Course Code: P15ISL57	Semester : V	L- T – P : 0 – 0 - 3	Credit: 1.5
Course Title: C# Programming & .Net Lab			
Contact period : Lecture : 39 Hrs, Exam: 3Hrs		Weightage: CIE:50; SEE:50	

Contents

1 a) Write a C# program to check whether a number is Palindrome or not.

b) Write a C# program to demonstrate command line arguments processing.

2 a) Write a C# program to find the roots of a Quadratic Equation.

b) Write a C# program to demonstrate Boxing and unBoxing.

3 a) Write a C# program to implement Stack of integers.

b) Write a C# program to demonstrate Operator overloading.

4 a) Write a C# program to find the second largest element in a single dimensional array.

b) Write a C# program to multiply two matrices using Rectangular arrays.

5 a) Write a C# program to find the sum of all the elements present in a jagged array of 3 inner arrays.

b) Design a simple calculator using switch statement in C#.

6 a) Demonstrate the use of virtual and override keyword in C# with a simple Program.

b) Implement Linked Lists in C# using the existing collections name space.

7 a) Write a C# program to demonstrate abstract class and abstract methods.

b) Write a C# program to build a class which implements an existing interface.

8 a) Write a C# program to illustrate the use of different properties.

b) Demonstrate arrays of interface types with a C# program.

9 a) Write a C# program to illustrate the creation of a dll file and then using it in a program.

b) Write a C# program to illustrate declaring, instantiating, and using a delegate.

Note:

1) In SEE, student has to execute any ONE full program out of NINE programs compulsorily.

2) In case of every program, sub-program a) has to be executed without using VS .NET IDE and sub-program b) has to be executed using VS .NET IDE.

Course Outcomes

The student is able to

CO1: Create Configuration for a given machine to host the .NET runtime.

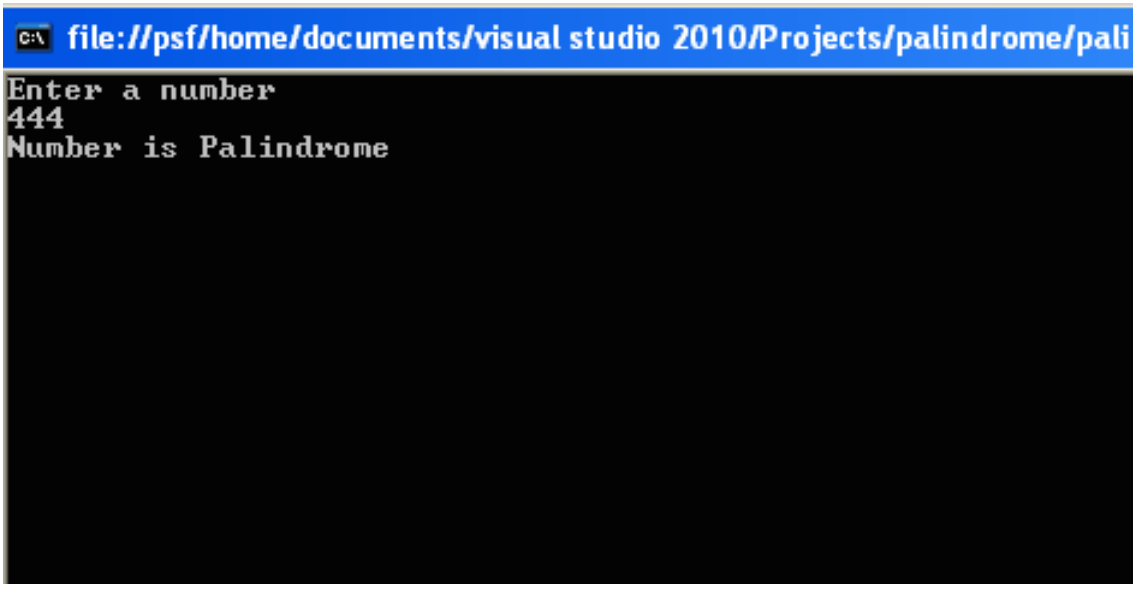
CO2: Develop and debug C# programs with well understanding of C# language constructs either by using VS .NET IDE or Microsoft .NET SDK.

CO3: Develop and use .NET assemblies.

1 a) Write a C# program to check whether a number is Palindrome or not.

```
using System;
class palindrome
{
        public static void Main()
        {
          int num=0,rev,num1=0,num2=0;
          Console.WriteLine("Enter a number");
          num=int.Parse(Console.ReadLine());
          num2=num;
          while(num>0)
          {
                rev=num%10;
                num=num/10;
                num1=num1*10+rev;
          }
        if(num1==num2)
                Console.WriteLine("Number is Palindrome");
        else
                Console.WriteLine("Number is NOT Palindrome");
        }
}
```

