

NAME : K G NANDHIKA
REG NO : 73772214167
DEPT : BE.CSE-B(III)
COURSE CODE : 60 IT L04
COURSE NAME : C# AND .NET FRAMEWORKS

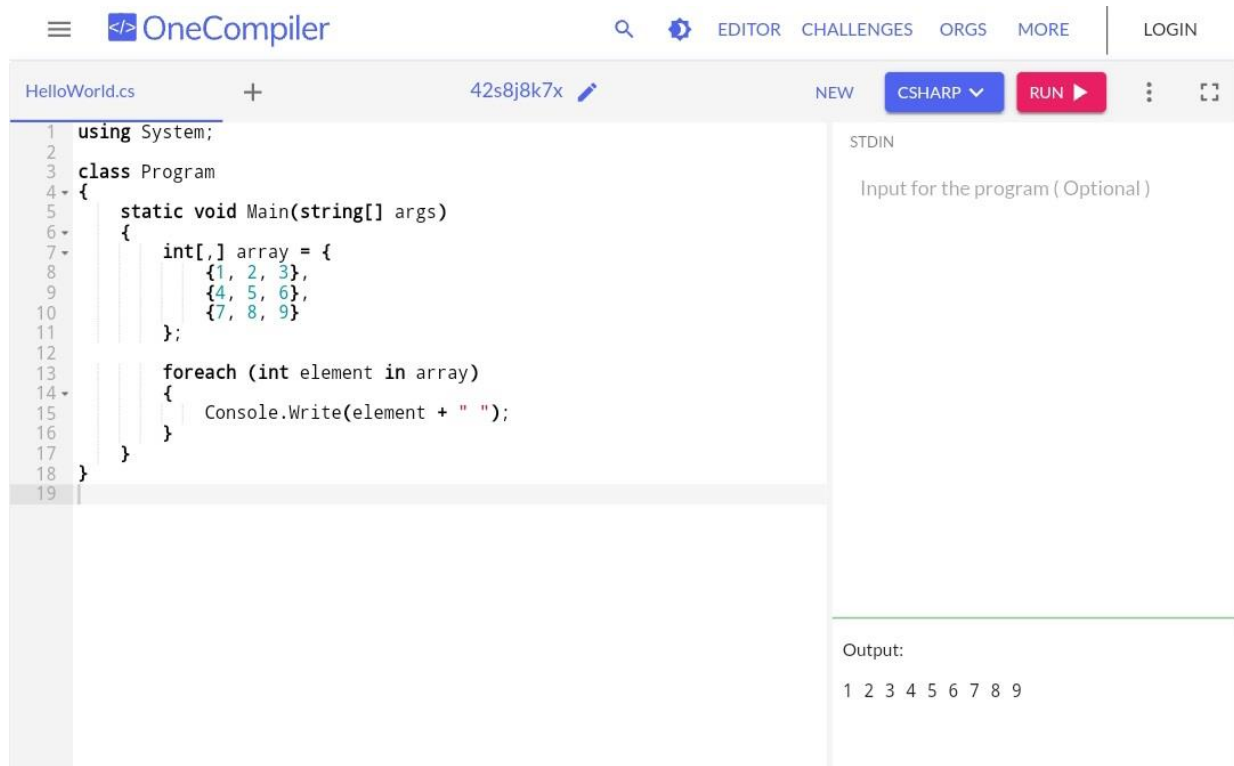
ASSIGNMENT 1

1)

Aim

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

Program



The screenshot shows the OneCompiler C# IDE. The editor displays a C# program named 'HelloWorld.cs' with the following code:

```
1 using System;
2
3 class Program
4 {
5     static void Main(string[] args)
6     {
7         int[,] array = {
8             {1, 2, 3},
9             {4, 5, 6},
10            {7, 8, 9}
11        };
12
13        foreach (int element in array)
14        {
15            Console.Write(element + " ");
16        }
17    }
18 }
19
```

