

# C# and .NET

## Assignment 1

**1. Develop the C# program to initialize two dimensional array and print all the elements of the array on the same line separated with space.**

**Aim:**

To develop the C# program to initialize two dimensional array and print all the elements of the array

**Program:**

```
using System;
class Program
{
    static void Main()
    {
        int[,] array = {
            { 1, 2, 3 },
            { 4, 5, 6 },
            { 7, 8, 9 }
        };
        int rows = array.GetLength(0);
        int cols = array.GetLength(1);
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                Console.Write(array[i, j] + " ");
            }
            Console.WriteLine();
        }
    }
}
```

```

    }
}
}

```

The screenshot shows a C# IDE with a file named 'Main.cs'. The code defines a class 'Program' with a static method 'Main()'. Inside 'Main()', a 2D array 'array' is initialized with the values:
   
 {
   
   { 1, 2, 3 },
   
   { 4, 5, 6 },
   
   { 7, 8, 9 }
   
 }.
   
 The program then calculates the number of rows and columns using 'array.GetLength(0)' and 'array.GetLength(1)'. It uses nested 'for' loops to iterate through each element of the array and prints it to the console using 'Console.Write(array[i, j] + " ");'. After each row is printed, 'Console.WriteLine()' is called to move to the next line.
   
 The 'Output' window on the right shows the execution path: 'mono /tmp/v3VjKQq8zy.exe', followed by the printed output:
   
 1 2 3
   
 4 5 6
   
 7 8 9
   
 The execution ends with the message '=== Code Execution Successful ==='.

## Output:

1 2 3 4 5 6 7 8 9

**2. Aravind wants to apply for competitive exam. He needs to know whether he is eligible to apply. The eligibility criteria is given below:**

- Age should be greater than 18 years, but not more than 30.
- The candidate should have passed 10 std with a minimum pass percentage of 65.

**Design the C# program to help him to know his eligibility. If the criteria gets satisfied, print he is eligible else print he is not eligible.**

## Aim:

To design the C# program to check eligibility for competitive exam

## Program:

```
using System;
```

```
class Program
```

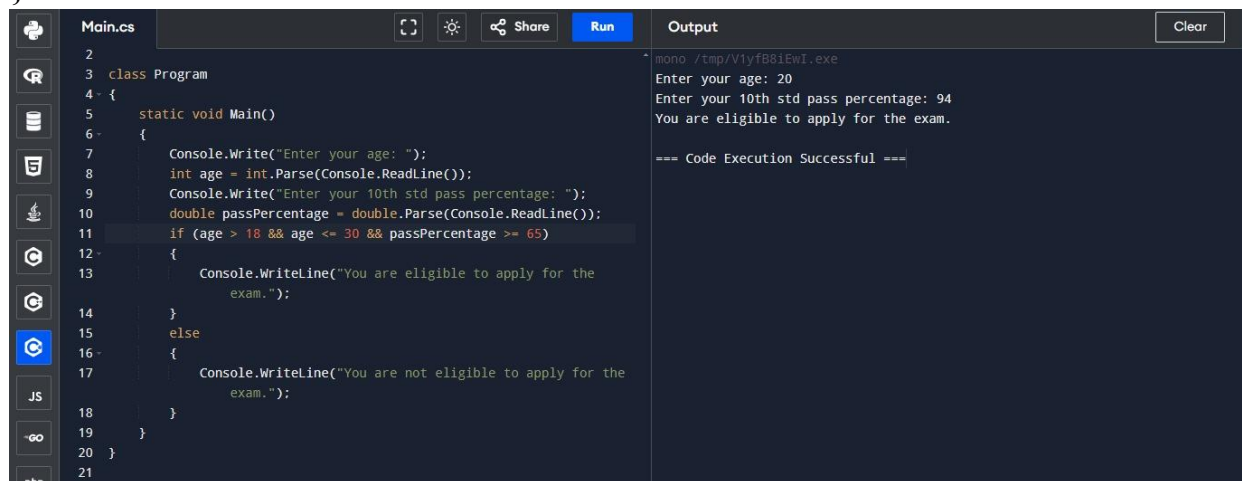
```
{
```

```
    static void Main()
```

```

{
    Console.Write("Enter your age: ");
    int age = int.Parse(Console.ReadLine());
    Console.Write("Enter your 10th std pass percentage: ");
    double passPercentage = double.Parse(Console.ReadLine());
    if (age > 18 && age <= 30 && passPercentage >= 65)
    {
        Console.WriteLine("You are eligible to apply for the exam.");
    }
    else
    {
        Console.WriteLine("You are not eligible to apply for the exam.");
    }
}
}

