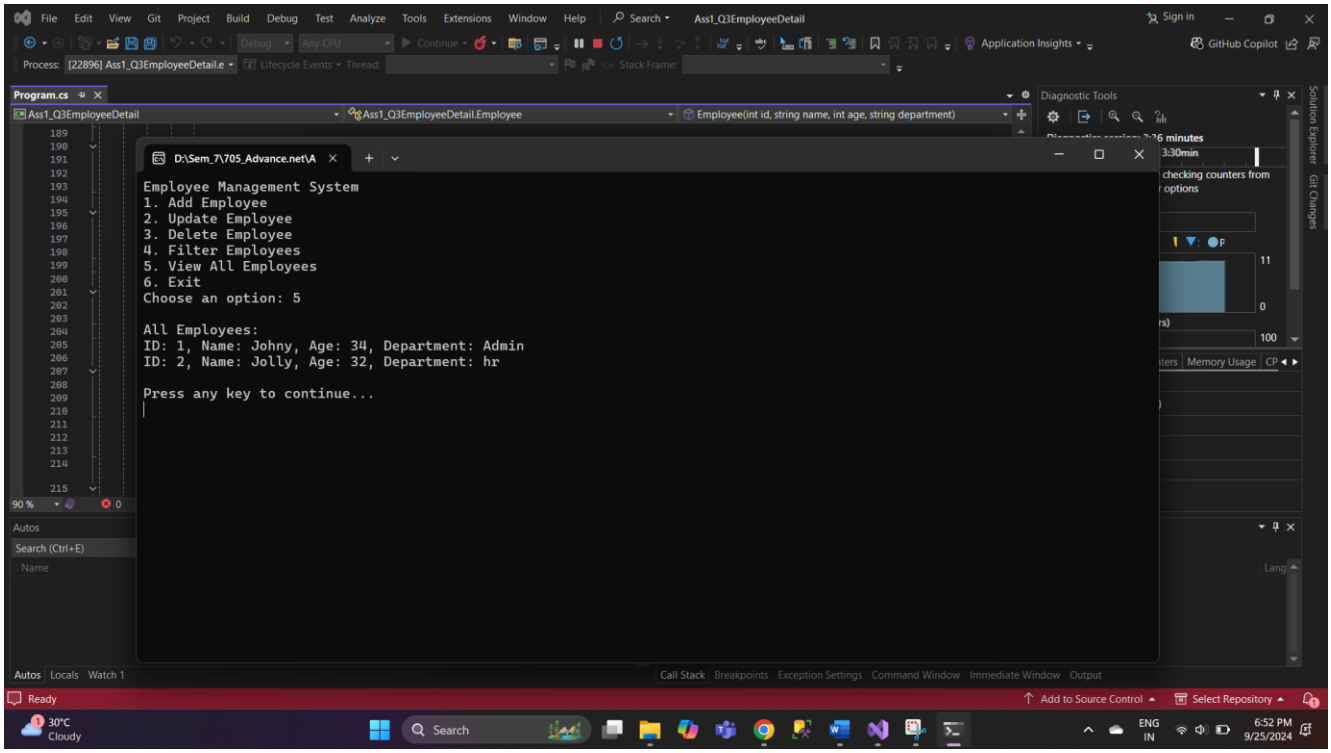


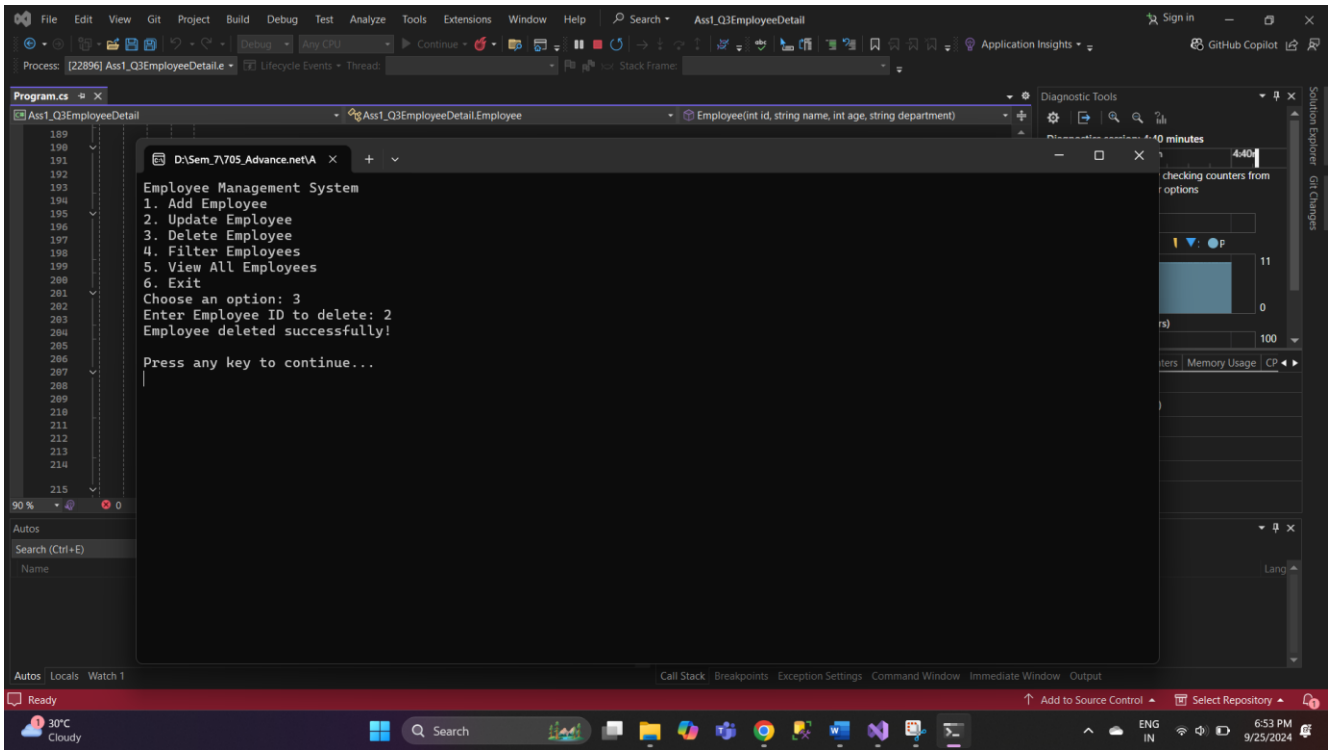
```
                if (!string.IsNullOrEmpty(multiAgeInput) && int.TryParse(multiAgeInput, out int multiAge))
                {
                    filteredEmployees = filteredEmployees.Where(e => e.Age == multiAge);
                }

                if (!string.IsNullOrEmpty(multiDept))
                {
                    filteredEmployees = filteredEmployees.Where(e => e.Department.Equals(multiDept, StringComparison.Ordi
                }
                break;

            default:
                Console.WriteLine("Invalid option.");
                break;
        }

        Console.WriteLine("\nFiltered Employees:");
        foreach (var emp in filteredEmployees)
        {
            Console.WriteLine(emp);
        }

        Pause();
    }

    1 reference
    static void ViewAllEmployees()
    {
        Console.WriteLine("\nAll Employees:");
        foreach (var employee in employees)
        {
            Console.WriteLine(employee);
        }
        Pause();
    }

    5 references
    static void Pause()
    {
```

Output:-


D:\Sem_7\705_Advance.net\A
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Filter Employees
5. View All Employees
6. Exit
Choose an option: 1
Enter Employee Name: Jolly
Enter Employee Age: 32
Enter Employee Department: hr
Employee added successfully!
Press any key to continue...
30°C
Cloudy
Search
ENG
IN
6:51 PM
9/25/2024