The right-hand side of the IDE shows the 'STDIN' section with the text 'Input for the program (Optional)'. Below it, the 'Output' section displays the result of the program execution:

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

Output

1 2 3 4 5 6 7 8 9

Result

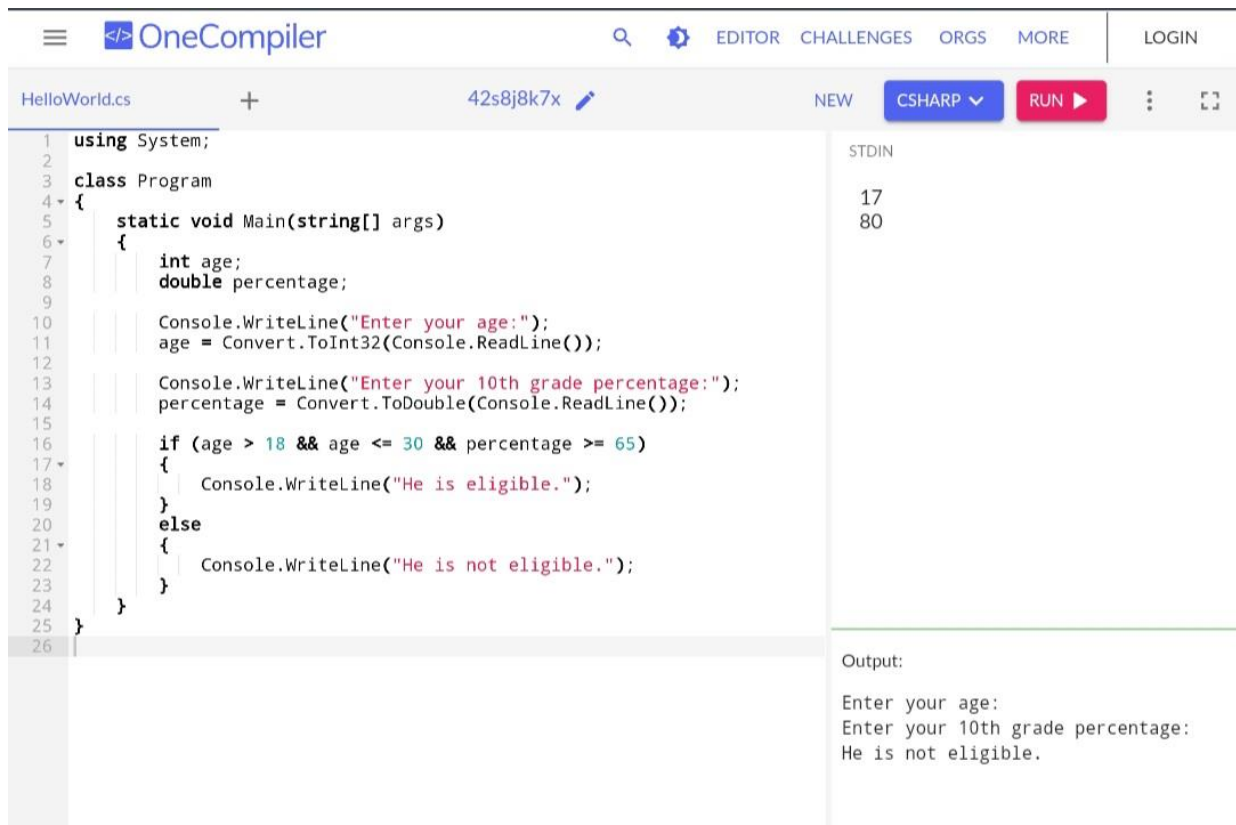
The program iterates through each element of the 2D array using nested for loops and prints each element followed by a space. After printing each row, it moves to the next line. This results in the 2D array being displayed in a grid-like format on the console.

