

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;

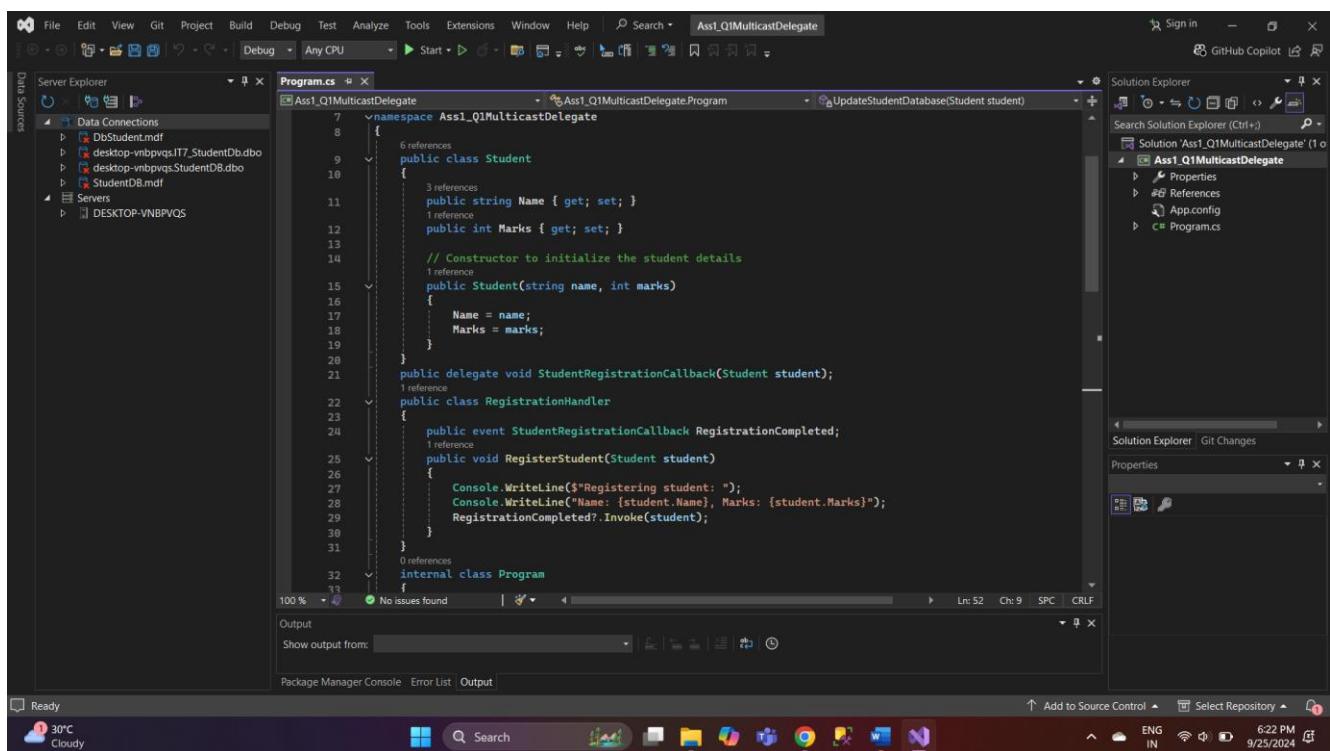
    var student = new Student("Aditi Jain", 75);

    registrationHandler.RegisterStudent(student);

    Console.WriteLine("Press any key to exit...");

    Console.ReadKey();
}
}

```



The screenshot shows the Microsoft Visual Studio IDE interface. The code editor displays the `Program.cs` file for the project `Ass1_Q1MulticastDelegate`. The code implements a multicast delegate pattern for student registration and database update. The Solution Explorer shows the project structure with files like `Properties`, `References`, `App.config`, and `Program.cs`. The Task List shows no issues found.

```
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();
    }
}
```

Output :-

The screenshot shows the Microsoft Visual Studio IDE interface with the output window open, displaying the execution results of the application. The application registered a student named "Aditi Jain" with marks 75, sent a welcome email, updated the database, and prompted the user to press any key to exit. A performance monitor window is also visible on the right side of the interface.

```
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...
```

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();

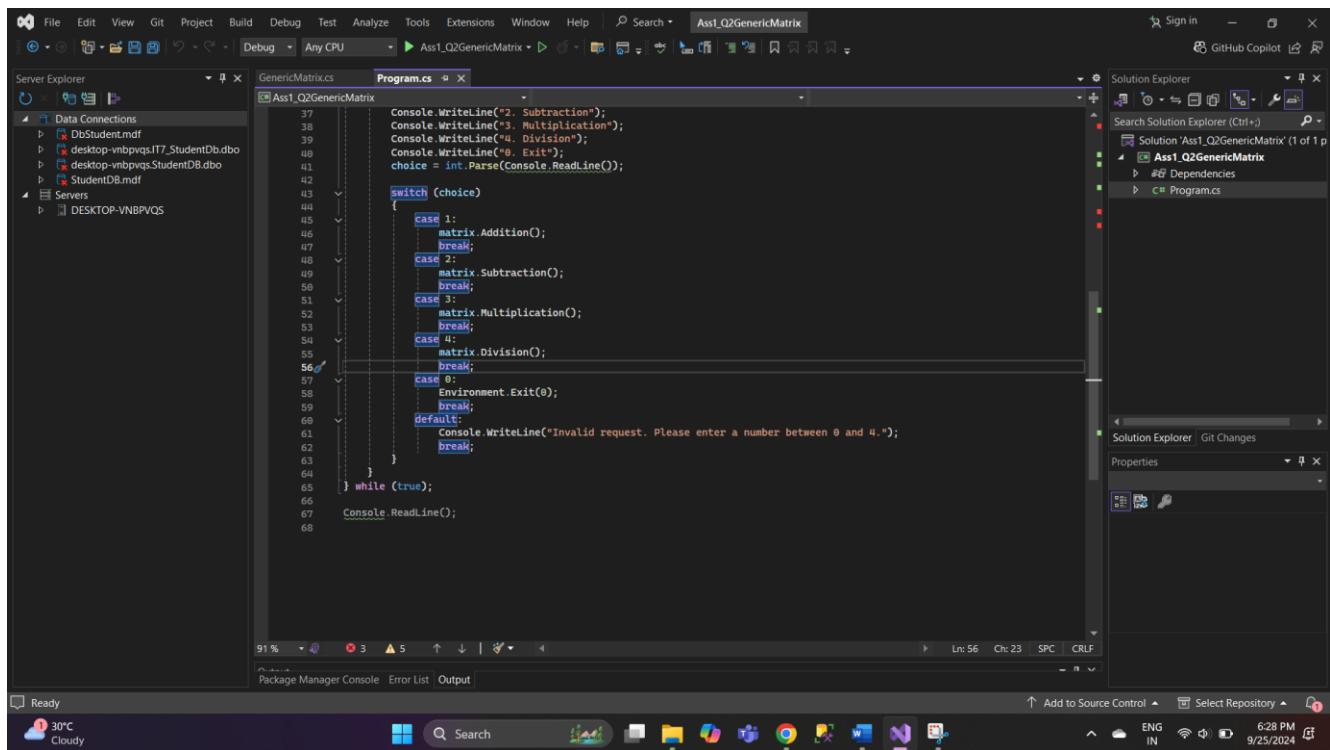
```