```



The screenshot shows a code editor with a file named `Main.cs`. The code is as follows:

```

2
3 class Program
4 {
5     static void Main()
6     {
7         Console.Write("Enter your age: ");
8         int age = int.Parse(Console.ReadLine());
9         Console.Write("Enter your 10th std pass percentage: ");
10        double passPercentage = double.Parse(Console.ReadLine());
11        if (age > 18 && age <= 30 && passPercentage >= 65)
12        {
13            Console.WriteLine("You are eligible to apply for the
exam.");
14        }
15        else
16        {
17            Console.WriteLine("You are not eligible to apply for the
exam.");
18        }
19    }
20 }
21

```

The `Run` button is highlighted in blue. The `Output` window on the right shows the following text:

```

mono: /tmp/V1yfb8iEwI.exe
Enter your age: 20
Enter your 10th std pass percentage: 94
You are eligible to apply for the exam.

=== Code Execution Successful ===

```

## Input:

Enter your age: 20

Enter your 10th std pass percentage: 94

## Output:

You are eligible to apply for the exam.

**3. Design the C# console application named validation to get mobile number as input from the user. Validate the mobile number with the following cases:**

- **The first four number must be followed by then followed by next six numbers(eg:9894-256874)**
- **Should contains only numbers**
- **Should be of length 10.**
- **The first number should start only with 9 Or 8.**

**Aim:**

To design a C# console application named Validation to validate the mobile number

**Program:**

```
using System;
using System.Text.RegularExpressions;
class Validation
{
    static void Main()
    {
        Console.Write("Enter your mobile number in the format 'XXXX_XXXXXX'
(e.g., 9894_256874): ");
        string mobileNumber = Console.ReadLine();
        string pattern = @"^[98]\d{3}[_]\d{6}$";
        if (Regex.IsMatch(mobileNumber, pattern))
        {
            Console.WriteLine("Mobile number is valid.");
        }
        else
        {
            Console.WriteLine("Mobile number is invalid. Please check the format.");
        }
    }
}
```

```
1 using System;
2 using System.Text.RegularExpressions;
3 class Validation
4 {
5     static void Main()
6     {
7         Console.WriteLine("Enter your mobile number in the format
8             'XXXX_XXXXXX' (e.g., 9894_256874): ");
9         string mobileNumber = Console.ReadLine();
10        string pattern = @"^[98]\d{3}[_]\d{6}$";
11        if (Regex.IsMatch(mobileNumber, pattern))
12        {
13            Console.WriteLine("Mobile number is valid.");
14        }
15        else
16        {
17            Console.WriteLine("Mobile number is invalid. Please
18                check the format.");
19        }
20    }
}
```

Output

```
mono /tmp/HwInngAyN9.exe
Enter your mobile number in the format 'XXXX_XXXXXX' (e.g., 9894_256874):
7890_456789
Mobile number is invalid. Please check the format.

=== Code Execution Successful ===
```

### Input:

Enter your mobile number in the format 'XXXX\_XXXXXX' (e.g., 9894\_256874):  
7890\_456789

### Output:

Mobile number is invalid. Please check the format.

**4. Write the missing code snippets and the statements in the C# program given below.**

**Class person {**

**\_\_\_\_\_name;**

**\_\_\_\_\_age;**

**\_\_\_\_\_weight;**

**Void printperson() {**

**// write the code to print name, age and weight of a person**

**}**

**}**

**Class persondata {**

**Static void Main(string[] args) {**

**person\_\_\_\_\_ = \_\_\_\_\_;**

**\_\_\_\_\_.name = "Kannan";**

**\_\_\_\_\_.age = 19;**

```
_____.weight = 58;  
// write the statement to access printperson() function  
}  
}
```

**Aim:**

To write the missing code snippets and the statements in the C# program

**Program:**

```
using System;  
class Person  
{  
    public string name;  
    public int age;  
    public float weight;  
    public void PrintPerson(){  
        Console.WriteLine("Name: " + name);  
        Console.WriteLine("Age: " + age);  
        Console.WriteLine("Weight: " + weight + " kg");  
    }  
}  
class PersonData{  
    static void Main(string[] args){  
        Person person = new Person();  
        person.name = "Kannan";  
        person.age = 19;  
        person.weight = 58;  
        person.PrintPerson();  
    }  
}
```

```

}
Main.cs
1 using System;
2 class Person
3 {
4     public string name;
5     public int age;
6     public float weight;
7     public void PrintPerson(){
8         Console.WriteLine("Name: " + name);
9         Console.WriteLine("Age: " + age);
10        Console.WriteLine("Weight: " + weight + " kg");
11    }
12 }
13 class PersonData{
14     static void Main(string[] args){
15         Person person = new Person();
16         person.name = "Kannan";
17         person.age = 19;
18         person.weight = 58;
19         person.PrintPerson();
20     }
21 }
22
Output
mono /tmp/H1JE4s0rkb.exe
Name: Kannan
Age: 19
Weight: 58 kg

=== Code Execution Successful ===