2)

Aim

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

Program



The screenshot shows the OneCompiler web-based IDE. The editor contains a C# program named 'HelloWorld.cs'. The program defines a 'Program' class with a 'Main' method. It prompts the user to enter their age and 10th grade percentage. The eligibility criteria are: age must be greater than 18 and less than or equal to 30, and the percentage must be greater than or equal to 65. If both conditions are met, it prints 'He is eligible.'; otherwise, it prints 'He is not eligible.'.

```
1 using System;
2
3 class Program
4 {
5     static void Main(string[] args)
6     {
7         int age;
8         double percentage;
9
10        Console.WriteLine("Enter your age:");
11        age = Convert.ToInt32(Console.ReadLine());
12
13        Console.WriteLine("Enter your 10th grade percentage:");
14        percentage = Convert.ToDouble(Console.ReadLine());
15
16        if (age > 18 && age <= 30 && percentage >= 65)
17        {
18            Console.WriteLine("He is eligible.");
19        }
20        else
21        {
22            Console.WriteLine("He is not eligible.");
23        }
24    }
25 }
26
```

The right-hand pane shows the 'STDIN' input with the values '17' and '80'. The 'Output' pane shows the program's execution: 'Enter your age:', 'Enter your 10th grade percentage:', and 'He is not eligible.'

Input

Enter your age:

25

Enter your 10th grade percentage:

70

Output

He is not Eligible

Result

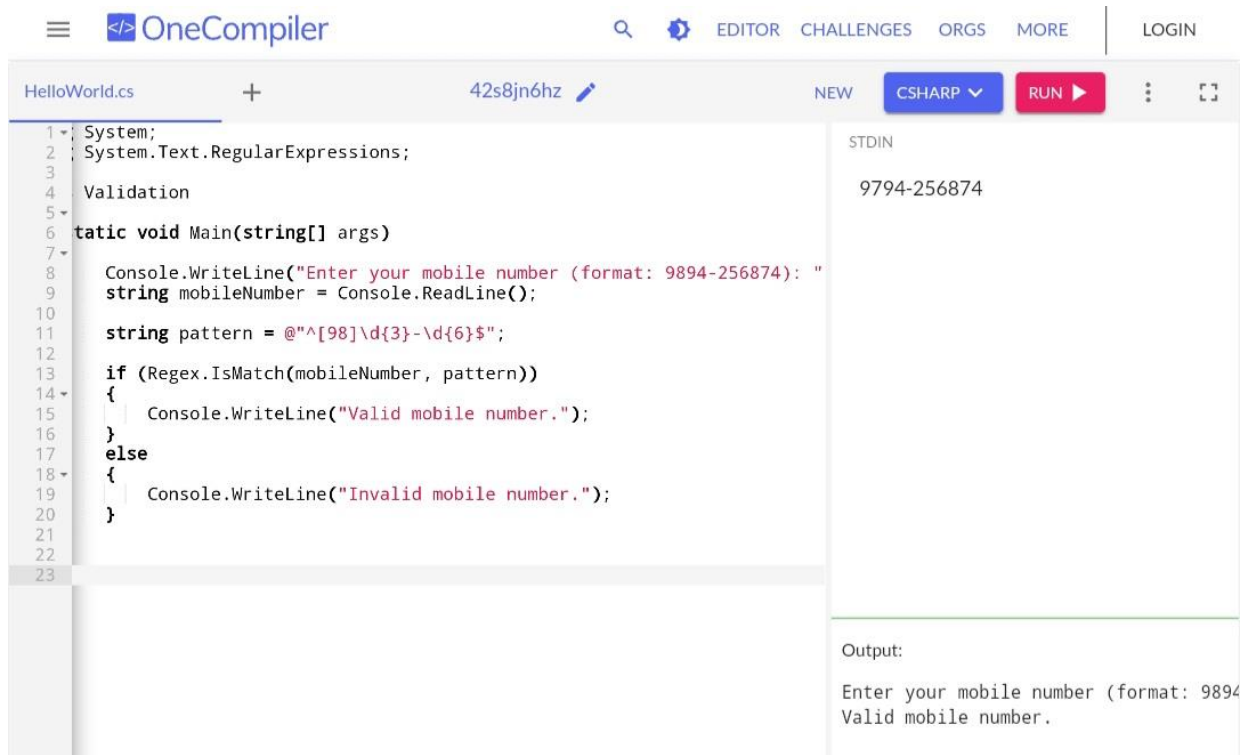
Thus the program executed successfully.