The screenshot shows the Microsoft Visual Studio interface. The code editor displays a C# program named Program.cs. The code handles matrix operations and a menu system. The Solution Explorer shows a solution named 'Ass1_Q2GenericMatrix' with a single project 'Ass1_Q2GenericMatrix' containing 'Program.cs'. The Taskbar at the bottom shows various open applications including a weather widget (30°C Cloudy), a search bar, and other development tools.

```

1  using Ass1.Q2GenericMatrix;
2
3  int row, col, choice, type;
4  dynamic matrix;
5
6  Console.WriteLine("Enter the number of rows:");
7  row = int.Parse(Console.ReadLine());
8  Console.WriteLine("Enter the number of columns:");
9  col = int.Parse(Console.ReadLine());
10
11 do
12 {
13     Console.WriteLine("Select type:\n1. Integer\n2. Float");
14     type = int.Parse(Console.ReadLine());
15
16     switch (type)
17     {
18         case 1:
19             matrix = new GenericMatrix<int>(row, col);
20             break;
21         case 2:
22             matrix = new GenericMatrix<float>(row, col);
23             break;
24         default:
25             Console.WriteLine("Invalid input. Please enter 1 or 2.");
26             continue;
27     }
28
29     matrix.SetMatrix();
30     matrix.GetMatrix();
31
32     bool menu = false;
33     while (!menu)
34     {
35         Console.WriteLine("1. Addition");
36         Console.WriteLine("2. Subtraction");
37         Console.WriteLine("3. Multiplication");
38         Console.WriteLine("4. Division");
39         Console.WriteLine("9. Exit");
40     }
}

```



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
{
    public class GenericMatrix<T>
    {
        public readonly int _Row;
        public readonly int _Col;
        private T[,] matrix1;
        private T[,] matrix2;
        private T[,] matrix3;
        public GenericMatrix(int row, int col)
        {
            _Row = row;
            _Col = col;
            matrix1 = new T[row, col];
            matrix2 = new T[row, col];
            matrix3 = new T[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($"Enter element of matrix [{i},{j}]:");
                    matrix1[i, j] = (TypeConvert.ChangeType(Console.ReadLine(), typeof(T)));
                }
            Console.WriteLine("Second Matrix");
            for (int i = 0; i < _Row; i++)
                for (int j = 0; j < _Col; j++)
                {
                    Console.WriteLine($"Enter element of matrix [{i},{j}]:");
                    matrix2[i, j] = (TypeConvert.ChangeType(Console.ReadLine(), typeof(T)));
                }
        }
        public void GetMatrix()
        {
        }
        public T Division(Type a, Type b)
        {
            if (b == 0)
                throw new DivideByZeroException();
            return a / b;
        }
    }
}
```

```
namespace Ass1_Q2GenericMatrix
{
    public class GenericMatrix<T>
    {
        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 T Addition(Type a, Type b)
        {
            return a + b;
        }
    }
}
```

The screenshot shows the Microsoft Visual Studio IDE interface. The code editor displays two files: `GenericMatrix.cs` and `Program.cs`. The `GenericMatrix.cs` file contains methods for addition, subtraction, multiplication, and division of generic matrices. The `Program.cs` file contains the main application logic, including matrix creation and operation selection. The Solution Explorer pane shows the project structure with files `Ass1_Q2GenericMatrix.cs`, `GenericMatrix.cs`, and `Program.cs`. The Task List pane is visible at the bottom.

```
91    {
92        for (int j = 0; j < _Col; j++)
93            Console.WriteLine(matrix3[i, j] + "\t");
94        Console.WriteLine("\n");
95    }
96}
97
98    i reference
99    private Type Subtraction(Type a, Type b)
100   {
101       dynamic x = a;
102       dynamic y = b;
103       return x - y;
104   }
105
106    0 references
107    public void Multiplication()
108   {
109       for (int i = 0; i < _Row; i++)
110           for (int j = 0; j < _Col; j++)
111               matrix3[i, j] = Multiplication(matrix1[i, j], matrix2[i, j]);
112
113       Console.WriteLine("New Array");
114       for (int i = 0; i < _Row; i++)
115       {
116           for (int j = 0; j < _Col; j++)
117               Console.WriteLine(matrix3[i, j] + "\t");
118           Console.WriteLine("\n");
119       }
120
121    i reference
122    private Type Multiplication(Type a, Type b)
123   {
124       dynamic x = a;
125       dynamic y = b;
126       return x * y;
127   }
128
129    0 references
130    public void Division()
131   {
132       for (int i = 0; i < _Row; i++)
133       {
134           for (int j = 0; j < _Col; j++)
135               matrix3[i, j] = Division(matrix1[i, j], matrix2[i, j]);
136
137           Console.WriteLine("New Array");
138           for (int i = 0; i < _Row; i++)
139           {
140               for (int j = 0; j < _Col; j++)
141                   Console.WriteLine(matrix3[i, j] + "\t");
142               Console.WriteLine("\n");
143           }
144       }
145   }
```

Output:-

The screenshot shows a terminal window titled `D:\Sem_7\705_Advance.net\A`. The program interacts with the user to create two matrices and perform operations on them. The output shows the input steps, the creation of the first matrix, the second matrix, and the final result of the multiplication operation.

