NAME : K G NANDHIKA

REG NO : 73772214167

DEPT : BE.CSE-B(III)

COURSE CODE : 60 IT L04

COURSE NAME : C# AND .NET FRAMEWORKS

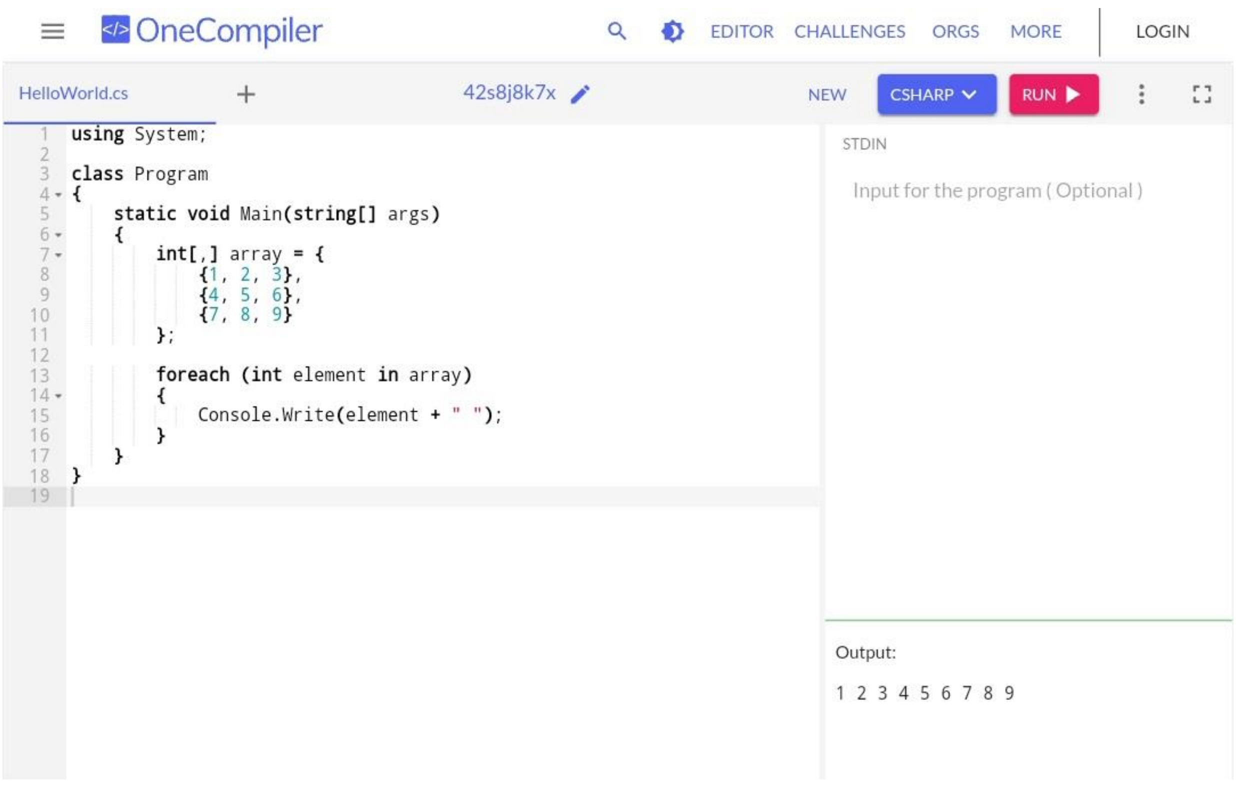
ASSIGNMENT 1

1)

Aim

To display the elements of a 2D array in a formatted grid on the console.

Program



```
using System;

class Program
{
    static void Main(string[] args)
    {
        int[,] array = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        foreach (int element in array)
        {
            Console.Write(element + " ");
        }
    }
}
```