3)

Aim

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

Program



The screenshot shows the OneCompiler web-based IDE. The editor displays a C# program for validating a mobile number. The code includes the necessary namespaces, a regular expression pattern, and a Main method that prompts the user for input and checks it against the pattern. The output pane shows the program's execution with the input '9794-256874' and the resulting output 'Valid mobile number.'.

```
1 System;
2 System.Text.RegularExpressions;
3
4 Validation
5
6 static void Main(string[] args)
7
8     Console.WriteLine("Enter your mobile number (format: 9894-256874): ");
9     string mobileNumber = Console.ReadLine();
10
11     string pattern = @"^[98]\d{3}-\d{6}$";
12
13     if (Regex.IsMatch(mobileNumber, pattern))
14     {
15         Console.WriteLine("Valid mobile number.");
16     }
17     else
18     {
19         Console.WriteLine("Invalid mobile number.");
20     }
21
22
23
```

STDIN

9794-256874

Output:

Enter your mobile number (format: 9894-256874):
Valid mobile number.

Input

9876-543210

Output

Valid mobile number

Result

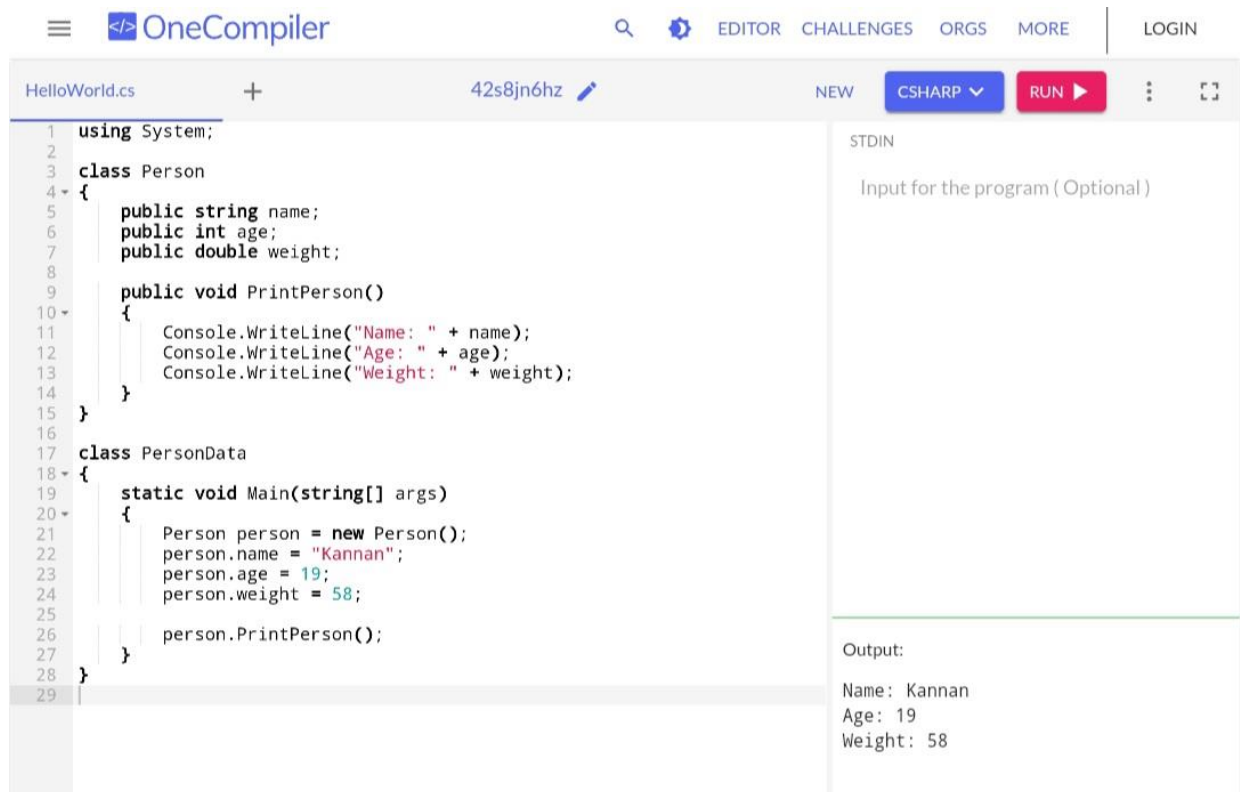
Thus the program executed successfully.