Document1 - Word
Search
File
Home
Insert
Draw
Design
Layout
References
Mailings
Review
View
Help
Comments
Editing
Share
Paste
Cut
Copy
Format Painter
Clipboard
Calibri (Body)
14
Normal
No Spacing
Heading 1
Heading 2
Title
Find
Replace
Select
Editing
Add-ins
D:\Sem_7\705_Advance.net\A
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Filter Employees
5. View All Employees
6. Exit
Choose an option: 2
Enter Employee ID to update: 1
Enter new Name (leave blank to keep current):
Enter new Age (leave blank to keep current): 34
Enter new Department (leave blank to keep current):
Employee updated successfully!
Press any key to continue...
Page 25 of 25
1439 words
English (United States)
Accessibility: Investigate
Focus
140%
30°C
Cloudy
Search
ENG
IN
6:52 PM
9/25/2024


D:\Sem_7\705_Advance.net\A
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Filter Employees
5. View All Employees
6. Exit
Choose an option: 5
All Employees:
ID: 1, Name: Johny, Age: 34, Department: Admin
ID: 2, Name: Jolly, Age: 32, Department: hr
Press any key to continue...


D:\Sem_7\705_Advance.net\A
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Filter Employees
5. View All Employees
6. Exit
Choose an option: 3
Enter Employee ID to delete: 2
Employee deleted successfully!
Press any key to continue...

Q.4. Create MVC core application to perform CRUD operation (on your choice of data) using in memory collection.

=>

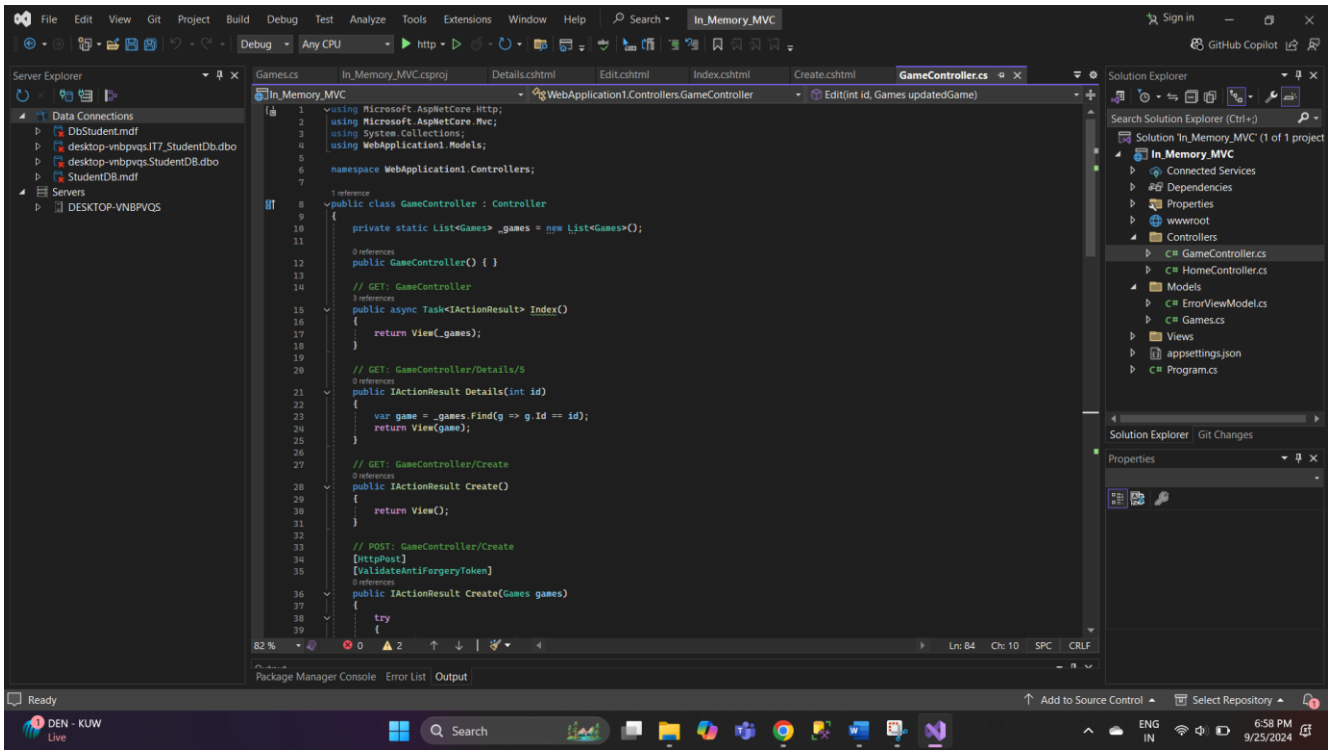
Controller:-

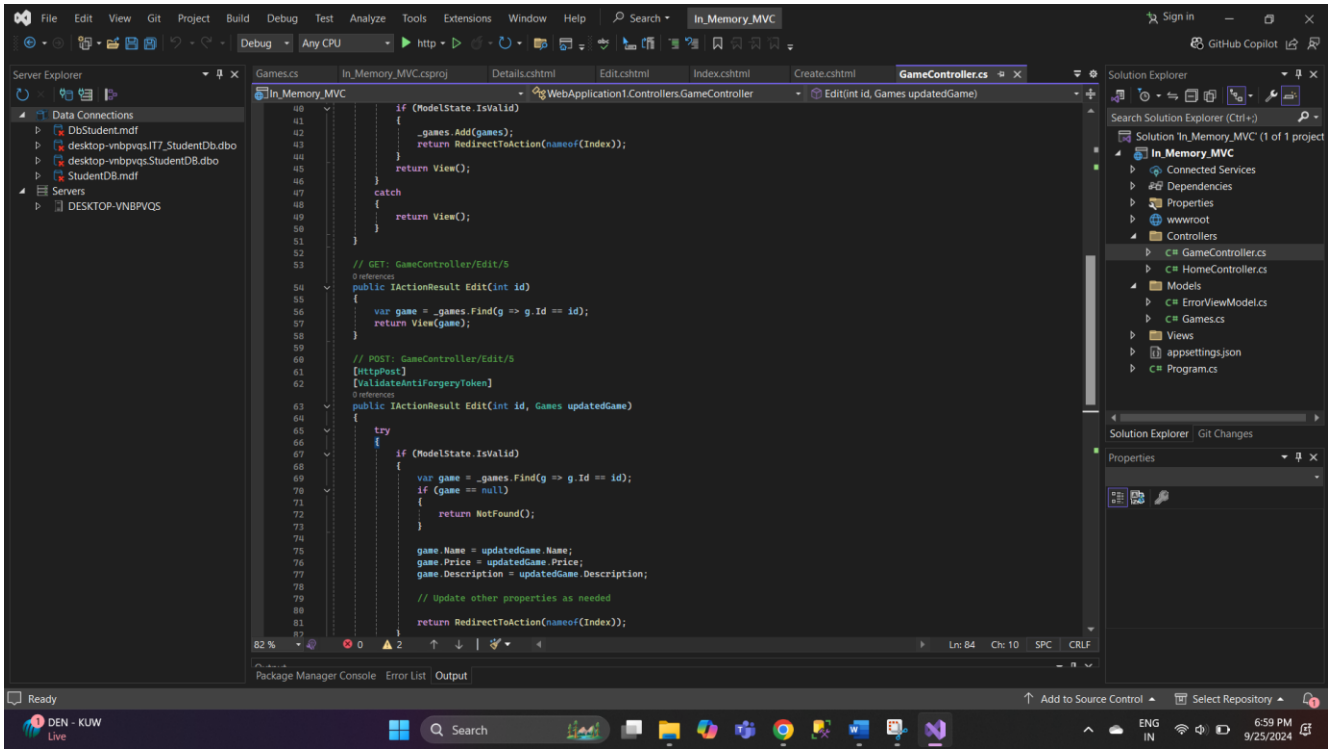
```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Collections;
using WebApplication1.Models;

namespace WebApplication1.Controllers;

public class GameController : Controller
{
    private static List<Games> _games = new List<Games>();

    public GameController() { }

    // GET: GameController
    public async Task<IActionResult> Index()
    {
        return View(_games);
    }

    // GET: GameController/Details/5
    public IActionResult Details(int id)
    {
        var game = _games.Find(g => g.Id == id);
        return View(game);
    }

    // GET: GameController/Create
    public IActionResult Create()
```