Output:

```
1 2 3 4 5 6 7 8 9
```

Output

```
1 2 3 4 5 6 7 8 9
```

Result

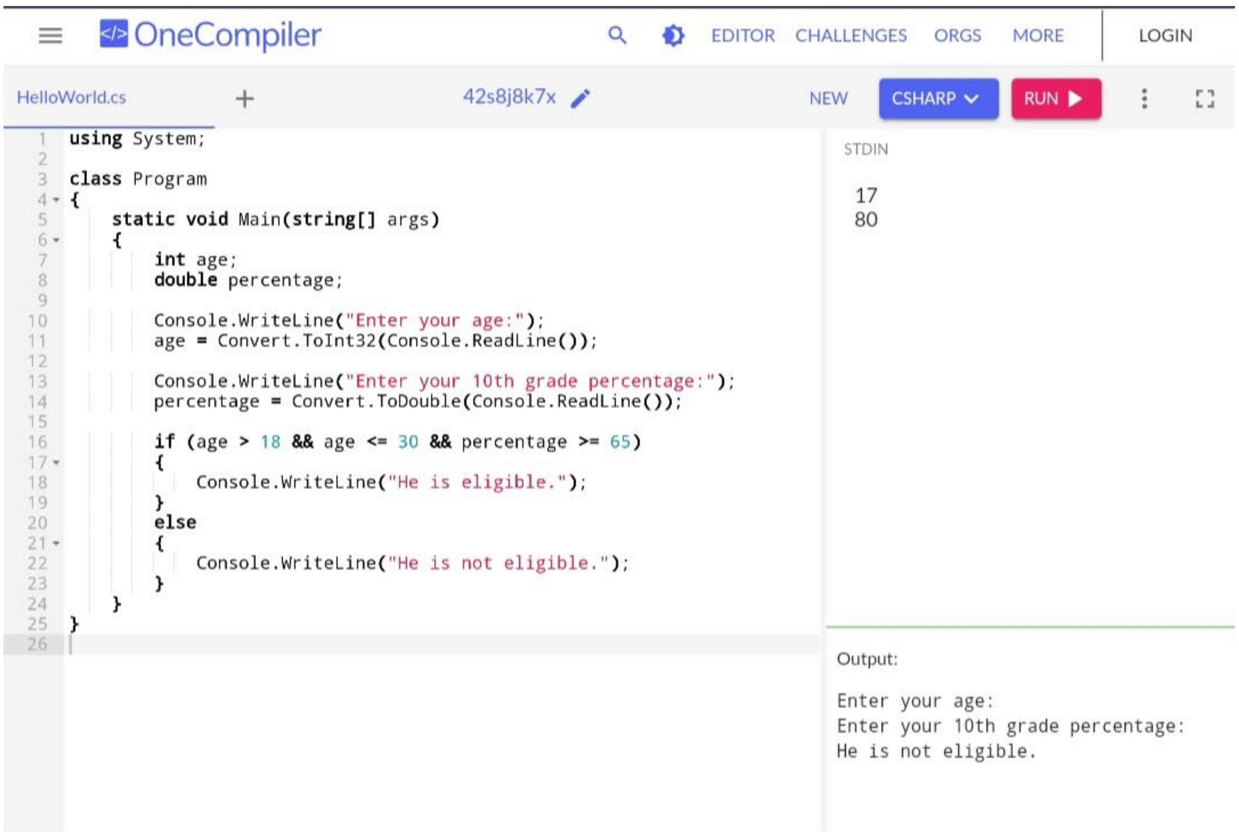
The program iterates through each element of the 2D array using nested for loops and prints each element followed by a space. After printing each row, it moves to the next line. This results in the 2D array being displayed in a grid-like format on the console.

2)

Aim

To determine eligibility based on age and percentage criteria and print the result.

Program



```
using System;

class Program
{
    static void Main(string[] args)
    {
        int age;
        double percentage;

        Console.WriteLine("Enter your age:");
        age = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Enter your 10th grade percentage:");
        percentage = Convert.ToDouble(Console.ReadLine());

        if (age > 18 && age <= 30 && percentage >= 65)
        {
            Console.WriteLine("He is eligible.");
        }
        else
        {
            Console.WriteLine("He is not eligible.");
        }
    }
}
```

STDIN

```
17
80
```

Output:

```
Enter your age:
Enter your 10th grade percentage:
He is not eligible.
```

Input

Enter your age:

25

Enter your 10th grade percentage:

70

Output

He is not Eligible

Result

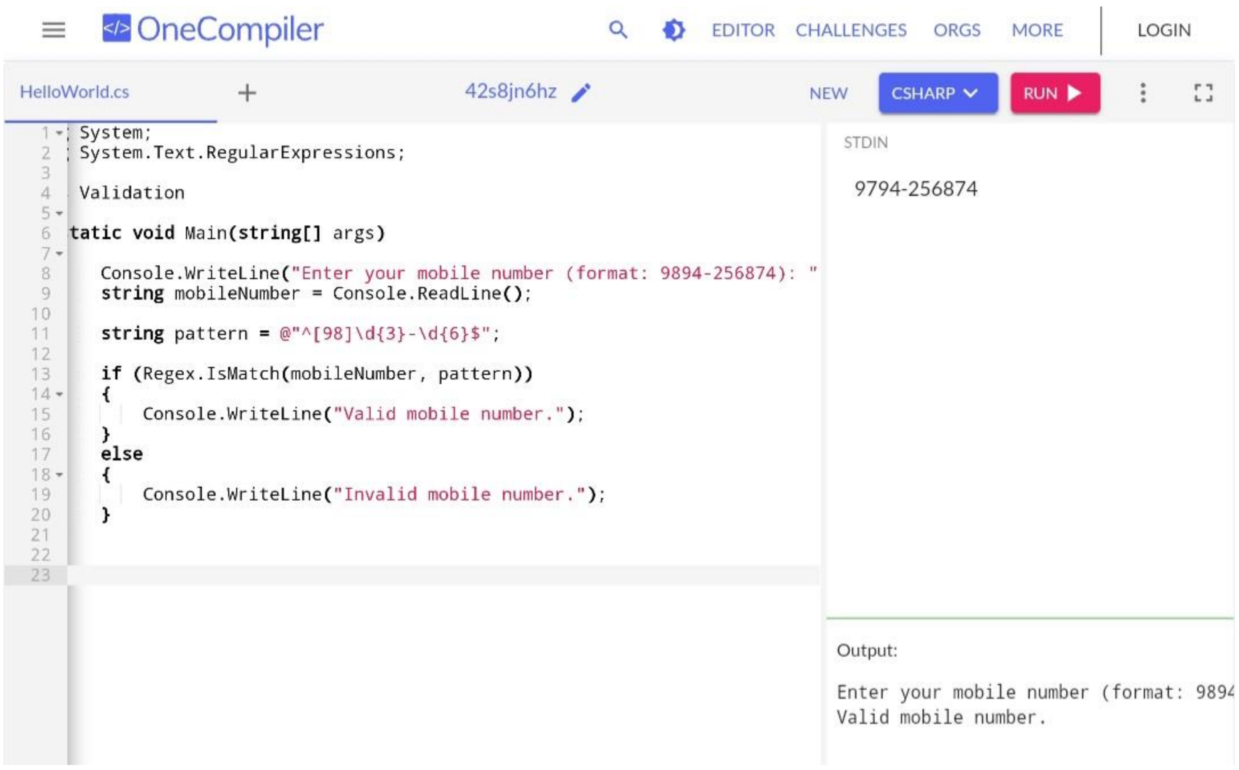
Thus the program executed successfully.

3)

Aim

To validate a mobile number based on specific criteria and print whether it is valid or invalid.

Program



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

 Validation

tatic void Main(string[] args)

   Console.WriteLine("Enter your mobile number (format: 9894-256874): "
   string mobileNumber = Console.ReadLine();

   string pattern = @"^[98]\d{3}-\d{6}$";

   if (Regex.IsMatch(mobileNumber, pattern))
   {
       Console.WriteLine("Valid mobile number.");
   }
   else
   {
       Console.WriteLine("Invalid mobile number.");
   }
```

STDIN

```
9794-256874
```

Output:

```
Enter your mobile number (format: 9894
Valid mobile number.
```

Input

9876-543210

Output

Valid mobile number

Result

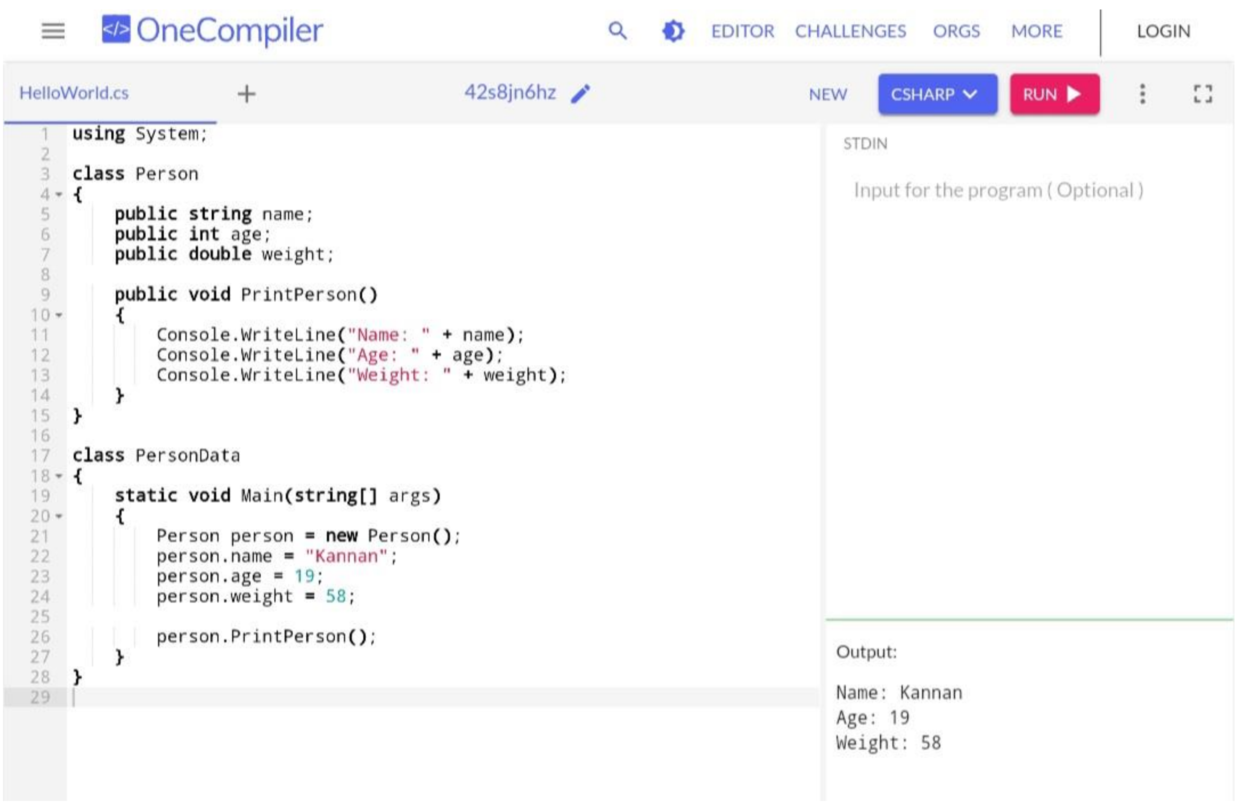
Thus the program executed successfully.

4)

Aim

To create a Person class with attributes and a method to print the person's details, and to use this class in the PersonData class to demonstrate its functionality.

Program