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



The screenshot shows the OneCompiler C# IDE. The editor displays a C# program with two classes: `Person` and `PersonData`. The `Person` class has three public attributes: `name` (string), `age` (int), and `weight` (double). It also has a public method `PrintPerson()` that prints the details. The `PersonData` class has a static `Main` method that creates a `Person` object, sets its attributes to "Kannan", 19, and 58, and then calls `PrintPerson()`. The output window on the right shows the result of the program execution.

```
1 using System;
2
3 class Person
4 {
5     public string name;
6     public int age;
7     public double weight;
8
9     public void PrintPerson()
10    {
11        Console.WriteLine("Name: " + name);
12        Console.WriteLine("Age: " + age);
13        Console.WriteLine("Weight: " + weight);
14    }
15 }
16
17 class PersonData
18 {
19     static void Main(string[] args)
20     {
21         Person person = new Person();
22         person.name = "Kannan";
23         person.age = 19;
24         person.weight = 58;
25
26         person.PrintPerson();
27     }
28 }
29
```

STDIN

Input for the program (Optional)

Output:

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

OneCompiler

42s8jn6hz

EDITOR CHALLENGES ORGS MORE

LOGIN

HelloWorld.cs

+

NEW

CSHARP

RUN

```
1 using System;
2
3 class Patient
4 {
5     public string Name { get; set; }
6     public string DateOfAdmission { get; set; }
7     public int Age { get; set; }
8     public string Disease { get; set; }
9     public string DateOfDischarge { get; set; }
10    public double TotalBillsPaid { get; set; }
11
12    public void GetPatientInfo()
13    {
14        Console.WriteLine("Enter patient's name:");
15        Name = Console.ReadLine();
16        Console.WriteLine("Enter date of admission:");
17        DateOfAdmission = Console.ReadLine();
18        Console.WriteLine("Enter age:");
19        Age = Convert.ToInt32(Console.ReadLine());
20        Console.WriteLine("Enter disease:");
21        Disease = Console.ReadLine();
22        Console.WriteLine("Enter date of discharge:");
23        DateOfDischarge = Console.ReadLine();
24        Console.WriteLine("Enter total bills paid:");
25        TotalBillsPaid = Convert.ToDouble(Console.ReadLine());
26    }
27
28    public void DisplayPatientInfo()
29    {
30        Console.WriteLine($"Patient Name: {Name}");
31        Console.WriteLine($"Date of Admission: {DateOfAdmission}");
32        Console.WriteLine($"Age: {Age}");
33        Console.WriteLine($"Disease: {Disease}");
34        Console.WriteLine($"Date of Discharge: {DateOfDischarge}");
35        Console.WriteLine($"Total Bills Paid: {TotalBillsPaid}");
36    }
37 }
38
39 class Hospital
40 {
41     static void Main(string[] args)
42     {
43         Patient patient = new Patient();
44         patient.GetPatientInfo();
45         patient.DisplayPatientInfo();
46     }
47 }
48
```