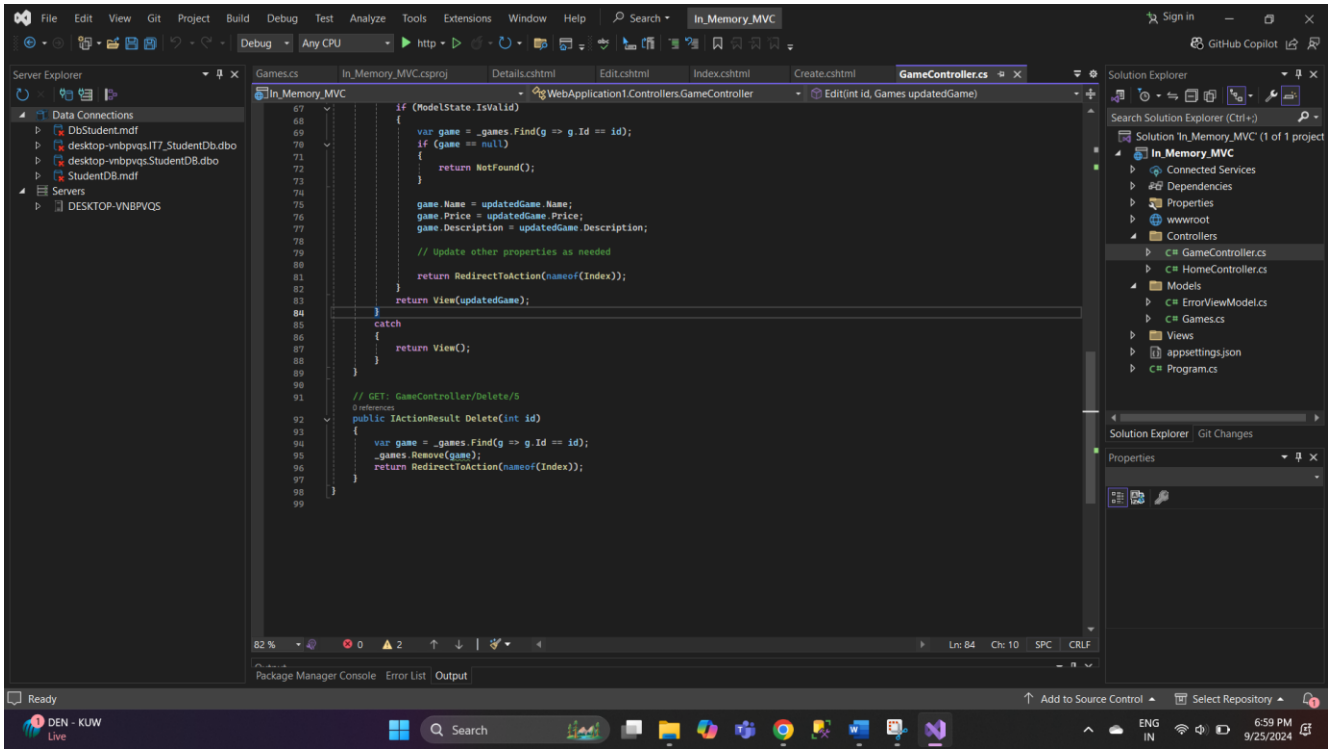
```
{
    return View();
}

// POST: GameController/Create
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(Games games)
{
    try
    {
        if (ModelState.IsValid)
        {
            _games.Add(games);
            return RedirectToAction(nameof(Index));
        }
        return View();
    }
    catch
    {
        return View();
    }
}

// GET: GameController/Edit/5
public IActionResult Edit(int id)
{
    var game = _games.Find(g => g.Id == id);
    return View(game);
}

// POST: GameController/Edit/5
[HttpPost]
```

```
[ValidateAntiForgeryToken]
public IActionResult Edit(int id, Games updatedGame)
{
    try
    {
        if (ModelState.IsValid)
        {
            var game = _games.Find(g => g.Id == id);
            if (game == null)
            {
                return NotFound();
            }

            game.Name = updatedGame.Name;
            game.Price = updatedGame.Price;
            game.Description = updatedGame.Description;

            // Update other properties as needed

            return RedirectToAction(nameof(Index));
        }
        return View(updatedGame);
    }
    catch
    {
        return View();
    }
}

// GET: GameController/Delete/5
public IActionResult Delete(int id)
{
    var game = _games.Find(g => g.Id == id);
```

```
        _games.Remove(game);
        return RedirectToAction(nameof(Index));
    }
}
```