```
using System;

class Person
{
    public string name;
    public int age;
    public double weight;

    public void PrintPerson()
    {
        Console.WriteLine("Name: " + name);
        Console.WriteLine("Age: " + age);
        Console.WriteLine("Weight: " + weight);
    }
}

class PersonData
{
    static void Main(string[] args)
    {
        Person person = new Person();
        person.name = "Kannan";
        person.age = 19;
        person.weight = 58;

        person.PrintPerson();
    }
}
```

```
Name: Kannan
Age: 19
Weight: 58
```

Output

Name: Kannan
Age: 19
Weight: 58

Result

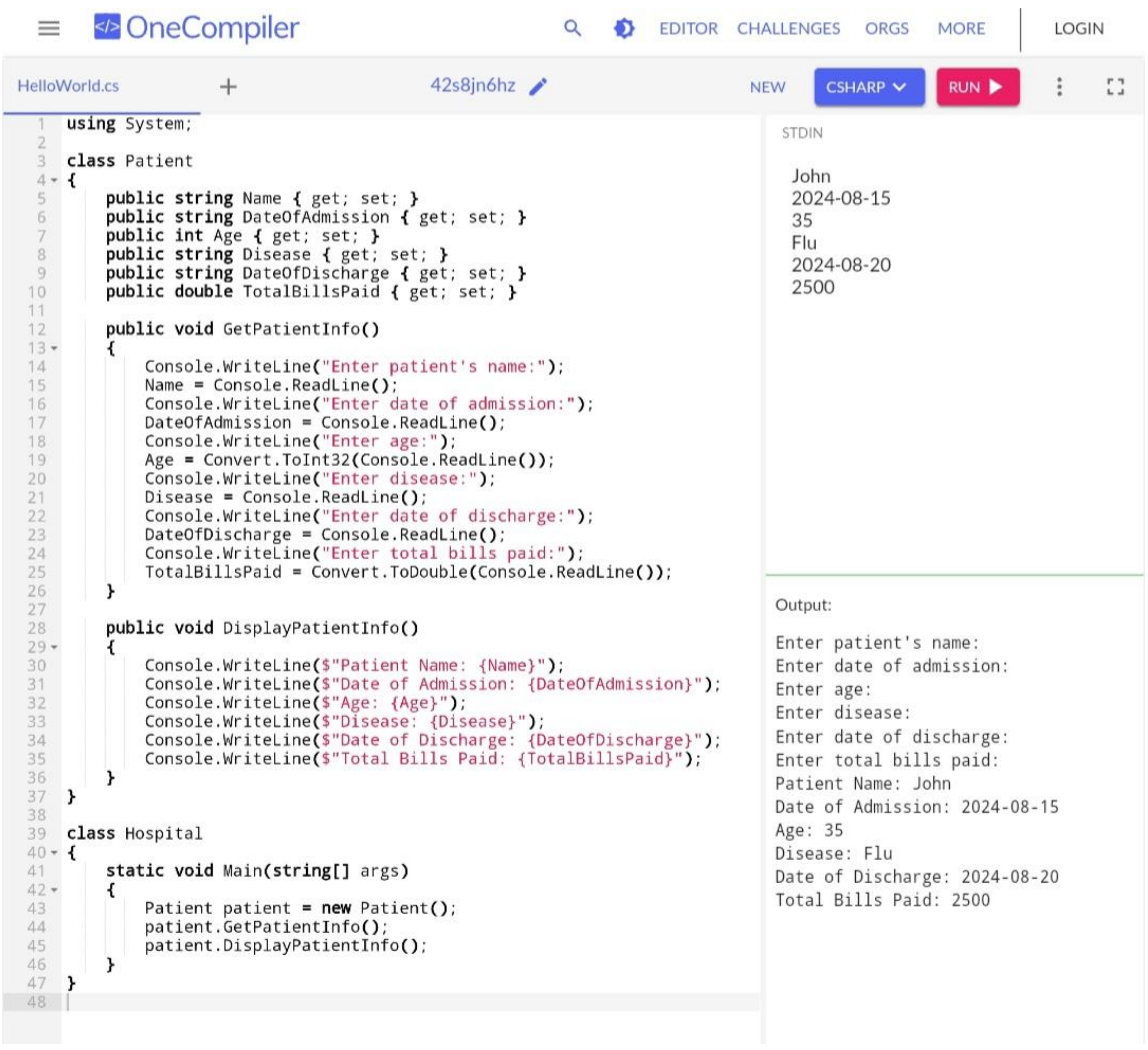
Thus the program executed successfully.

5)

Aim

To manage and display patient information including details like name, admission and discharge dates, age, disease, and total bills paid.

Program