```

## Output:

Name: Kannan

Age: 19

Weight: 58 kg

**5. A hospital wants to create a console application to maintain its impatient details. The information to store includes:**

- Name of the patient
- Date of admission
- Age of patient
- Disease
- Date of discharge
- Total bills paid

**Design the C# program with the class name patient with necessary data members to store the above information. The class should have two member functions, one to get the patients information and other to display the information. Create a main class called hospital to create necessary instances, methods calling statements and display all the details about the patient.**

## Aim:

To design the C# program with the class name patient with necessary data members to store the information.

**Program:**

```
using System;
class Patient
{
    public string name;
    public string dateOfAdmission;
    public int age;
    public string disease;
    public string dateOfDischarge;
    public double totalBillsPaid;
    public void GetPatientInfo(){
        Console.Write("Enter patient name: ");
        name = Console.ReadLine();
        Console.Write("Enter date of admission (dd/mm/yyyy): ");
        dateOfAdmission = Console.ReadLine();
        Console.Write("Enter age: ");
        age = int.Parse(Console.ReadLine());
        Console.Write("Enter disease: ");
        disease = Console.ReadLine();
        Console.Write("Enter date of discharge (dd/mm/yyyy): ");
        dateOfDischarge = Console.ReadLine();
        Console.Write("Enter total bills paid: ");
        totalBillsPaid = double.Parse(Console.ReadLine());
    }
    public void DisplayPatientInfo(){
        Console.WriteLine("\nPatient Details:");
        Console.WriteLine("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();
    }
}

```

The screenshot shows a Visual Studio IDE with a C# project. The code defines a `Patient` class with properties for name, date of admission, age, disease, date of discharge, and total bills paid. It also defines a `Hospital` class with a `Main` method that creates a `Patient` object and calls its `GetPatientInfo` and `DisplayPatientInfo` methods. The `DisplayPatientInfo` method formats and prints the patient's details.

The Output window on the right shows the program's execution. It displays the prompts for patient information, the user's input, and the formatted output of the patient's details. The output is as follows:

```

Enter patient name: Rangini
Enter date of admission (dd/mm/yyyy): 20/02/2024
Enter age: 43
Enter disease: Diabetics
Enter date of discharge (dd/mm/yyyy): 26/02/2024
Enter total bills paid: 4000

Patient Details:
Name: Rangini
Date of Admission: 20/02/2024
Age: 43
Disease: Diabetics
Date of Discharge: 26/02/2024
Total Bills Paid: 4000

--- Code Execution Successful ---

```

## Input:

Enter patient name: Rangini

Enter date of admission (dd/mm/yyyy): 20/02/2024

Enter age: 43

Enter disease: Diabetics

Enter date of discharge (dd/mm/yyyy): 26/02/2024

Enter total bills paid: 4000

**Output:**

Patient Details:

Name: Rangini

Date of Admission: 20/02/2024

Age: 43

Disease: Diabetics

Date of Discharge: 26/02/2024

Total Bills Paid: 4000

**6.Implement the C# code to get two vector number as input, add them and print the sum as another vector. Make use of operator overloading to perform addition of vector numbers.**

**Aim:**

To implement a C# program that uses operator overloading to add two vector numbers

**Program:**

```
using System;
```

```
class Vector{
```

```
    public int x, y, z;
```

```
    public Vector(int x, int y, int z)
```

```
    {
```

```
        this.x = x;
```

```
        this.y = y;
```

```
        this.z = z;
```

```
    }
```