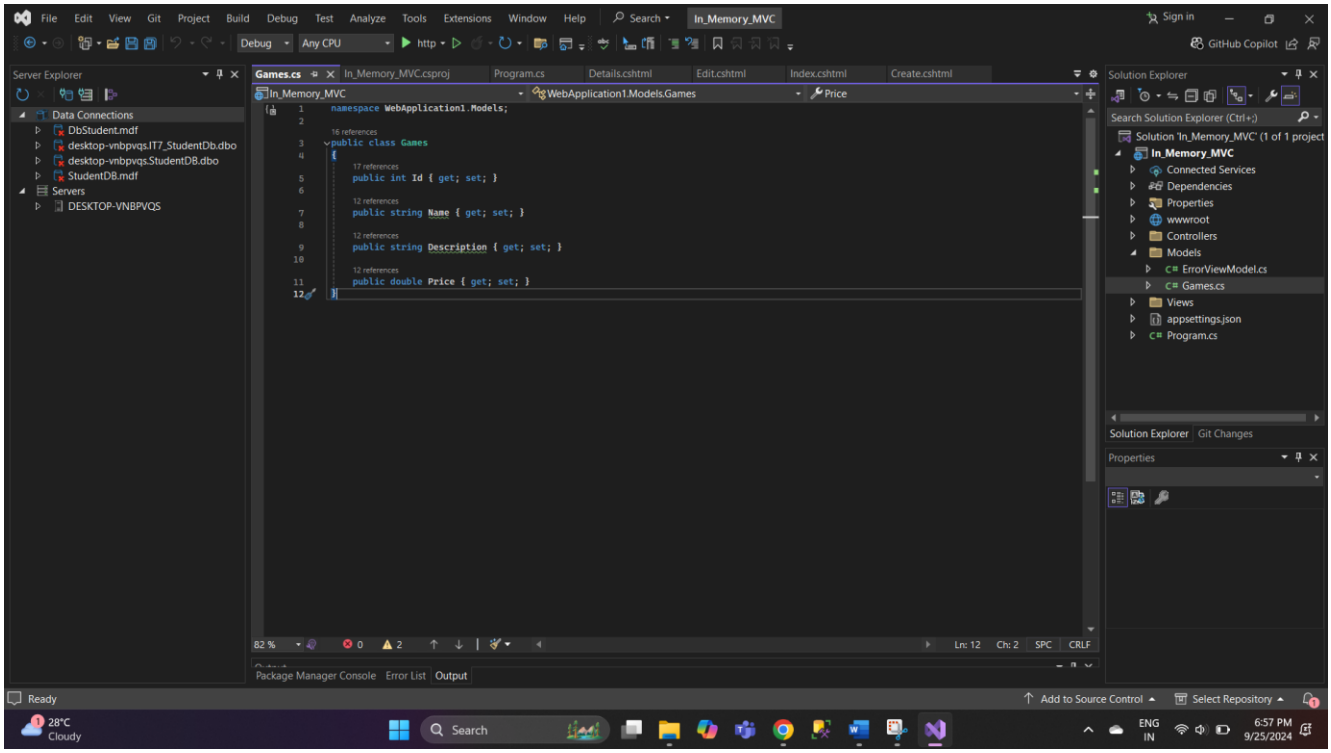




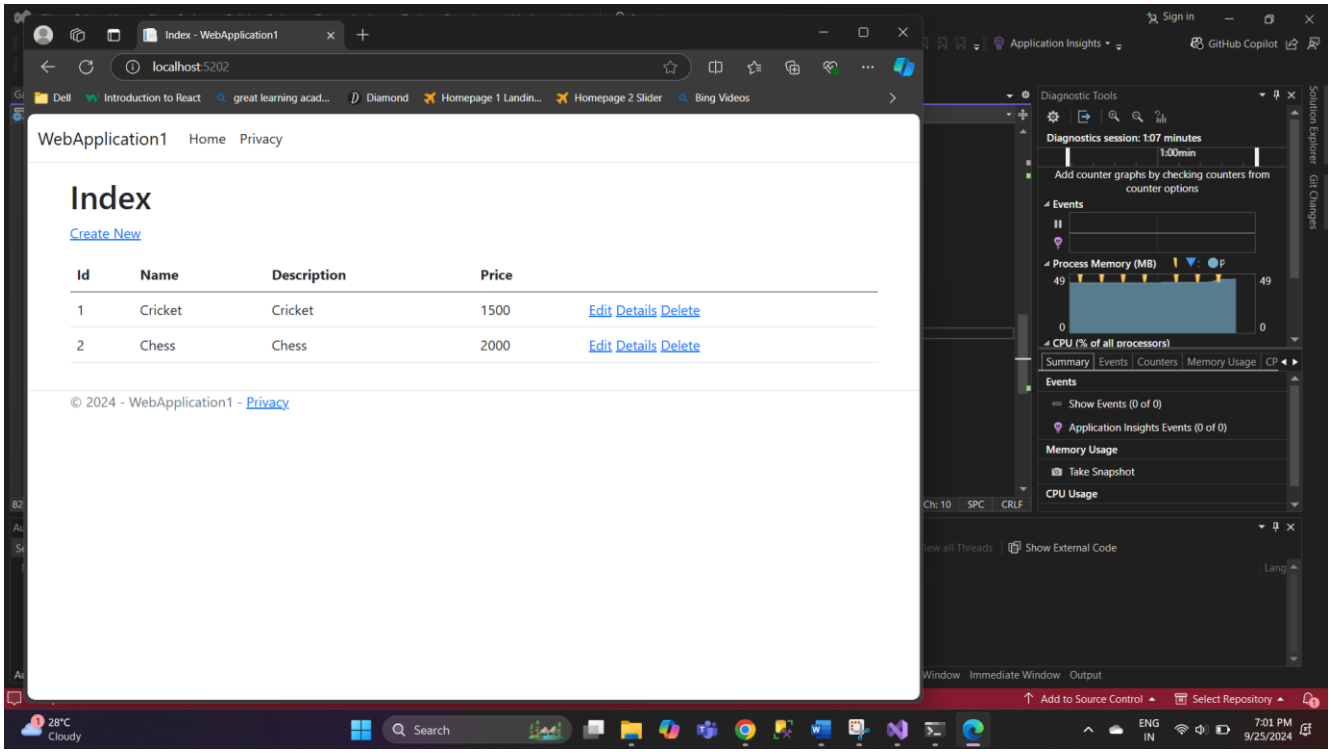


Games.cs

```
namespace WebApplication1.Models;


public class Games

{

    public int Id { get; set; }


    public string Name { get; set; }


    public string Description { get; set; }


    public double Price { get; set; }

}
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
In_Memory_MVC
Debug
Any CPU
Games.cs
In_Memory_MVC.csproj
Program.cs
Details.cshtml
Edit.cshtml
Index.cshtml
Create.cshtml
namespace WebApplication1.Models;
public class Games
{
public int Id { get; set; }
public string Name { get; set; }
public string Description { get; set; }
public double Price { get; set; }
}
Solution Explorer
In_Memory_MVC
Connected Services
Dependencies
Properties
wwwroot
Controllers
Models
ErrorViewModel.cs
Games.cs
Views
appsettings.json
Program.cs
Package Manager Console
Error List
Output
Ready
Add to Source Control
Select Repository
28°C
Cloudy
Search
ENG
IN
6:57 PM
9/25/2024

Output:-


Index - WebApplication1
localhost:5202
WebApplication1
Home
Privacy
Index
Create New
Id
Name
Description
Price
1
Cricket
Cricket
1500
Edit Details Delete
2
Chess
Chess
2000
Edit Details Delete
© 2024 - WebApplication1 - Privacy
Diagnostic Tools
Diagnostics session: 1:07 minutes
Add counter graphs by checking counters from counter options
Events
Process Memory (MB)
CPU (% of all processors)
Summary
Events
Counters
Memory Usage
Show Events (0 of 0)
Application Insights Events (0 of 0)
Memory Usage
Take Snapshot
CPU Usage
Show External Code
Add to Source Control
Select Repository
28°C
Cloudy
Search
ENG
IN
7:01 PM
9/25/2024

Q. 5. Create a C# MVC core Application to perform CRUD operations (on your choice of data) using Stack, queue or tuple.

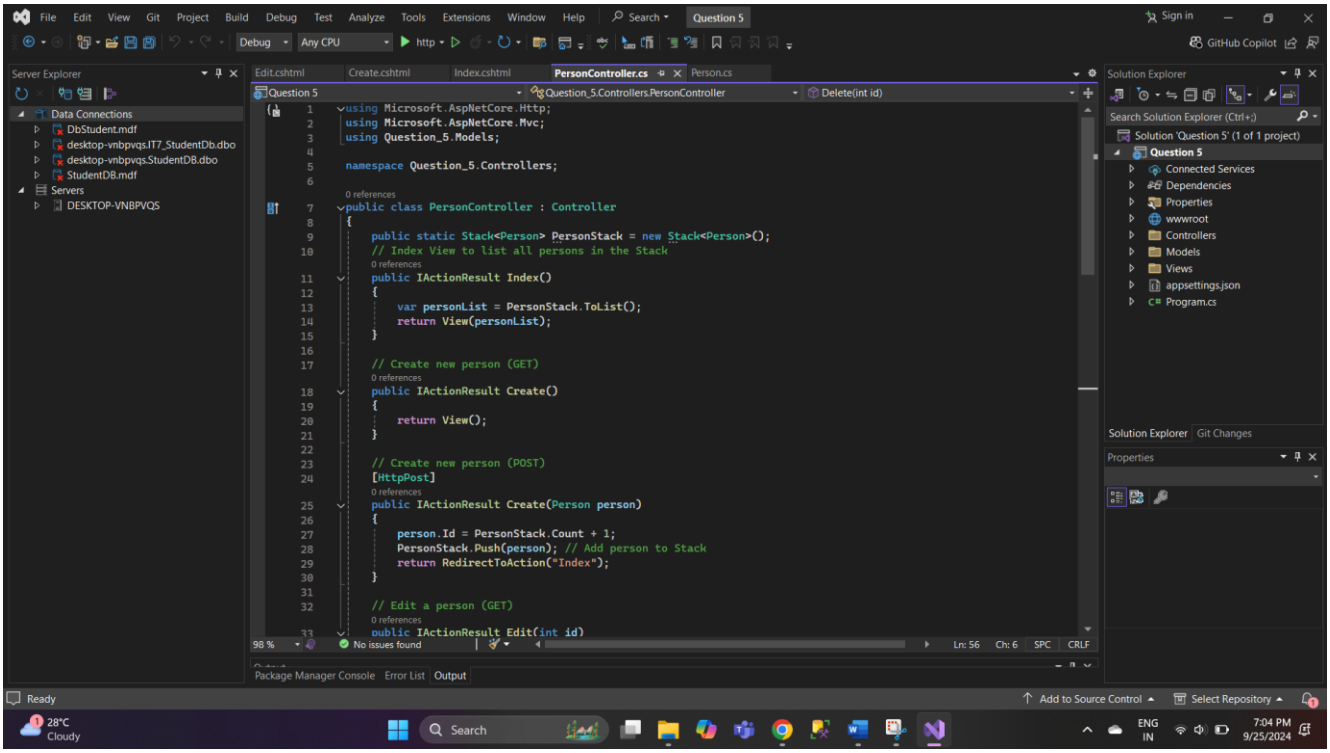
=>

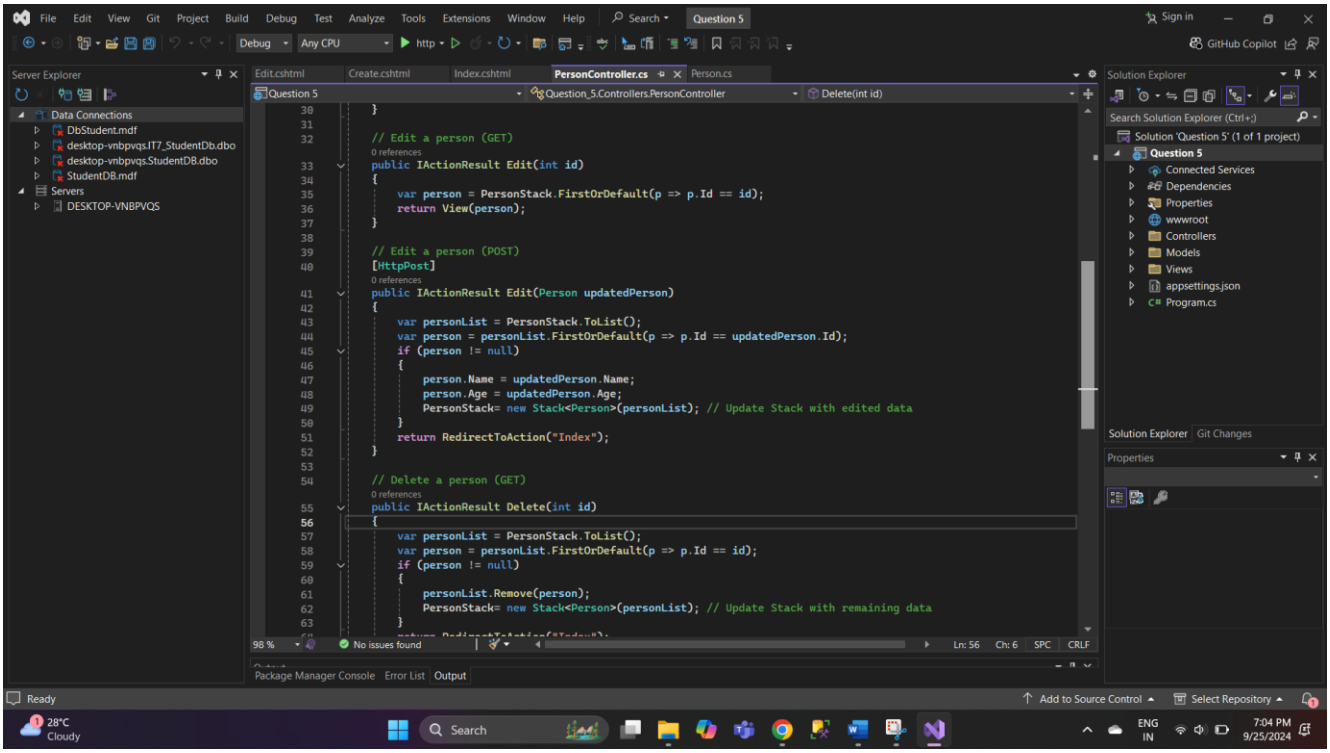