```
using System;

class Patient
{
    public string Name { get; set; }
    public string DateOfAdmission { get; set; }
    public int Age { get; set; }
    public string Disease { get; set; }
    public string DateOfDischarge { get; set; }
    public double TotalBillsPaid { get; set; }

    public void GetPatientInfo()
    {
        Console.WriteLine("Enter patient's name:");
        Name = Console.ReadLine();
        Console.WriteLine("Enter date of admission:");
        DateOfAdmission = Console.ReadLine();
        Console.WriteLine("Enter age:");
        Age = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter disease:");
        Disease = Console.ReadLine();
        Console.WriteLine("Enter date of discharge:");
        DateOfDischarge = Console.ReadLine();
        Console.WriteLine("Enter total bills paid:");
        TotalBillsPaid = Convert.ToDouble(Console.ReadLine());
    }

    public void DisplayPatientInfo()
    {
        Console.WriteLine($"Patient Name: {Name}");
        Console.WriteLine($"Date of Admission: {DateOfAdmission}");
        Console.WriteLine($"Age: {Age}");
        Console.WriteLine($"Disease: {Disease}");
        Console.WriteLine($"Date of Discharge: {DateOfDischarge}");
        Console.WriteLine($"Total Bills Paid: {TotalBillsPaid}");
    }
}

class Hospital
{
    static void Main(string[] args)
    {
        Patient patient = new Patient();
        patient.GetPatientInfo();
        patient.DisplayPatientInfo();
    }
}
```

STDIN

```
John
2024-08-15
35
Flu
2024-08-20
2500
```

Output:

```
Enter patient's name:
Enter date of admission:
Enter age:
Enter disease:
Enter date of discharge:
Enter total bills paid:
Patient Name: John
Date of Admission: 2024-08-15
Age: 35
Disease: Flu
Date of Discharge: 2024-08-20
Total Bills Paid: 2500
```

Input

```
Enter patient’s name:
John
Enter date of admission:
2024-08-15
Enter age:
35
Enter disease:
Flu
Enter date of discharge:
2024-08-20
Enter total bills paid:
2500
```

Output