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

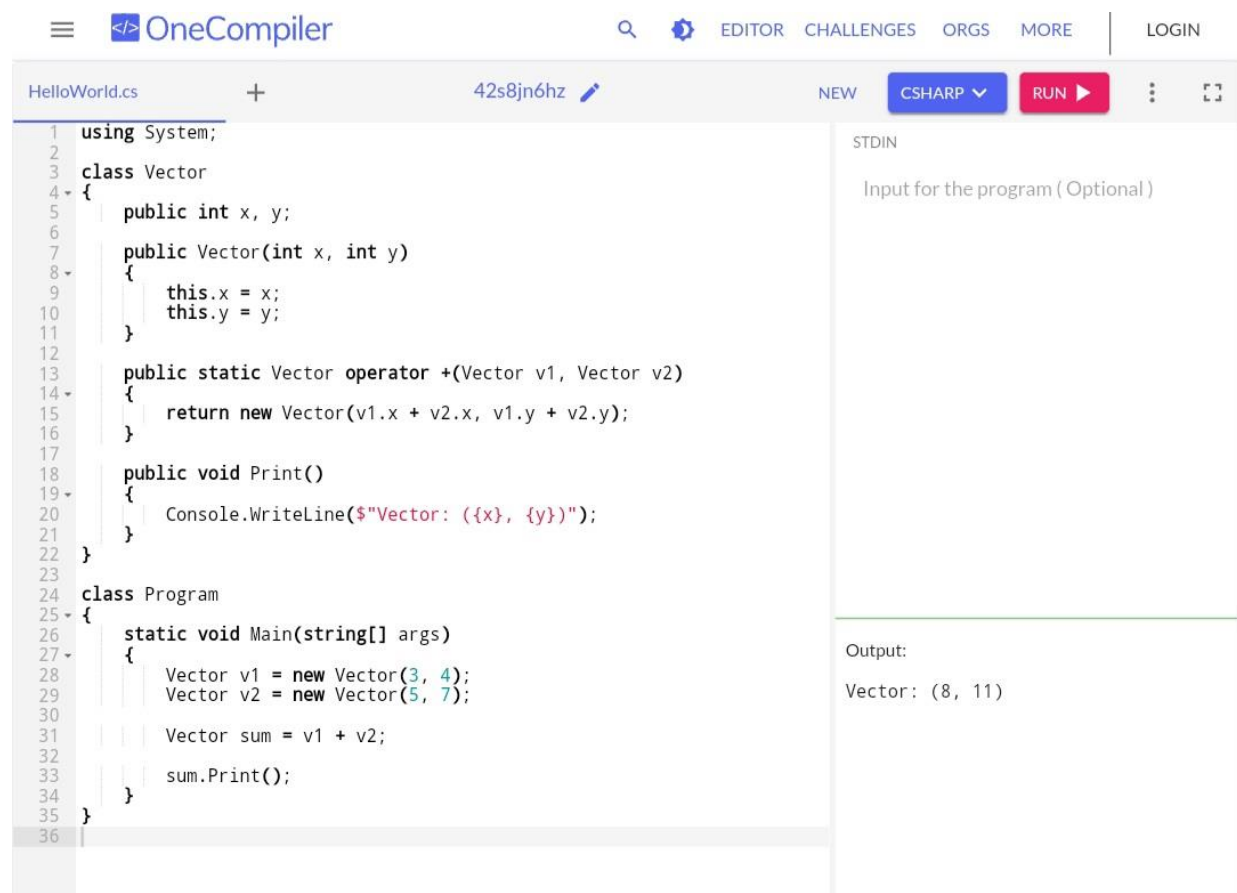
Thus the program executed successfully.

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



The screenshot shows the OneCompiler web IDE. The editor displays a C# program with the following code:

```
1 using System;
2
3 class Vector
4 {
5     public int x, y;
6
7     public Vector(int x, int y)
8     {
9         this.x = x;
10        this.y = y;
11    }
12
13    public static Vector operator +(Vector v1, Vector v2)
14    {
15        return new Vector(v1.x + v2.x, v1.y + v2.y);
16    }
17
18    public void Print()
19    {
20        Console.WriteLine($"Vector: ({x}, {y})");
21    }
22 }
23
24 class Program
25 {
26     static void Main(string[] args)
27     {
28         Vector v1 = new Vector(3, 4);
29         Vector v2 = new Vector(5, 7);
30
31         Vector sum = v1 + v2;
32
33         sum.Print();
34     }
35 }
36
```