PersonController:

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Question_5.Models;

namespace Question_5.Controllers;

public class PersonController : Controller
{
    public static Stack<Person> PersonStack = new Stack<Person>();
    // Index View to list all persons in the Stack
    public IActionResult Index()
    {
        var personList = PersonStack.ToList();
        return View(personList);
    }

    // Create new person (GET)
    public IActionResult Create()
    {
        return View();
    }

    // Create new person (POST)
    [HttpPost]
    public IActionResult Create(Person person)
    {
        person.Id = PersonStack.Count + 1;
```

```
    PersonStack.Push(person); // Add person to Stack
    return RedirectToAction("Index");
}

// Edit a person (GET)
public IActionResult Edit(int id)
{
    var person = PersonStack.FirstOrDefault(p => p.Id == id);
    return View(person);
}

// Edit a person (POST)
[HttpPost]
public IActionResult Edit(Person updatedPerson)
{
    var personList = PersonStack.ToList();
    var person = personList.FirstOrDefault(p => p.Id == updatedPerson.Id);
    if (person != null)
    {
        person.Name = updatedPerson.Name;
        person.Age = updatedPerson.Age;
        PersonStack= new Stack<Person>(personList); // Update Stack with edited data
    }
    return RedirectToAction("Index");
}

// Delete a person (GET)
public IActionResult Delete(int id)
{
    var personList = PersonStack.ToList();
    var person = personList.FirstOrDefault(p => p.Id == id);
    if (person != null)
    {
```

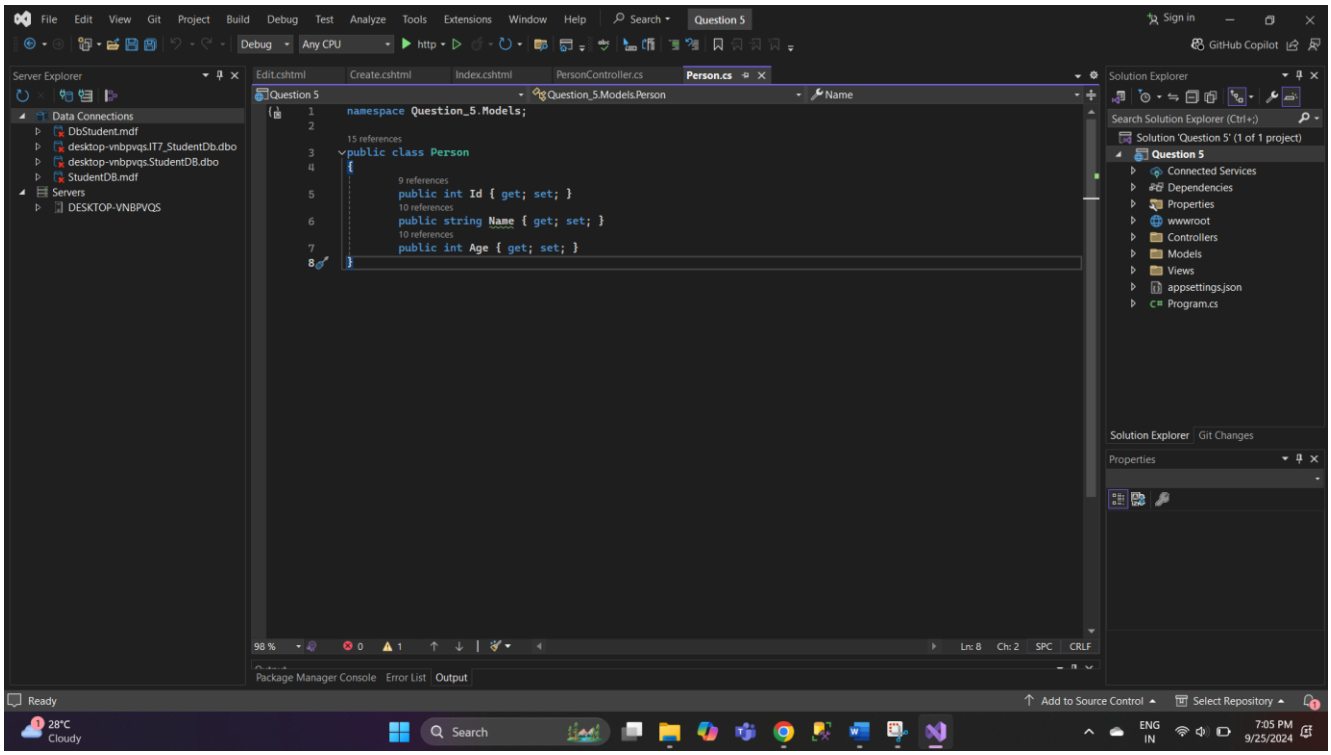
```
            personList.Remove(person);
            PersonStack= new Stack<Person>(personList); // Update Stack with remaining data
        }
        return RedirectToAction("Index");
    }
}
```