```
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]:
88
Enter element of matrix [1,0]:
2
First Matrix
87
6
Second Matrix
88
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
|
```

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}";
        }
    }
}
```



```
        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.WriteLine("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();
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;

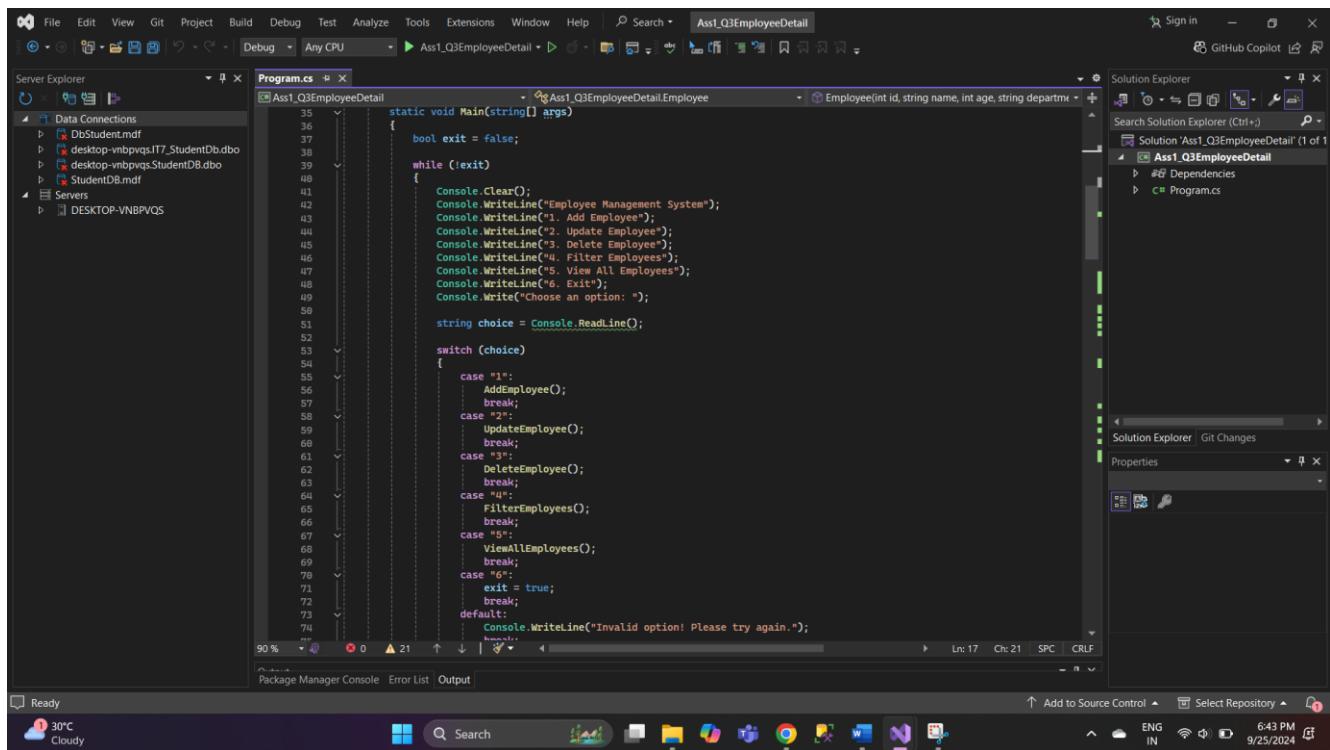
namespace Ass1_Q3EmployeeDetail
{
    // Define the Employee class
    public class Employee
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }
        public string Department { get; set; }

        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}";
        }
    }

    // Generic collection to store employee data
    static List<Employee> employees = new List<Employee>();
    static int employeeIdCounter = 1; // Auto-increment for employee IDs
}