```
Patient Name: John
Date of Admission: 2024-08-15
Age: 35
Disease: Flu
Date of Discharge: 2024-08-20
Total Bills Paid: 2500
```

Result

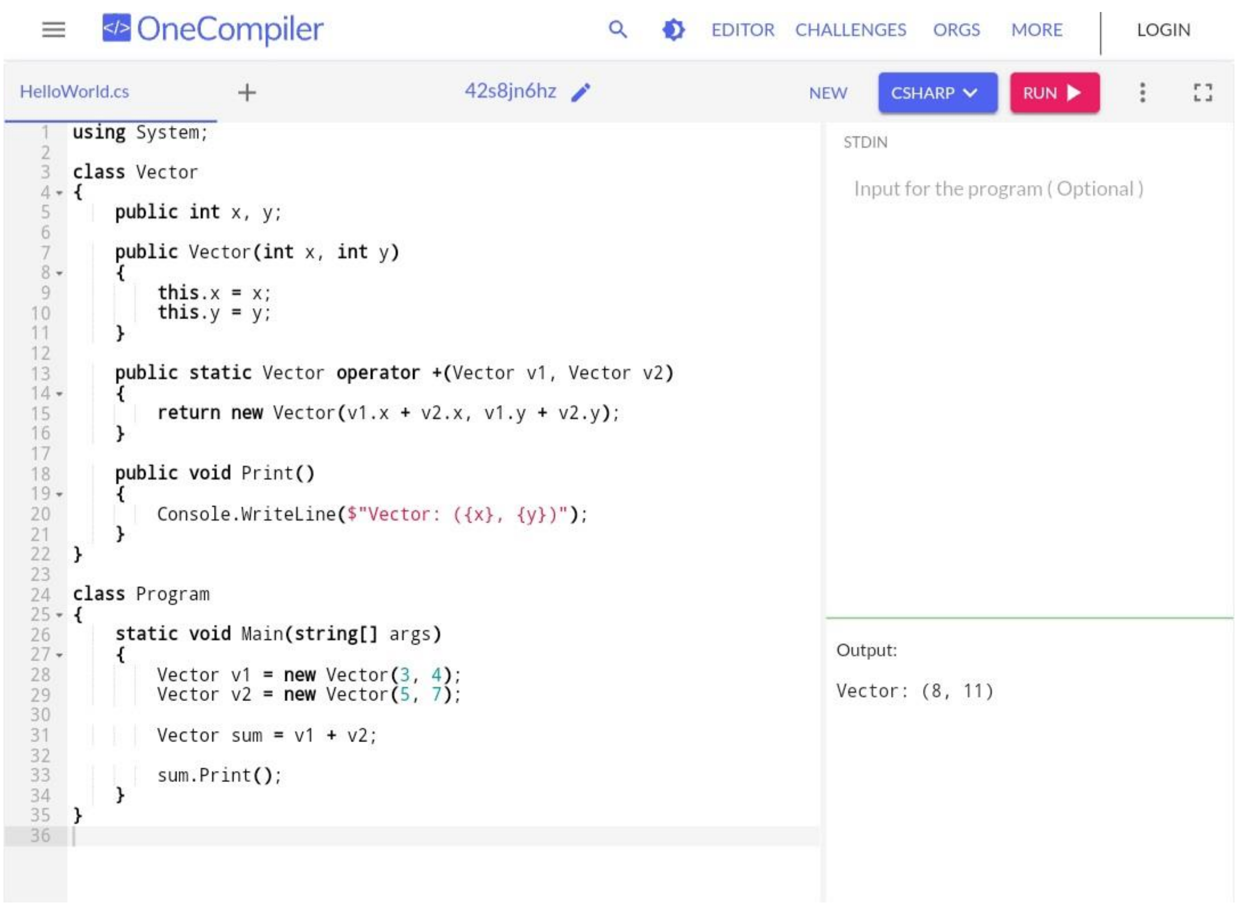
Thus the program executed successfully.

6)

Aim

To demonstrate operator overloading in C# by implementing the + operator for a Vector class, which allows for the addition of two vectors.

Program



OneCompiler
EDITOR CHALLENGES ORGS MORE LOGIN
HelloWorld.cs
42s8jn6hz
NEW CSHARP RUN
STDIN
Input for the program (Optional)

```csharp
using System;

class Vector
{
    public int x, y;

    public Vector(int x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public static Vector operator +(Vector v1, Vector v2)
    {
        return new Vector(v1.x + v2.x, v1.y + v2.y);
    }

    public void Print()
    {
        Console.WriteLine($"Vector: ({x}, {y})");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Vector v1 = new Vector(3, 4);
        Vector v2 = new Vector(5, 7);

        Vector sum = v1 + v2;

        sum.Print();
    }
}
```

Output:

```
Vector: (8, 11)
```

Output

```
Vector:
(8,11)
```

Result

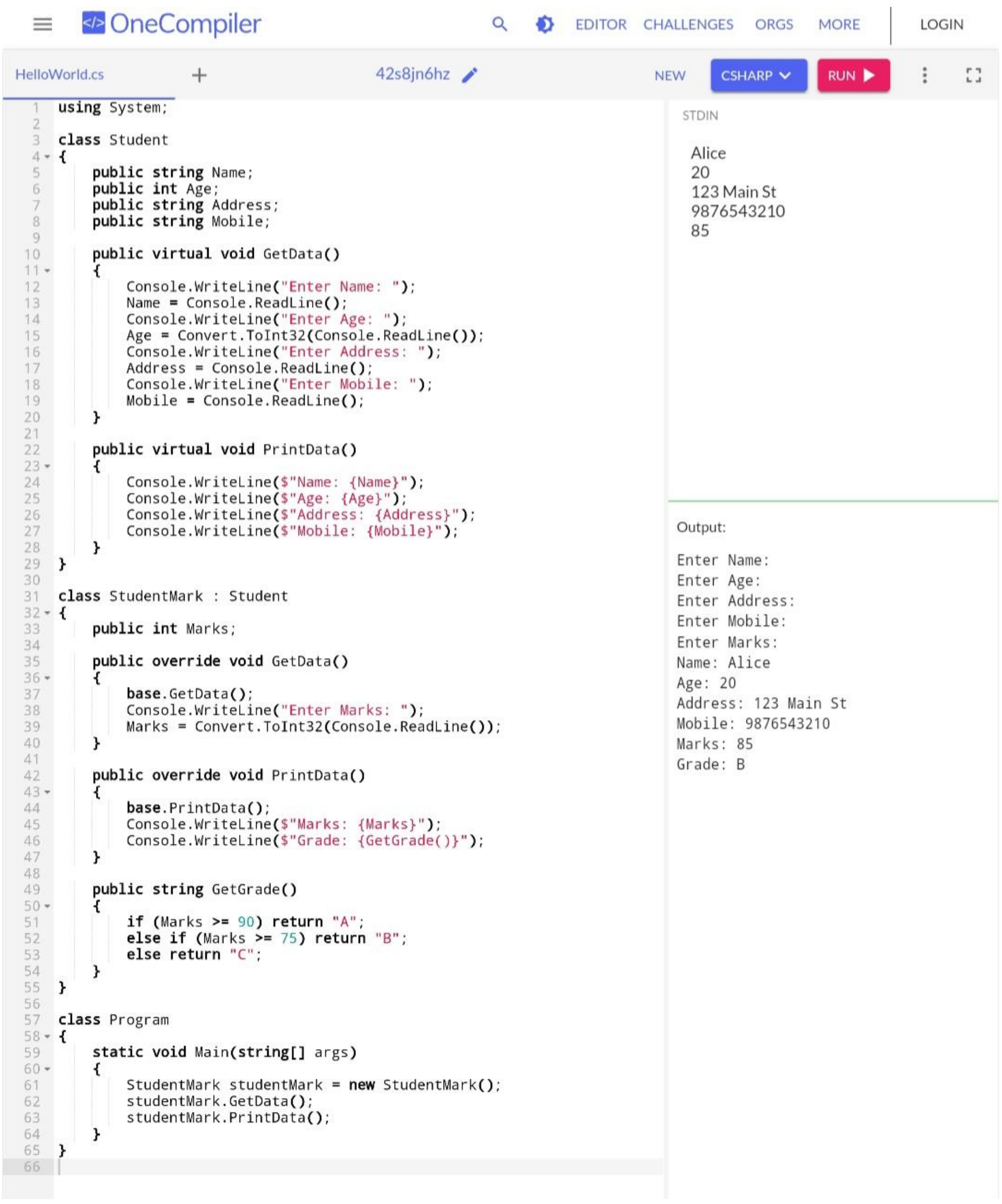
Thus the program executed successfully.

7)

Aim

To demonstrate inheritance and method overriding in C# by creating a base class Student and a derived class StudentMark that adds additional functionality.

Program