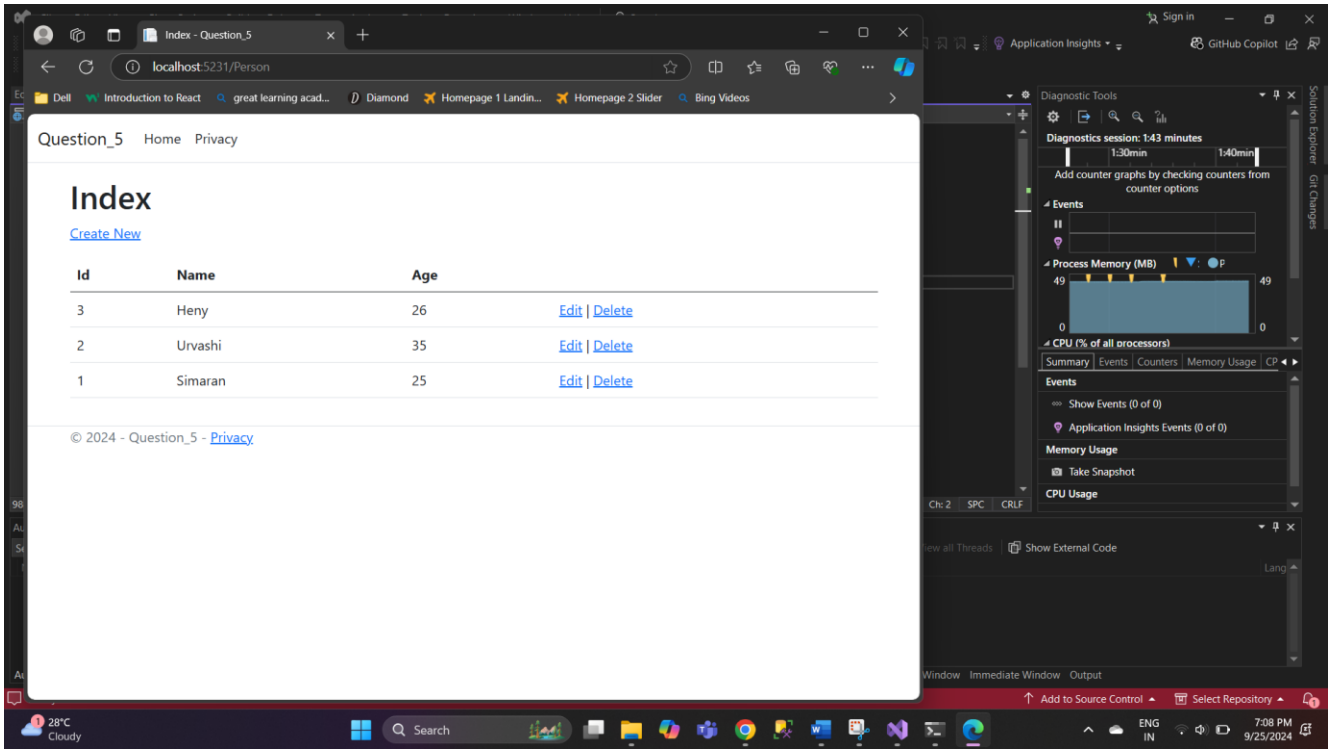


Person.cs

```
namespace Question_5.Models;


public class Person
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
}
```



Output:


Index - Question_5
localhost:5231/Person
Dell
Introduction to React
great learning acad...
Diamond
Homepage 1 Landin...
Homepage 2 Slider
Bing Videos
Question_5
Home
Privacy
Index
Create New
Id
Name
Age
3
Heny
26
Edit | Delete
2
Urvashi
35
Edit | Delete
1
Simaran
25
Edit | Delete
© 2024 - Question_5 - Privacy
Sign in
Application Insights
GitHub Copilot
Diagnostic Tools
Diagnostics session: 1:43 minutes
1:30min
1:40min
Add counter graphs by checking counters from counter options
Events
Process Memory (MB)
CPU (% of all processors)
Summary
Events
Counters
Memory Usage
Show Events (0 of 0)
Application Insights Events (0 of 0)
Memory Usage
Take Snapshot
CPU Usage
Show External Code
Window
Immediate Window
Output
Add to Source Control
Select Repository
28°C
Cloudy
Search
ENG
IN
7:08 PM
9/25/2024

Q. 6. Generate a result of student class taking marks of 3 subjects using collection and print error if the obtained marks are greater than max marks. Take student details as per you choice and max marks as user input.