```



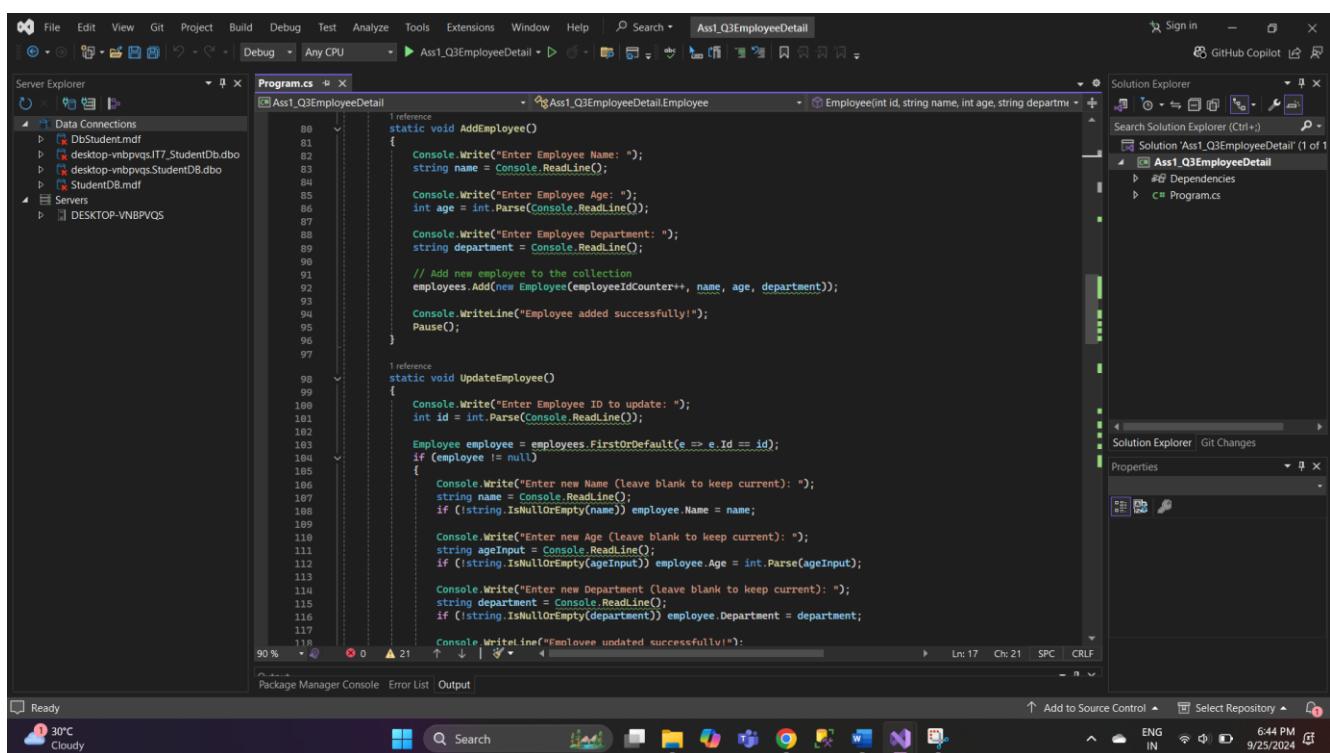
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search Sign in GitHub Copilot

Program.cs

```
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.");
        }
    }
}
```

Ready 30°C Cloudy 643 PM 9/25/2024



File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search Sign in GitHub Copilot

Program.cs

```
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!");
}
```

Ready 30°C Cloudy 644 PM 9/25/2024

```
125 }  
126  
127     static void DeleteEmployee()  
128     {  
129         Console.WriteLine("Enter Employee ID to delete: ");  
130         int id = int.Parse(Console.ReadLine());  
131  
132         Employee employee = employees.FirstOrDefault(e => e.Id == id);  
133         if (employee != null)  
134         {  
135             employees.Remove(employee);  
136             Console.WriteLine("Employee deleted successfully!");  
137         }  
138         else  
139         {  
140             Console.WriteLine("Employee not found.");  
141         }  
142         Pause();  
143     }  
144  
145     static void FilterEmployees()  
146     {  
147         Console.WriteLine("Filter Options:");  
148         Console.WriteLine("1 By Name");  
149         Console.WriteLine("2 By Age");  
150         Console.WriteLine("3 By Department");  
151         Console.WriteLine("4 Multiple Criteria");  
152         Console.Write("Choose a filter option: ");  
153  
154         string choice = Console.ReadLine();  
155         Ienumerable<Employee> filteredEmployees = employees;  
156  
157         switch (choice)  
158         {  
159             case "1":  
160                 Console.WriteLine("Enter Name to filter by: ");  
161                 string name = Console.ReadLine();  
162                 filteredEmployees = employees.Where(e => e.Name.Contains(name, StringComparison.OrdinalIgnoreCase));  
163             break;  
164  
165             case "2":  
166                 Console.WriteLine("Enter Age to filter by: ");  
167                 string multiAgeInput = Console.ReadLine();  
168                 if (!string.IsNullOrEmpty(multiAgeInput) && int.TryParse(multiAgeInput, out int multiAge))  
169                 {  
170                     filteredEmployees = filteredEmployees.Where(e => e.Age == multiAge);  
171                 }  
172  
173             case "3":  
174                 Console.WriteLine("Enter Department to filter by: ");  
175                 string multiDept = Console.ReadLine();  
176                 if (!string.IsNullOrEmpty(multiDept))  
177                 {  
178                     filteredEmployees = filteredEmployees.Where(e => e.Department.Equals(multiDept, StringComparison.OrdinalIgnoreCase));  
179                 }  
180             break;  
181  
182             default:  
183                 Console.WriteLine("Invalid option.");  
184             break;  
185         }  
186  
187         Console.WriteLine("\nFiltered Employees:");  
188         foreach (var emp in filteredEmployees)  
189         {  
190             Console.WriteLine(emp);  
191         }  
192         Pause();  
193     }  
194  
195     static void ViewAllEmployees()  
196     {  
197         Console.WriteLine("\nAll Employees:");  
198         foreach (var employee in employees)  
199         {  
200             Console.WriteLine(employee);  
201         }  
202         Pause();  
203     }  
204  
205     static void Pause()  
206     {  
207         Console.WriteLine("Press any key to continue...");  
208         Console.ReadKey();  
209     }  
210  
211     static void Main(string[] args)  
212     {  
213         Program p = new Program();  
214         p.FilterEmployees();  
215     }  
216 }  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226
```

```
189         if (!string.IsNullOrEmpty(multiAgeInput) && int.TryParse(multiAgeInput, out int multiAge))  
190         {  
191             filteredEmployees = filteredEmployees.Where(e => e.Age == multiAge);  
192         }  
193  
194         if (!string.IsNullOrEmpty(multiDept))  
195         {  
196             filteredEmployees = filteredEmployees.Where(e => e.Department.Equals(multiDept, StringComparison.OrdinalIgnoreCase));  
197         }  
198         break;  
199  
200         default:  
201             Console.WriteLine("Invalid option.");  
202         break;  
203     }  
204  
205     Console.WriteLine("\nFiltered Employees:");  
206     foreach (var emp in filteredEmployees)  
207     {  
208         Console.WriteLine(emp);  
209     }  
210     Pause();  
211 }  
212  
213 static void ViewAllEmployees()  
214 {  
215     Console.WriteLine("\nAll Employees:");  
216     foreach (var employee in employees)  
217     {  
218         Console.WriteLine(employee);  
219     }  
220     Pause();  
221 }  
222  
223 static void Pause()  
224 {  
225     Console.WriteLine("Press any key to continue...");  
226     Console.ReadKey();  
227 }
```