```csharp
using System;

class Student
{
    public string Name;
    public int Age;
    public string Address;
    public string Mobile;

    public virtual void GetData()
    {
        Console.WriteLine("Enter Name: ");
        Name = Console.ReadLine();
        Console.WriteLine("Enter Age: ");
        Age = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter Address: ");
        Address = Console.ReadLine();
        Console.WriteLine("Enter Mobile: ");
        Mobile = Console.ReadLine();
    }

    public virtual void PrintData()
    {
        Console.WriteLine($"Name: {Name}");
        Console.WriteLine($"Age: {Age}");
        Console.WriteLine($"Address: {Address}");
        Console.WriteLine($"Mobile: {Mobile}");
    }
}

class StudentMark : Student
{
    public int Marks;

    public override void GetData()
    {
        base.GetData();
        Console.WriteLine("Enter Marks: ");
        Marks = Convert.ToInt32(Console.ReadLine());
    }

    public override void PrintData()
    {
        base.PrintData();
        Console.WriteLine($"Marks: {Marks}");
        Console.WriteLine($"Grade: {GetGrade()}");
    }

    public string GetGrade()
    {
        if (Marks >= 90) return "A";
        else if (Marks >= 75) return "B";
        else return "C";
    }
}

class Program
{
    static void Main(string[] args)
    {
        StudentMark studentMark = new StudentMark();
        studentMark.GetData();
        studentMark.PrintData();
    }
}
```

STDIN

```
Alice
20
123 Main St
9876543210
85
```

Output:

```
Enter Name:
Enter Age:
Enter Address:
Enter Mobile:
Enter Marks:
Name: Alice
Age: 20
Address: 123 Main St
Mobile: 9876543210
Marks: 85
Grade: B
```

Input

```
Enter Name:
Alice
Enter Age:
20
Enter Address:
123 Main St
Enter Mobile:
9876543210
Enter Marks:
85
```

Output