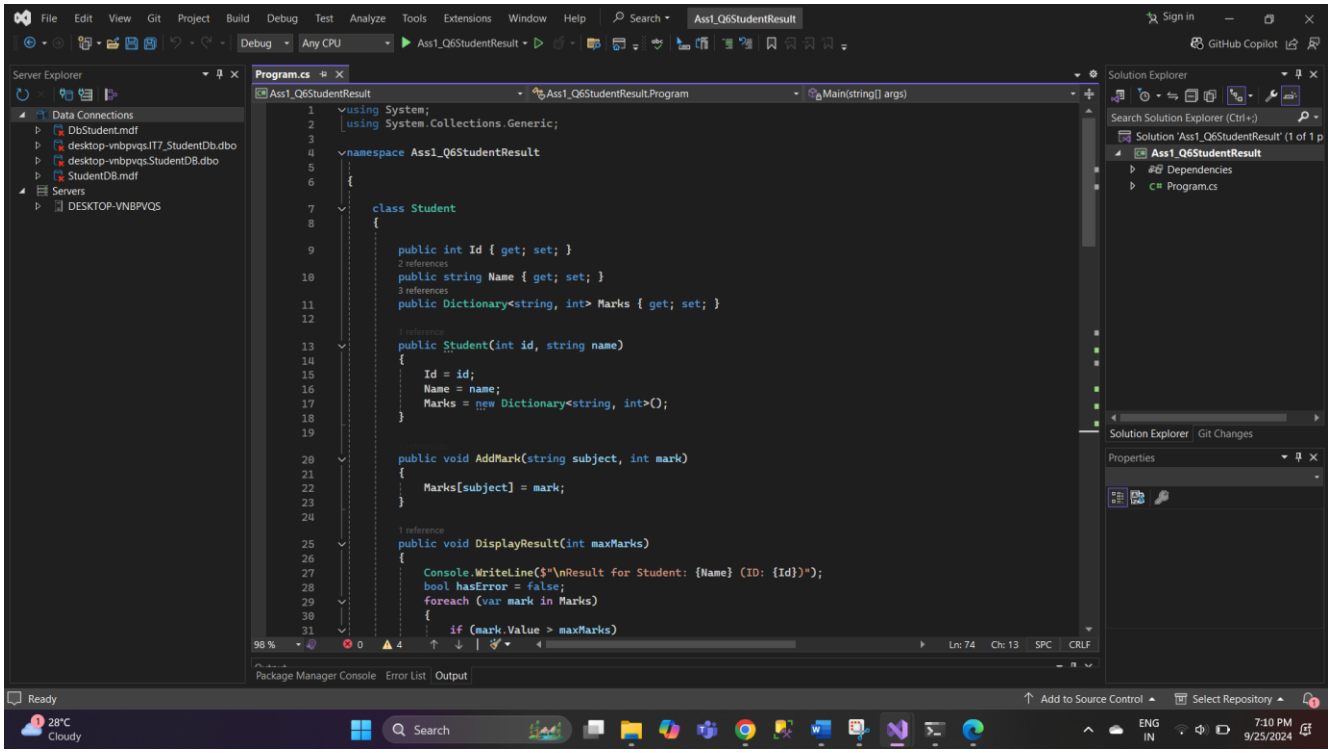
=>

Program.cs

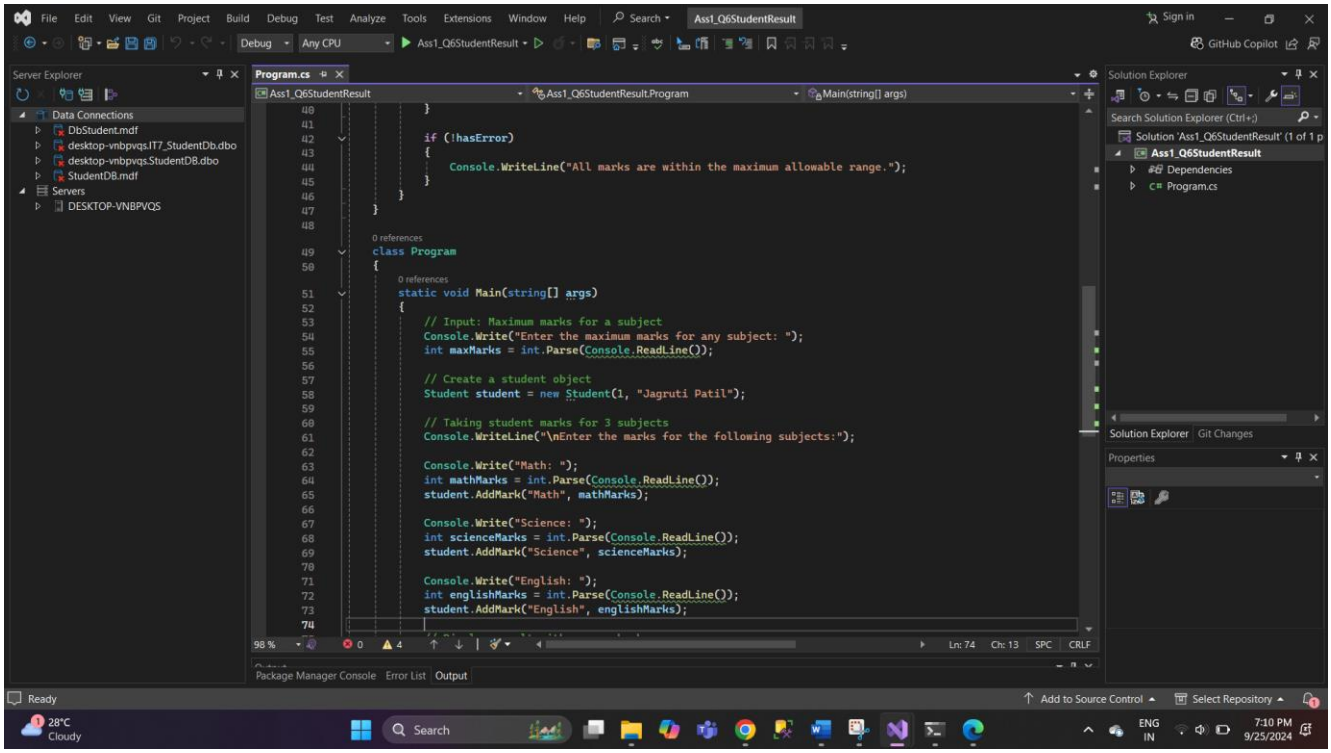
```
using System;
using System.Collections.Generic;

namespace Ass1_Q6StudentResult

{
   class Student
   {
      public int Id { get; set; }
      public string Name { get; set; }
      public Dictionary<string, int> Marks { get; set; }

      public Student(int id, string name)
      {
         Id = id;
         Name = name;
         Marks = new Dictionary<string, int>();
      }

      public void AddMark(string subject, int mark)
      {
         Marks[subject] = mark;
      }

      public void DisplayResult(int maxMarks)
      {
         Console.WriteLine($"\nResult for Student: {Name} (ID: {Id})");
```

```
        bool hasError = false;
        foreach (var mark in Marks)
        {
            if (mark.Value > maxMarks)
            {
                Console.WriteLine($"Error: Marks in {mark.Key} exceed the maximum allowable marks ({maxMarks}).
Obtained: {mark.Value}");
                hasError = true;
            }
            else
            {
                Console.WriteLine($"{mark.Key}: {mark.Value}");
            }
        }

        if (!hasError)
        {
            Console.WriteLine("All marks are within the maximum allowable range.");
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Input: Maximum marks for a subject
        Console.Write("Enter the maximum marks for any subject: ");
        int maxMarks = int.Parse(Console.ReadLine());

        // Create a student object
        Student student = new Student(1, "Jagruti Patil");
```

```
        // Taking student marks for 3 subjects
        Console.WriteLine("\nEnter the marks for the following subjects:");

        Console.Write("Math: ");
        int mathMarks = int.Parse(Console.ReadLine());
        student.AddMark("Math", mathMarks);

        Console.Write("Science: ");
        int scienceMarks = int.Parse(Console.ReadLine());
        student.AddMark("Science", scienceMarks);

        Console.Write("English: ");
        int englishMarks = int.Parse(Console.ReadLine());
        student.AddMark("English", englishMarks);

        // Display result with error check
        student.DisplayResult(maxMarks);
        Console.ReadLine();
    }
  }
}
```