Output:-

```
D:\Sem_7\705_Advance.netA
```

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 6:51 PM IN 9/25/2024
```

Document1 - Word

File Home Insert Draw Design References Mailings Review View Help

Comments Editing Share

Normal No Spacing Heading 1 Heading 2 Title

Find Replace Select Add-ins

Clipboard

D:\Sem_7\705_Advance.netA

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

30°C Cloudy 6:52 PM IN 9/25/2024

```
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...
```

```
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));
}

}

```

```

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

namespace WebApplication1.Controllers;

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

    public GameController()
    {
    }

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

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

    // GET: GameController/Create
    public ActionResult Create()
    {
        return View();
    }

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

```

```

if (ModelState.IsValid)
{
    _games.Add(game);
    return RedirectToAction(nameof(Index));
}
return View();
}

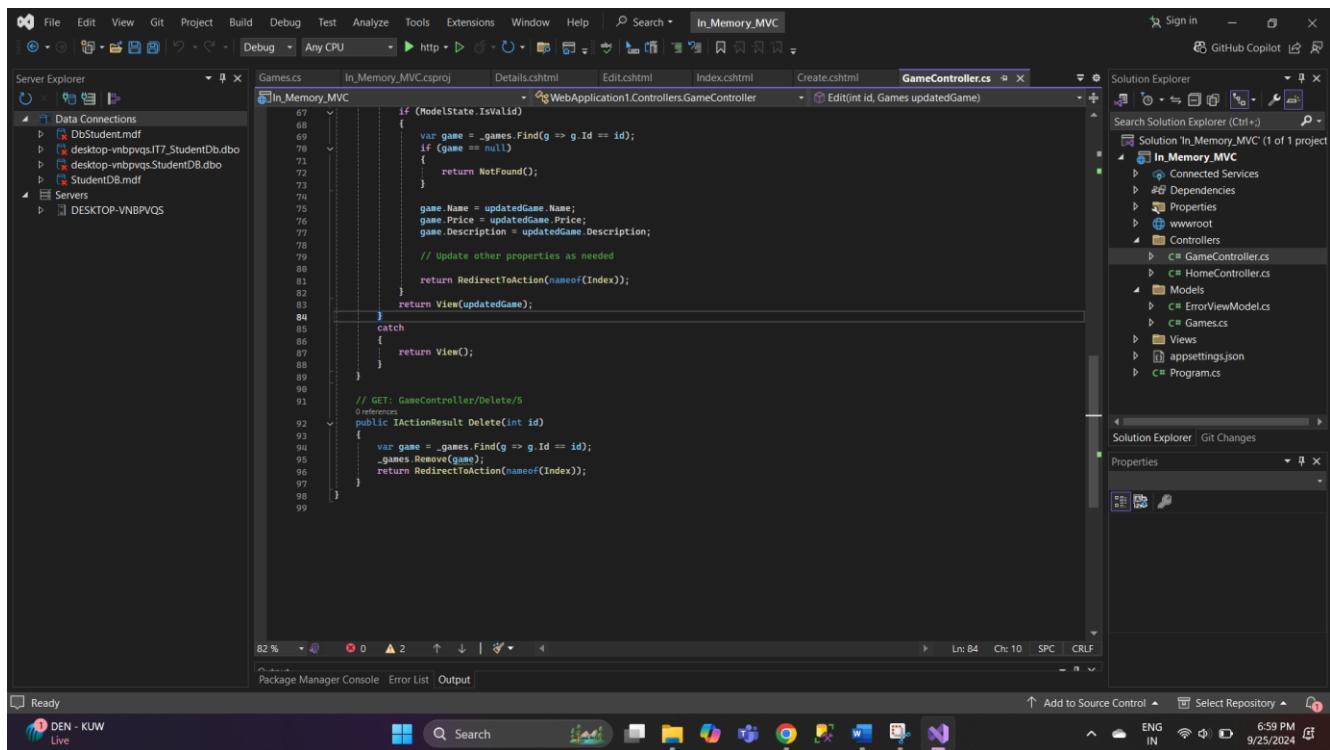
// GET: GameController/Edit/5
public ActionResult Edit(int id)
{
    var game = _games.Find(g => g.Id == id);
    return View(game);
}

// POST: GameController/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(int id, Game 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));
        }
    }
    catch
    {
        return View();
    }
}

```



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; }
```

```
}
```

The screenshot shows the Visual Studio IDE interface. The code editor displays the `Games.cs` file from the `In_Memory_MVC` project. The file contains the following C# code:

```
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; }
}
```

The Solution Explorer window on the right shows the project structure for `In_Memory_MVC`, including files like `Program.cs`, `Index.cshtml`, `Create.cshtml`, `Details.cshtml`, `ErrorViewModel.cs`, `Games.cs`, `Views`, `appsettings.json`, and `Program.cs`. The Task List at the bottom indicates no tasks are present.

Output:-

The screenshot shows a web browser displaying the `Index` page of the `WebApplication1` application. The URL is `localhost:5202`. The page content includes:

- A header with `WebApplication1`, `Home`, and `Privacy` links.
- A title `Index`.
- A link `Create New`.
- A table with the following data:

ID	Name	Description	Price	Action
1	Cricket	Cricket	1500	Edit Details Delete
2	Chess	Chess	2000	Edit Details Delete

At the bottom, there is a copyright notice: `© 2024 - WebApplication1 - Privacy`. To the right of the browser window, the Visual Studio Diagnostic Tools window is open, showing a memory usage graph for a process. The graph has a peak value of 49 MB. The window also includes tabs for Summary, Events, Counters, and Memory Usage.

