NAME : VINOTHINI B

REG NO : 73772214222

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()
    {
        int[,] array = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

        for (int i = 0; i < array.GetLength(0); i++)
        {
            for (int j = 0; j < array.GetLength(1); j++)
            {
                Console.Write(array[i, j] + " ");
            }
            Console.WriteLine();
        }
    }
}
```

Input

```
{
   { 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()

   {

      int age = 20;  // example input

      double percentage = 70;  // example input


      if (age > 18 && age <= 30 && percentage >= 65)

      {

         Console.WriteLine("Eligible");

      }

      else

      {

         Console.WriteLine("Not Eligible");

      }

   }

}
```

Input

- age: 20
- percentage: 70

Output

```
   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

```
using System;

class Program

{

   static void Main()

   {

       Console.WriteLine("Enter mobile number:");

       string mobile = Console.ReadLine();


       if (mobile.Length == 10 && (mobile.StartsWith("9") || mobile.StartsWith("8")) &&
long.TryParse(mobile, out _))

       {

           Console.WriteLine("Valid mobile number");

       }

       else

       {

           Console.WriteLine("Invalid mobile number");

       }

   }

}
```

Input

9876543210

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

```
class Person
{
    public string name;
    public int age;
    public float weight;

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

class PersonData
{
    static void Main(string[] args)
    {
        Person person = new Person();
        person.name = "Kannan";
        person.age = 19;
        person.weight = 60.5f;  // Example weight

        person.PrintPerson();
    }
}
```

Input

- name: "Kannan"
- age: 19
- weight: 60.5f

Output

```
Kannan, Age: 19, Weight: 60.5
```

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;

    public DateTime dateOfAdmission;

    public int age;

    public string disease;

    public DateTime dateOfDischarge;

    public double totalBillsPaid;


    public void GetPatientInfo()

    {

        Console.WriteLine("Enter patient details:");

        Console.Write("Name: ");

        name = Console.ReadLine();

        Console.Write("Date of Admission (dd/mm/yyyy): ");

        dateOfAdmission = DateTime.Parse(Console.ReadLine());

        Console.Write("Age: ");

        age = int.Parse(Console.ReadLine());

        Console.Write("Disease: ");

        disease = Console.ReadLine();

        Console.Write("Date of Discharge (dd/mm/yyyy): ");

        dateOfDischarge = DateTime.Parse(Console.ReadLine());

        Console.Write("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:dd/MM/yyyy}");
        Console.WriteLine($"Age: {age}");
        Console.WriteLine($"Disease: {disease}");
        Console.WriteLine($"Date of Discharge: {dateOfDischarge:dd/MM/yyyy}");
        Console.WriteLine($"Total Bills Paid: {totalBillsPaid}");
    }
}

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

Input

- name: Anu
- dateOfAdmission: 23/2/2005
- disease: fever
- dateOfDischarge: 24/2/2005

- totalBillsPaid: 2,000

Output

- name: Anu
- dateOfAdmission: 23/2/2005
- disease: fever
- dateOfDischarge: 24/2/2005
- totalBillsPaid: 2,000

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

```
using System;

class Vector

{

   public int x, y;


   public Vector(int x, int y)

   {

      this.x = x;

      this.y = y;

   }


   // Overloading the + operator

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

   {

      return new Vector(v1.x + v2.x, v1.y + v2.y);

   }


   public void Display()

   {

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

   }

}


class Program
```

```
{
   static void Main()
   {
      Vector v1 = new Vector(2, 3);
      Vector v2 = new Vector(4, 5);

      Vector v3 = v1 + v2;

      v1.Display();
      v2.Display();
      Console.WriteLine("Sum of vectors:");
      v3.Display();
   }
}
```

Input

Two Vector objects:

v1: (2, 3).

v2: (4, 5).

Output

```
V3: (6,8)
```

Resullt

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

```
using System;

class Student

{

  public string name;

  public string address;

  public string mobileNumber;


  public void GetData()

  {

    Console.Write("Enter name: ");

    name = Console.ReadLine();

    Console.Write("Enter address: ");

    address = Console.ReadLine();

    Console.Write("Enter mobile number: ");

    mobileNumber = Console.ReadLine();

  }


  public virtual void PrintData()

  {

    Console.WriteLine($"Name: {name}");

    Console.WriteLine($"Address: {address}");

    Console.WriteLine($"Mobile Number: {mobileNumber}");

  }

}
```

```
class StudentMark : Student
{
   public int marks;

   public override void GetData()
   {
      base.GetData();
      Console.Write("Enter marks: ");
      marks = int.Parse(Console.ReadLine());
   }

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

class Program
{
   static void Main()
   {
      StudentMark student = new StudentMark();
      student.GetData();
      student.PrintData();
   }
}
```

Input

- **From User:**
  - For Student class:
    - name: Vinothini
    - address: Tiruchengode,Namakal
    - mobileNumber: 9234567890
  - For StudentMark class (in addition to Student data):
    - marks: 599

Output

- name: Vinothini
- address: Tiruchengode,Namakal
- mobileNumber: 9234567890
- marks: 599

Result

Thus the program 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

```
using System;

class Employee

{

    public int jobCatg;

    public string empName;

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


    public void Input()

    {

        Console.Write("Enter job category (1 or 2): ");

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

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

        empName = Console.ReadLine();


        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;
        }
    }

    public double CalculateSalary()
    {
        return (basicSalary + hra + da) - (loan + pf);
    }

    public void Display()
    {
        Console.WriteLine($"Employee Name: {empName}");
        Console.WriteLine($"Basic Salary: {basicSalary}");
        Console.WriteLine($"HRA: {hra}");
        Console.WriteLine($"DA: {da}");
        Console.WriteLine($"Loan: {loan}");
        Console.WriteLine($"PF: {pf}");
        Console.WriteLine($"Net Salary: {CalculateSalary()}");
    }
}

class Program
{
    static void Main()
    {
        Employee employee = new Employee();
        employee.Input();
        employee.Display();
    }
}
```

Input

```
Job Category: 1
Employee Name: Alice
```

Output

```
Employee Name: Alice
Basic Salary: 8000
HRA: 800
DA: 1600
Loan: 300
PF: 500
Net Salary: 9600
```

Result

Thus the program executed successfully.