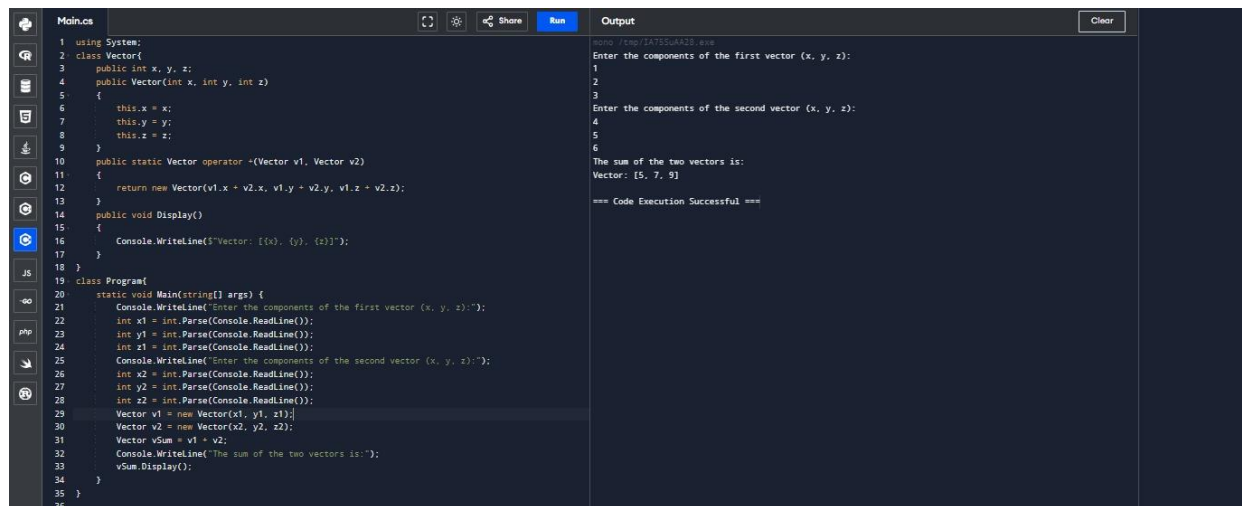
```
    public static Vector operator +(Vector v1, Vector v2)
```

```
    {
```

```

        return new Vector(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z);
    }
    public void Display()
    {
        Console.WriteLine($"Vector: [{x}, {y}, {z}]");
    }
}
class Program{
    static void Main(string[] args) {
        Console.WriteLine("Enter the components of the first vector (x, y, z):");
        int x1 = int.Parse(Console.ReadLine());
        int y1 = int.Parse(Console.ReadLine());
        int z1 = int.Parse(Console.ReadLine());
        Console.WriteLine("Enter the components of the second vector (x, y, z):");
        int x2 = int.Parse(Console.ReadLine());
        int y2 = int.Parse(Console.ReadLine());
        int z2 = int.Parse(Console.ReadLine());
        Vector v1 = new Vector(x1, y1, z1);
        Vector v2 = new Vector(x2, y2, z2);
        Vector vSum = v1 + v2;
        Console.WriteLine("The sum of the two vectors is:");
        vSum.Display();
    }
}

```



```
1 using System;
2 class Vector{
3     public int x, y, z;
4     public Vector(int x, int y, int z)
5     {
6         this.x = x;
7         this.y = y;
8         this.z = z;
9     }
10    public static Vector operator +(Vector v1, Vector v2)
11    {
12        return new Vector(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z);
13    }
14    public void Display()
15    {
16        Console.WriteLine($"Vector: [{x}, {y}, {z}]");
17    }
18 }
19 class Program
20 {
21     static void Main(string[] args) {
22         Console.WriteLine("Enter the components of the first vector (x, y, z):");
23         int x1 = int.Parse(Console.ReadLine());
24         int y1 = int.Parse(Console.ReadLine());
25         int z1 = int.Parse(Console.ReadLine());
26         Console.WriteLine("Enter the components of the second vector (x, y, z):");
27         int x2 = int.Parse(Console.ReadLine());
28         int y2 = int.Parse(Console.ReadLine());
29         int z2 = int.Parse(Console.ReadLine());
30         Vector v1 = new Vector(x1, y1, z1);
31         Vector v2 = new Vector(x2, y2, z2);
32         Vector vSum = v1 + v2;
33         Console.WriteLine("The sum of the two vectors is:");
34         vSum.Display();
35     }
36 }
```

Output

```
Enter the components of the first vector (x, y, z):
1
2
3
Enter the components of the second vector (x, y, z):
4
5
6
The sum of the two vectors is:
Vector: [5, 7, 9]
=== Code Execution Successful ===
```

## Input:

Enter the components of the first vector (x, y, z):

1

2

3

Enter the components of the second vector (x, y, z):

4

5

6

## Output:

The sum of the two vectors is:

Vector: [5, 7, 9]

**7. Create the class student with necessary members to maintain the basic details of a student such as name, age, address and mobile number. Add method getDate() to read the basic details and printData() to print the details of the student. Inherit the student class into the sub class called studentmark with necessary members to maintain student mark details. Override the getDate() and printData() in student mark class to read mark details and print the marks, respectively. Also, define a method to find the grade of the student based on his/her marks. Design the student main class to access the member of both the classes in C#.**

**Aim:**

To create a C# program that uses inheritance and method overriding to manage basic student details along with student marks and grades.