```
Name: Alice
Age: 20
Address: 123 Main St
Mobile: 9876543210
Marks: 85
Grade: B
```

Result

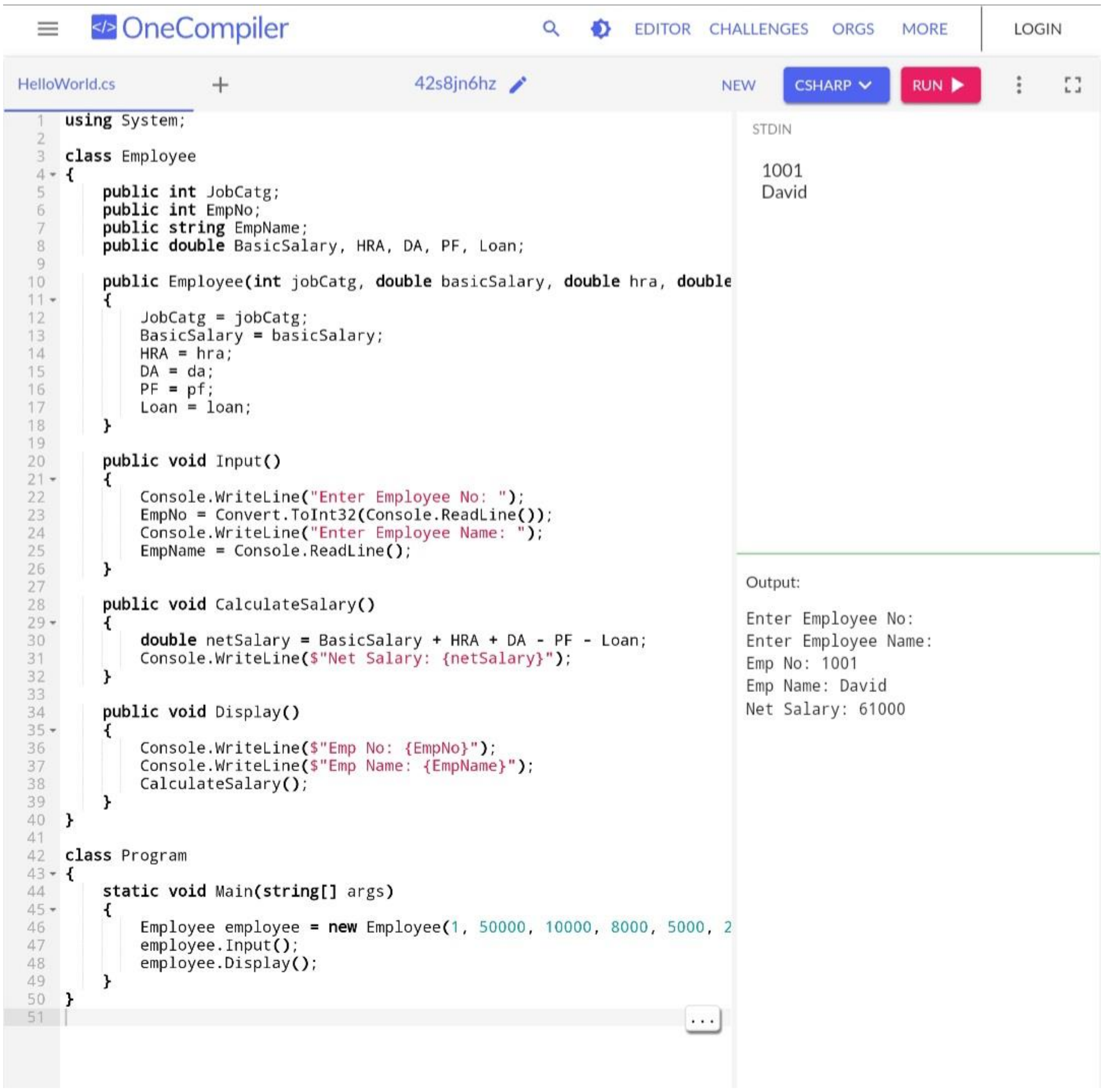
Thus the program has been executed successfully

8)

Aim

To calculate and display the salary details of an employee based on their job category, including components like basic salary, HRA (House Rent Allowance), DA (Dearness Allowance), loan, and PF (Provident Fund).

Program



```csharp
using System;

class Employee
{
    public int JobCatg;
    public int EmpNo;
    public string EmpName;
    public double BasicSalary, HRA, DA, PF, Loan;

    public Employee(int jobCatg, double basicSalary, double hra, double
    {
        JobCatg = jobCatg;
        BasicSalary = basicSalary;
        HRA = hra;
        DA = da;
        PF = pf;
        Loan = loan;
    }

    public void Input()
    {
        Console.WriteLine("Enter Employee No: ");
        EmpNo = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter Employee Name: ");
        EmpName = Console.ReadLine();
    }

    public void CalculateSalary()
    {
        double netSalary = BasicSalary + HRA + DA - PF - Loan;
        Console.WriteLine($"Net Salary: {netSalary}");
    }

    public void Display()
    {
        Console.WriteLine($"Emp No: {EmpNo}");
        Console.WriteLine($"Emp Name: {EmpName}");
        CalculateSalary();
    }
}

class Program
{
    static void Main(string[] args)
    {
        Employee employee = new Employee(1, 50000, 10000, 8000, 5000, 2
        employee.Input();
        employee.Display();
    }
}
```

STDIN

```
1001
David
```

Output:

```
Enter Employee No:
Enter Employee Name:
Emp No: 1001
Emp Name: David
Net Salary: 61000
```

Input

```
Enter
Employee No:
1001
Enter
Employee
Name:
David
```

Output

```
Emp No: 1001
Emp Name: David
Net Salary: 61000
```

Result

Thus the program executed successfully.