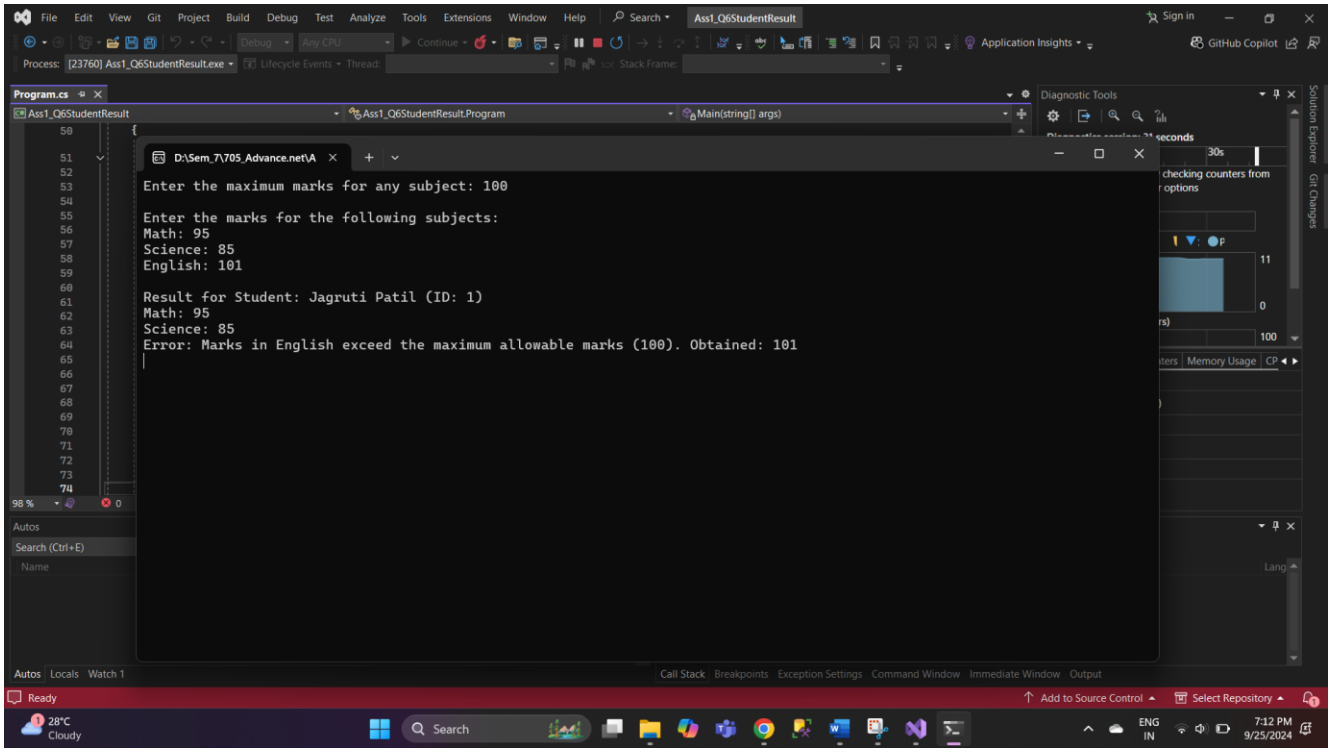


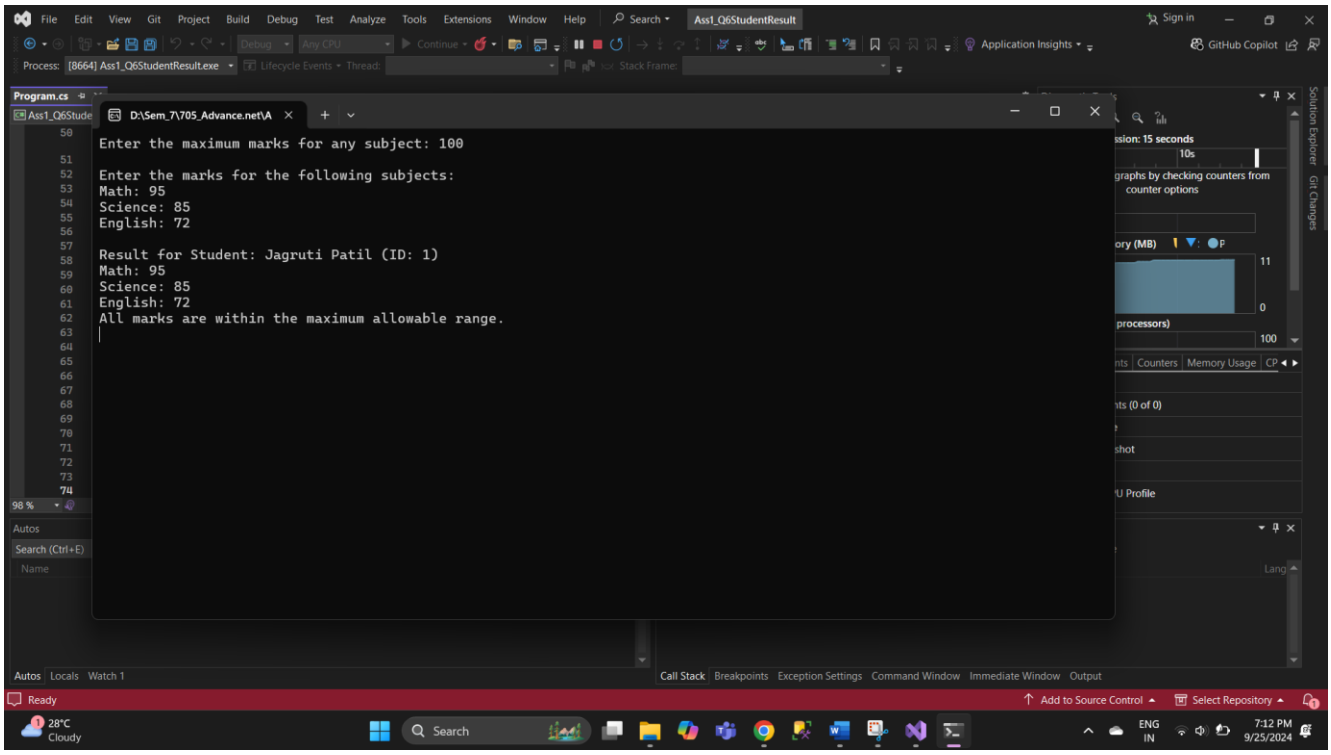
```
using System;
using System.Collections.Generic;

namespace Ass1_Q6StudentResult
{
    class Student
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public Dictionary<string, int> Marks { get; set; }

        public Student(int id, string name)
        {
            Id = id;
            Name = name;
            Marks = new Dictionary<string, int>();
        }

        public void AddMark(string subject, int mark)
        {
            Marks[subject] = mark;
        }

        public void DisplayResult(int maxMarks)
        {
            Console.WriteLine($"\nResult for Student: {Name} (ID: {Id})");
            bool hasError = false;
            foreach (var mark in Marks)
            {
                if (mark.Value > maxMarks)
```



```
            }

            if (!hasError)
            {
                Console.WriteLine("All marks are within the maximum allowable range.");
            }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // Input: Maximum marks for a subject
            Console.Write("Enter the maximum marks for any subject: ");
            int maxMarks = int.Parse(Console.ReadLine());

            // Create a student object
            Student student = new Student(1, "Jagruti Patil");

            // Taking student marks for 3 subjects
            Console.WriteLine("\nEnter the marks for the following subjects:");

            Console.Write("Math: ");
            int mathMarks = int.Parse(Console.ReadLine());
            student.AddMark("Math", mathMarks);

            Console.Write("Science: ");
            int scienceMarks = int.Parse(Console.ReadLine());
            student.AddMark("Science", scienceMarks);

            Console.Write("English: ");
            int englishMarks = int.Parse(Console.ReadLine());
            student.AddMark("English", englishMarks);
```

Output:


D:\Sem_7\705_Advance.net\A
Enter the maximum marks for any subject: 100
Enter the marks for the following subjects:
Math: 95
Science: 85
English: 101
Result for Student: Jagruti Patil (ID: 1)
Math: 95
Science: 85
Error: Marks in English exceed the maximum allowable marks (100). Obtained: 101


D:\Sem_7\705_Advance.net\A
Enter the maximum marks for any subject: 100
Enter the marks for the following subjects:
Math: 95
Science: 85
English: 72
Result for Student: Jagruti Patil (ID: 1)
Math: 95
Science: 85
English: 72
All marks are within the maximum allowable range.