The right-hand sidebar shows the STDIN input field with the text "Input for the program (Optional)". Below it, the Output section displays the result of the program execution:

```
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



The screenshot shows the OneCompiler C# IDE. The editor displays a C# program with the following code:

```
1 using System;
2
3 class Student
4 {
5     public string Name;
6     public int Age;
7     public string Address;
8     public string Mobile;
9
10    public virtual void GetData()
11    {
12        Console.WriteLine("Enter Name: ");
13        Name = Console.ReadLine();
14        Console.WriteLine("Enter Age: ");
15        Age = Convert.ToInt32(Console.ReadLine());
16        Console.WriteLine("Enter Address: ");
17        Address = Console.ReadLine();
18        Console.WriteLine("Enter Mobile: ");
19        Mobile = Console.ReadLine();
20    }
21
22    public virtual void PrintData()
23    {
24        Console.WriteLine($"Name: {Name}");
25        Console.WriteLine($"Age: {Age}");
26        Console.WriteLine($"Address: {Address}");
27        Console.WriteLine($"Mobile: {Mobile}");
28    }
29 }
30
31 class StudentMark : Student
32 {
33     public int Marks;
34
35     public override void GetData()
36     {
37         base.GetData();
38         Console.WriteLine("Enter Marks: ");
39         Marks = Convert.ToInt32(Console.ReadLine());
40     }
41
42     public override void PrintData()
43     {
44         base.PrintData();
45         Console.WriteLine($"Marks: {Marks}");
46         Console.WriteLine($"Grade: {GetGrade()}");
47     }
48
49     public string GetGrade()
50     {
51         if (Marks >= 90) return "A";
52         else if (Marks >= 75) return "B";
53         else return "C";
54     }
55 }
56
57 class Program
58 {
59     static void Main(string[] args)
60     {
61         StudentMark studentMark = new StudentMark();
62         studentMark.GetData();
63         studentMark.PrintData();
64     }
65 }
66
```

The output of the program is displayed on the right side of the IDE:

```
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

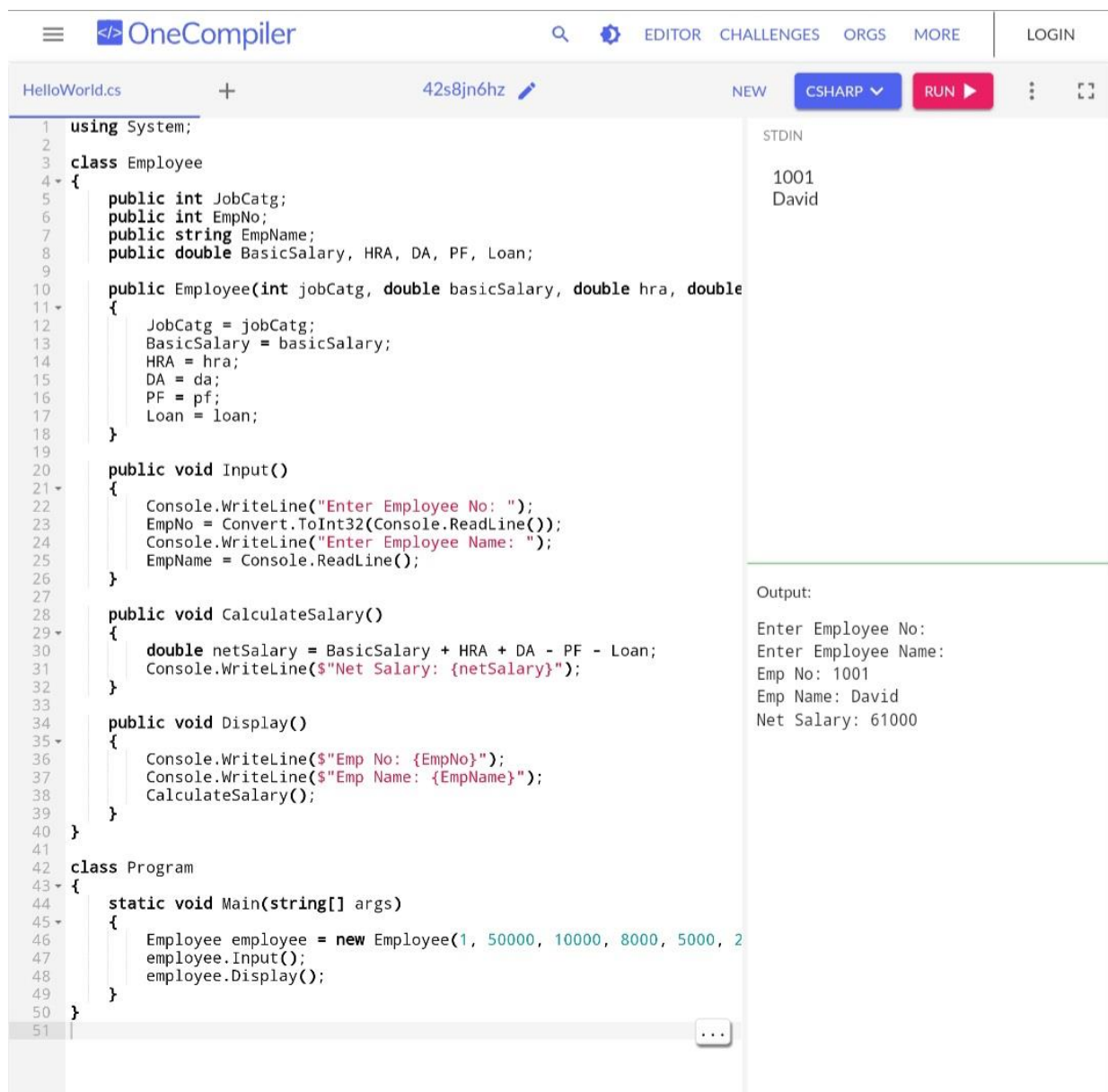
Thus the program has been executed successfully

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



The screenshot shows the OneCompiler C# IDE interface. The editor displays a C# program for calculating an employee's net salary. The program defines an `Employee` class with properties for job category, employee number, name, and various allowances/deductions. It includes methods for input, calculation, and display. The `Program` class contains the `Main` method, which creates an `Employee` object with specific values and calls the `Input` and `Display` methods. The output pane on the right shows the program's execution, displaying the input values and the calculated net salary of 61000.

```
1 using System;
2
3 class Employee
4 {
5     public int JobCatg;
6     public int EmpNo;
7     public string EmpName;
8     public double BasicSalary, HRA, DA, PF, Loan;
9
10    public Employee(int jobCatg, double basicSalary, double hra, double
11    {
12        JobCatg = jobCatg;
13        BasicSalary = basicSalary;
14        HRA = hra;
15        DA = da;
16        PF = pf;
17        Loan = loan;
18    }
19
20    public void Input()
21    {
22        Console.WriteLine("Enter Employee No: ");
23        EmpNo = Convert.ToInt32(Console.ReadLine());
24        Console.WriteLine("Enter Employee Name: ");
25        EmpName = Console.ReadLine();
26    }
27
28    public void CalculateSalary()
29    {
30        double netSalary = BasicSalary + HRA + DA - PF - Loan;
31        Console.WriteLine($"Net Salary: {netSalary}");
32    }
33
34    public void Display()
35    {
36        Console.WriteLine($"Emp No: {EmpNo}");
37        Console.WriteLine($"Emp Name: {EmpName}");
38        CalculateSalary();
39    }
40 }
41
42 class Program
43 {
44     static void Main(string[] args)
45     {
46         Employee employee = new Employee(1, 50000, 10000, 8000, 5000, 2
47         employee.Input();
48         employee.Display();
49     }
50 }
51
```

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.