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);

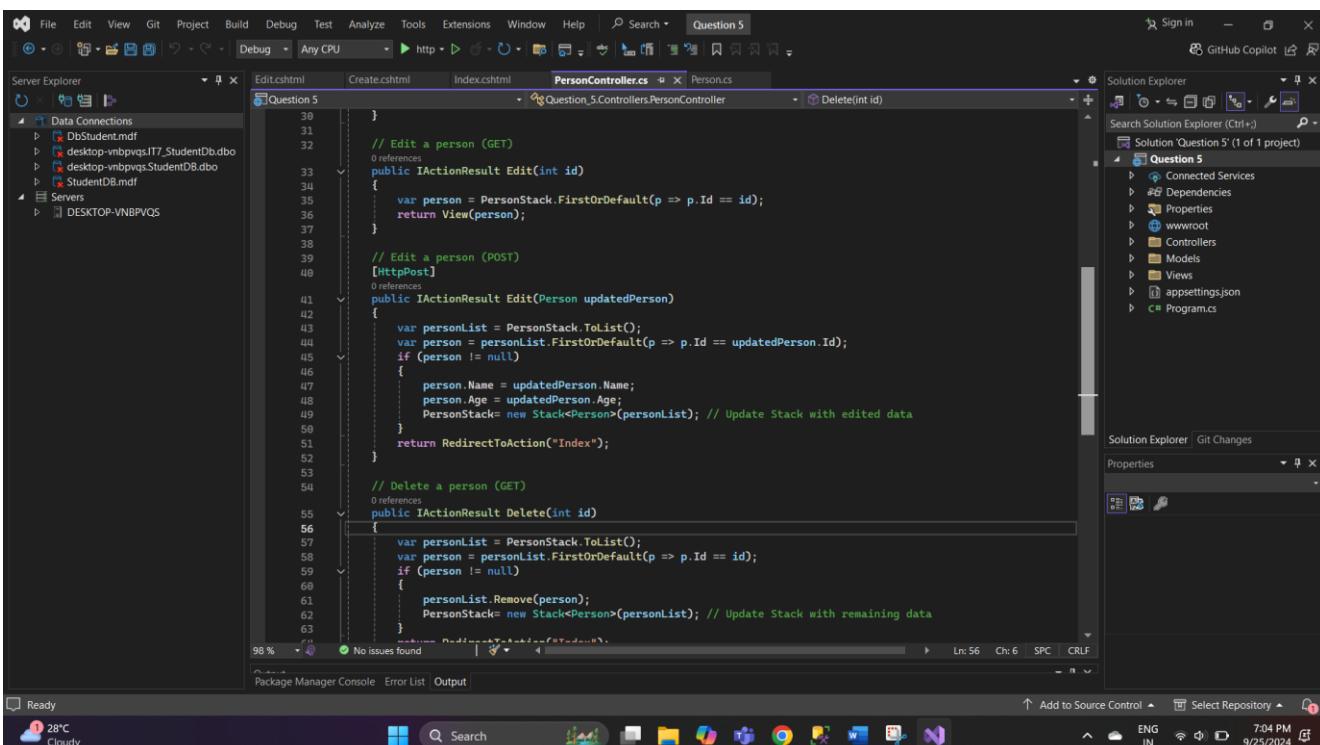
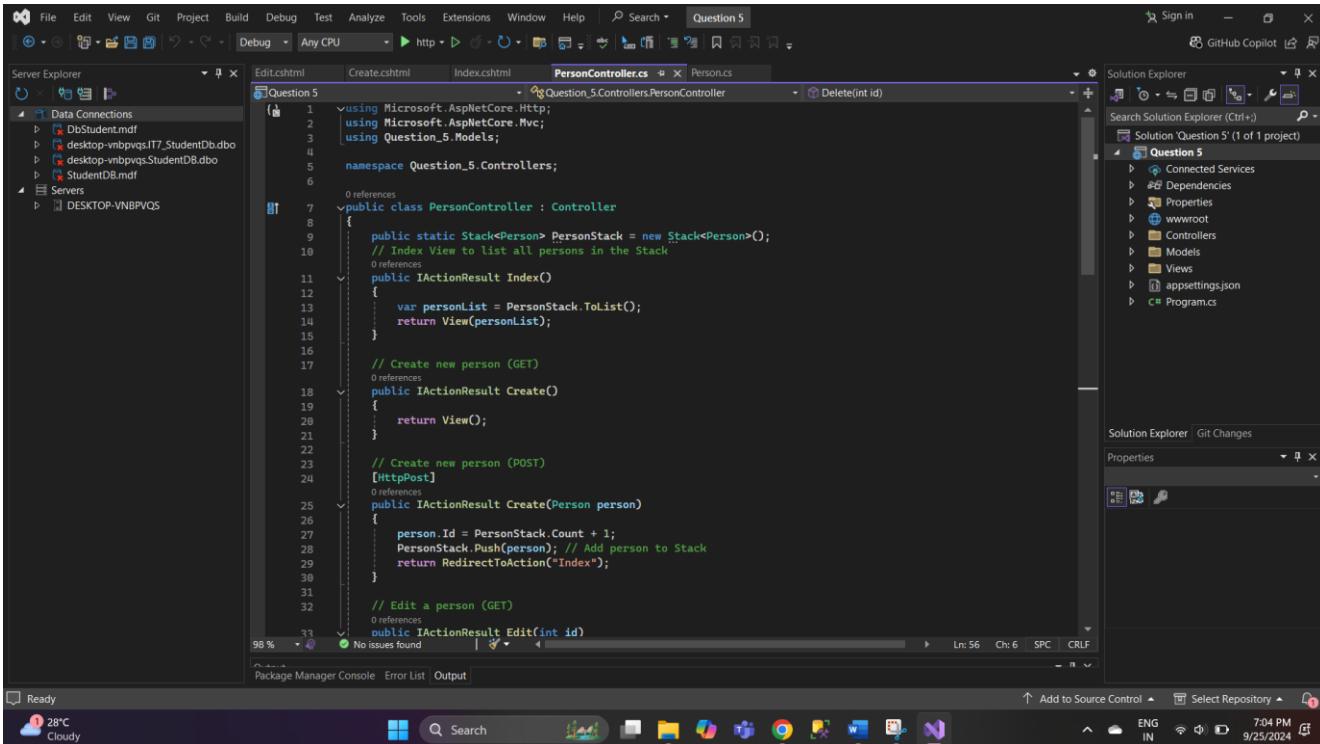
    PersonStack= new Stack<Person>(personList); // Update Stack with remaining data