**Program:**

```
using System;

class Student
{
    public string name;
    public int age;
    public string address;
    public string mobileNumber;
    public virtual void GetData()
    {
        Console.Write("Enter student name: ");
        name = Console.ReadLine();
        Console.Write("Enter student age: ");
        age = int.Parse(Console.ReadLine());
        Console.Write("Enter student address: ");
        address = Console.ReadLine();
        Console.Write("Enter student mobile number: ");
        mobileNumber = Console.ReadLine();
    }
    public virtual void PrintData()
    {
        Console.WriteLine("\nStudent Details:");
        Console.WriteLine($"Name: {name}");
        Console.WriteLine($"Age: {age}");
        Console.WriteLine($"Address: {address}");
        Console.WriteLine($"Mobile Number: {mobileNumber}");
    }
}
```

```

}
class StudentMark : Student
{
    public int mark1, mark2, mark3;
    public override void GetData()
    {
        base.GetData();
        Console.Write("Enter marks for subject 1: ");
        mark1 = int.Parse(Console.ReadLine());
        Console.Write("Enter marks for subject 2: ");
        mark2 = int.Parse(Console.ReadLine());
        Console.Write("Enter marks for subject 3: ");
        mark3 = int.Parse(Console.ReadLine());
    }
    public override void PrintData()
    {
        base.PrintData();
        Console.WriteLine("\nMarks Details:");
        Console.WriteLine($"Subject 1: {mark1}");
        Console.WriteLine($"Subject 2: {mark2}");
        Console.WriteLine($"Subject 3: {mark3}");
        Console.WriteLine("Grade: " + CalculateGrade());
    }
    public string CalculateGrade(){
        int total = mark1 + mark2 + mark3;
        double average = total / 3.0;

        if (average >= 90)
            return "O";
        else if (average >= 80)

```

```

        return "A";
    else if (average >= 70)
        return "B";
    else if (average >= 60)
        return "C";
    else if (average >= 50)
        return "D";
    else
        return "F";
    }
}

class StudentMain{
    static void Main(string[] args){
        StudentMark student = new StudentMark();

        student.GetData();

        student.PrintData();
    }
}

```

The screenshot shows a C# IDE with a file named 'Main.cs'. The code defines a 'Student' class with properties for name, age, address, and mobile number, and methods for getting and printing data. It also defines a 'StudentMark' class that inherits from 'Student' and has properties for marks in three subjects and a grade. The 'Main' method creates a 'StudentMark' object, calls 'GetData()' to input student details, and 'PrintData()' to display them.

**Output:**

```

C:\Program Files\dotnet\cli\dotnet.exe /tmp/tdGJELtIgF.exe
Enter student name: Siva
Enter student age: 20
Enter student address: Metu theru,Rasipuram
Enter student mobile number: 9080605070
Enter marks for subject 1: 98
Enter marks for subject 2: 98
Enter marks for subject 3: 97

Student Details:
Name: Siva
Age: 20
Address: Metu theru,Rasipuram
Mobile Number: 9080605070

Marks Details:
Subject 1: 98
Subject 2: 98
Subject 3: 97
Grade: 0

=== Code Execution Successful ===

```

## Input:

Enter student name: Siva

Enter student age: 20

Enter student address: Metu theru,Rasipuram

Enter student mobile number: 9080605070

Enter marks for subject 1: 98

Enter marks for subject 2: 98

Enter marks for subject 3: 97

### **Output:**

Student Details:

Name: Siva

Age: 20

Address: Metu theru,Rasipuram

Mobile Number: 9080605070

Marks Details:

Subject 1: 98

Subject 2: 98

Subject 3: 97

Grade: O

**8. Design sample C# program with class name employee to compute netsalary of the employee using the basic salary, if for the job\_catg is 1 use table-I else use table-II. Use constructor to initialize basic salary,hra,da,pf and loan. The employee class should contain input() method to get input for job\_catg, empno, empname, calculateSalary() method to compute salary and display() method to print the details.**

Table-I	Table-II
<b>BASIC=Rs. 8,000</b> <b>HRA=10% of basic</b> <b>DA=20% of basic</b> <b>LOAN=Rs. 300</b> <b>PF=Rs. 500</b>	<b>BASIC=Rs. 15,000</b> <b>HRA=20% of basic</b> <b>DA=30% of basic</b> <b>LOAN=Rs. 600</b> <b>PF=1000</b>



**Aim:**

To design a C# program that computes the net salary of an employee based on their job category

**Program:**

```
using System;
```

```
class Employee
```

```
{
```

```
    public int empNo;
```

```
    public string empName;
```

```
    public int jobCatg;
```

```
    public double basicSalary, hra, da, pf, loan;
```

```
    public double netSalary;
```

```
    public Employee(double basicSalary)
```

```
    {
```

```
        this.basicSalary = basicSalary;
```

```
        hra = da = pf = loan = 0;
```

```
    }
```

```
    public void Input()
```

```
    {
```

```
        Console.Write("Enter employee number: ");
```

```
        empNo = int.Parse(Console.ReadLine());
```

```
        Console.Write("Enter employee name: ");
```

```
        empName = Console.ReadLine();
```

```
        Console.Write("Enter job category (1 for Table-I, 2 for Table-II): ");
```

```
        jobCatg = int.Parse(Console.ReadLine());
```

```
    }
```

```
    public void CalculateSalary()
```

```
{
    if (jobCatg == 1)
    {
        basicSalary = 8000;
        hra = 0.10 * basicSalary;
        da = 0.20 * basicSalary;
        loan = 300;
        pf = 500;
    }
    else if (jobCatg == 2)
    {
        basicSalary = 15000;
        hra = 0.20 * basicSalary;
        da = 0.30 * basicSalary;
        loan = 600;
        pf = 1000;
    }
    else
    {
        Console.WriteLine("Invalid job category entered!");
        return;
    }
    netSalary = basicSalary + hra + da - (pf + loan);
}

public void Display()
{
    Console.WriteLine("\nEmployee Details:");
    Console.WriteLine("Employee Number: " + empNo);
    Console.WriteLine("Employee Name: " + empName);
}
```

```

        Console.WriteLine("Job Category: " + (jobCatg == 1 ? "Table-I" : "Table-II"));
        Console.WriteLine("Basic Salary: " + basicSalary);
        Console.WriteLine("HRA: " + hra);
        Console.WriteLine("DA: " + da);
        Console.WriteLine("Loan Deduction: " + loan);
        Console.WriteLine("PF Deduction: " + pf);
        Console.WriteLine("Net Salary: " + netSalary);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Employee emp = new Employee(0);
        emp.Input();
        emp.CalculateSalary();
        emp.Display();
    }
}

```

The screenshot shows a C# IDE with two panes. The left pane displays the source code for `Main.cs`, which includes the `Employee` class and the `Program` class. The right pane shows the output of the program execution.

```

Main.cs
1 using System;
2 class Employee
3 {
4     public int empNo;
5     public string empName;
6     public int jobCatg;
7     public double basicSalary, hra, da, pf, loan;
8     public double netSalary;
9     public Employee(double basicSalary)
10    {
11        this.basicSalary = basicSalary;
12        hra = da = pf = loan = 0;
13    }
14    public void Input()
15    {
16        Console.Write("Enter employee number: ");
17        empNo = int.Parse(Console.ReadLine());
18
19        Console.Write("Enter employee name: ");
20        empName = Console.ReadLine();
21    }
22 }

```

```

Output
mono: /tmp/3x0CbSPpP.exe
Enter employee number: 43
Enter employee name: Sakthi
Enter job category (1 for Table-I, 2 for Table-II): 2

Employee Details:
Employee Number: 43
Employee Name: Sakthi
Job Category: Table-II
Basic Salary: 15000
HRA: 3000
DA: 4500
Loan Deduction: 600
PF Deduction: 1000
Net Salary: 20900

=== Code Execution Successful ===

```

## Input:

Enter employee number: 43

Enter employee name: Sakthi

Enter job category (1 for Table-I, 2 for Table-II): 2

**Output:**

Employee Details:

Employee Number: 43

Employee Name: Sakthi

Job Category: Table-II

Basic Salary: 15000

HRA: 3000

DA: 4500

Loan Deduction: 600

PF Deduction: 1000

Net Salary: 20900

**BY:**

**SHOBICA R**

**73772214206**

**III – B.E CSE**