}

return RedirectToAction("Index");

}

```



Person.cs

```
namespace Question_5.Models;
```

```
public class Person
```

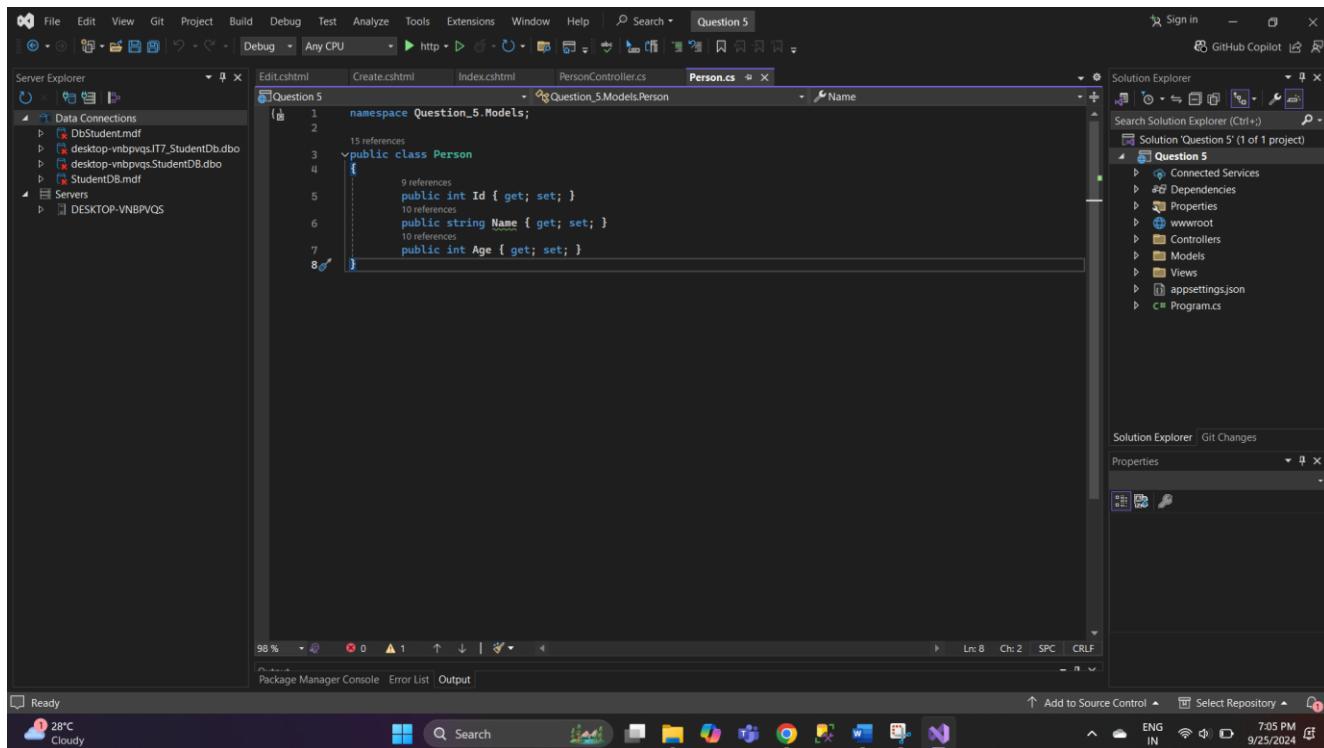
```
{
```

```
    public int Id { get; set; }
```

```
    public string Name { get; set; }
```

```
    public int Age { get; set; }
```

```
}
```



Output:

The screenshot shows a Windows desktop environment. In the center, a Microsoft Edge browser window displays a local application at localhost:5231/Person. The page title is "Index - Question_5". The content shows a table with three rows of data:

Id	Name	Age	
3	Heny	26	Edit Delete
2	Urvashi	35	Edit Delete
1	Simaran	25	Edit Delete

Below the table, there is a copyright notice: "© 2024 - Question_5 - [Privacy](#)".

To the right of the browser window, the Visual Studio Diagnostic Tools window is open. It shows a timeline from 1:30min to 1:40min. A graph for "Process Memory (MB)" shows values fluctuating between 49 and 50 MB. Below the graph, the "CPU (% of all processors)" section shows a value of 0%. The Diagnostic Tools interface includes tabs for Summary, Events, Counters, Memory Usage, and CPU.

The taskbar at the bottom of the screen shows various pinned icons, including File Explorer, Task View, Search, and several Microsoft applications like Word, Excel, and Power BI. The system tray indicates the date and time as 9/25/2024, 7:08 PM, and the weather as 28°C Cloudy.

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();
}

}

}
```

The screenshot shows the Visual Studio IDE interface. The main window displays the `Program.cs` file for the `Ass1_Q6StudentResult` project. The code defines a `Student` class with properties for ID, Name, and Marks (a dictionary of subject and marks). It includes methods to add marks and display results. The Solution Explorer on the right shows the project structure with files `Ass1_Q6StudentResult.cs`, `Dependencies`, and `Program.cs`. The status bar at the bottom indicates the date and time as 9/25/2024 7:10 PM.

```
1 using System;
2 using System.Collections.Generic;
3
4 namespace Ass1_Q6StudentResult
5 {
6     class Student
7     {
8         public int Id { get; set; }
9         public string Name { get; set; }
10        public Dictionary<string, int> Marks { get; set; }
11
12        public Student(int id, string name)
13        {
14            Id = id;
15            Name = name;
16            Marks = new Dictionary<string, int>();
17        }
18
19        public void AddMark(string subject, int mark)
20        {
21            Marks[subject] = mark;
22        }
23
24        public void DisplayResult(int maxMarks)
25        {
26            Console.WriteLine($"\\nResult for Student: {Name} (ID: {Id})");
27            bool hasError = false;
28            foreach (var mark in Marks)
29            {
30                if (mark.Value > maxMarks)
31                {
32                    hasError = true;
33                }
34            }
35            if (!hasError)
36            {
37                Console.WriteLine("All marks are within the maximum allowable range.");
38            }
39        }
40    }
41
42    class Program
43    {
44        static void Main(string[] args)
45        {
46            // Input: Maximum marks for a subject
47            Console.Write("Enter the maximum marks for any subject: ");
48            int maxMarks = int.Parse(Console.ReadLine());
49
50            // Create a student object
51            Student student = new Student(1, "Jagriti Patil");
52
53            // Taking student marks for 3 subjects
54            Console.WriteLine("\\nEnter the marks for the following subjects:");
55
56            Console.Write("Math: ");
57            int mathMarks = int.Parse(Console.ReadLine());
58            student.AddMark("Math", mathMarks);
59
60            Console.Write("Science: ");
61            int scienceMarks = int.Parse(Console.ReadLine());
62            student.AddMark("Science", scienceMarks);
63
64            Console.Write("English: ");
65            int englishMarks = int.Parse(Console.ReadLine());
66            student.AddMark("English", englishMarks);
67
68            student.DisplayResult(maxMarks);
69        }
70    }
71
72    static void Main(string[] args)
73    {
74    }
75}
```

This screenshot shows the Visual Studio IDE with the completed `Program.cs` code. The code now includes logic to read maximum marks from the user, create a student object, and take marks for three subjects (Math, Science, English) before displaying the result. The Solution Explorer and status bar remain the same as in the previous screenshot.

```
1 using System;
2 using System.Collections.Generic;
3
4 namespace Ass1_Q6StudentResult
5 {
6     class Student
7     {
8         public int Id { get; set; }
9         public string Name { get; set; }
10        public Dictionary<string, int> Marks { get; set; }
11
12        public Student(int id, string name)
13        {
14            Id = id;
15            Name = name;
16            Marks = new Dictionary<string, int>();
17        }
18
19        public void AddMark(string subject, int mark)
20        {
21            Marks[subject] = mark;
22        }
23
24        public void DisplayResult(int maxMarks)
25        {
26            Console.WriteLine($"\\nResult for Student: {Name} (ID: {Id})");
27            bool hasError = false;
28            foreach (var mark in Marks)
29            {
30                if (mark.Value > maxMarks)
31                {
32                    hasError = true;
33                }
34            }
35            if (!hasError)
36            {
37                Console.WriteLine("All marks are within the maximum allowable range.");
38            }
39        }
40    }
41
42    class Program
43    {
44        static void Main(string[] args)
45        {
46            // Input: Maximum marks for a subject
47            Console.Write("Enter the maximum marks for any subject: ");
48            int maxMarks = int.Parse(Console.ReadLine());
49
50            // Create a student object
51            Student student = new Student(1, "Jagriti Patil");
52
53            // Taking student marks for 3 subjects
54            Console.WriteLine("\\nEnter the marks for the following subjects:");
55
56            Console.Write("Math: ");
57            int mathMarks = int.Parse(Console.ReadLine());
58            student.AddMark("Math", mathMarks);
59
60            Console.Write("Science: ");
61            int scienceMarks = int.Parse(Console.ReadLine());
62            student.AddMark("Science", scienceMarks);
63
64            Console.Write("English: ");
65            int englishMarks = int.Parse(Console.ReadLine());
66            student.AddMark("English", englishMarks);
67
68            student.DisplayResult(maxMarks);
69        }
70    }
71
72    static void Main(string[] args)
73    {
74    }
75}
```

Output:

```
50
51 {
52     D:\Sem_7\705_Advance.netA
53
54     Enter the maximum marks for any subject: 100
55
56     Enter the marks for the following subjects:
57     Math: 95
58     Science: 85
59     English: 101
60
61     Result for Student: Jagruti Patil (ID: 1)
62     Math: 95
63     Science: 85
64     Error: Marks in English exceed the maximum allowable marks (100). Obtained: 101
65
66
67
68
69
70
71
72
73
74
```

Process: [23760] Ass1_Q6StudentResult.exe - Lifecycle Events - Thread: Stack Frame: Diagnostic Tools Solution Explorer Git Changes

Program.cs

Ass1_Q6StudentResult

Ass1_Q6StudentResult.Program

Main(string[] args)

Autos

Search (Ctrl+E)

Name

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output

Ready 28°C Cloudy Search ENG IN 7:12 PM 9/25/2024

```
50
51
52     Enter the maximum marks for any subject: 100
53
54     Enter the marks for the following subjects:
55     Math: 95
56     Science: 85
57     English: 72
58
59     Result for Student: Jagruti Patil (ID: 1)
60     Math: 95
61     Science: 85
62     English: 72
63     All marks are within the maximum allowable range.
64
65
66
67
68
69
70
71
72
73
74
```

Process: [8664] Ass1_Q6StudentResult.exe - Lifecycle Events - Thread: Stack Frame: Diagnostic Tools Solution Explorer Git Changes

Program.cs

D:\Sem_7\705_Advance.netA

Ass1_Q6StudentResult

Ass1_Q6StudentResult.Program

Main(string[] args)

Autos

Search (Ctrl+E)

Name

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output

Ready 28°C Cloudy Search ENG IN 7:12 PM 9/25/2024