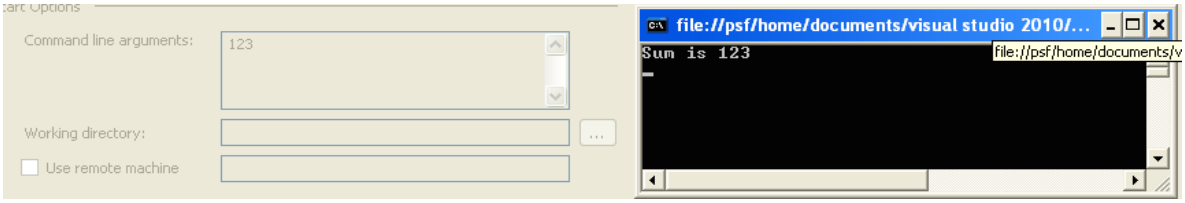
OUTPUT:



1 b) Write a C# program to demonstrate Command line arguments processing.

```
using System;
class cmdarg
{
        public static void Main()
        {
                int num=0;
                String[ ] argument=Environment.GetCommandLineArgs();
                for(inti=1;i<argument.Length;i++)
                        num=num+int.Parse(argument[i]);
                Console.WriteLine("Sum is "+ num);
        }
}
```

OUTPUT:



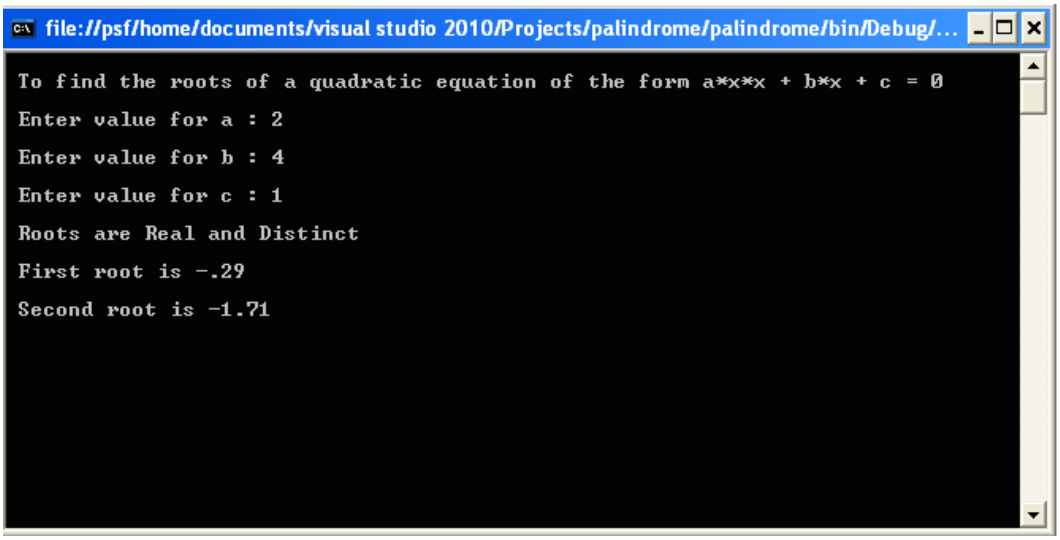
2 a) Write a C# program to find the roots of a Quadratic Equation.

```
using System;
class Quadraticroots
{
   double a, b, c;
    public void read()
   {
     Console.WriteLine(" \n To find the roots of a quadratic equation of the form a*x*x + b*x + c = 0");
     Console.Write("\n Enter value for a : ");
      a = double.Parse(Console.ReadLine());
     Console.Write("\n Enter value for b : ");
      b = double.Parse(Console.ReadLine());
      Console.Write("\n Enter value for c : ");
      c = double.Parse(Console.ReadLine());
    }
```

```
public void compute()
{
    int m;
    double r1, r2, d1;
    d1 = b * b - 4 * a * c;
    if (a == 0)
       m = 1;
    else if (d1 > 0)
       m = 2;
    else if (d1 == 0)
       m = 3;
    else
       m = 4;
    switch (m)
    {
    case 1: Console.WriteLine("\n Not a Quadratic equation,     Linear equation");
            Console.ReadLine();
            break;
    case 2: Console.WriteLine("\n Roots are Real and Distinct");
            r1 = (-b + Math.Sqrt(d1)) / (2 * a);
            r2 = (-b - Math.Sqrt(d1)) / (2 * a);
            Console.WriteLine("\n First root is {0:#.##}", r1);
            Console.WriteLine("\n Second root is {0:#.##}", r2);
            Console.ReadLine();
            break;
    case 3: Console.WriteLine("\n Roots are Real and Equal");
             r1 = r2 = (-b) / (2 * a);
            Console.WriteLine("\n First root is {0:#.##}", r1);
            Console.WriteLine("\n Second root is {0:#.##}", r2);
            Console.ReadLine();
            break;
```

```
            case 4: Console.WriteLine("\n Roots are Imaginary");
                    r1 = (-b) / (2 * a);
                    r2 = Math.Sqrt(-d1) / (2 * a);
                    Console.WriteLine("\n First root is {0:#.##} + i {1:#.##}", r1, r2);
                    Console.WriteLine("\n Second root is {0:#.##} - i {1:#.##}", r1, r2);
                    Console.ReadLine();
                    break;
            }
        }
    }
class Roots
{
    public static void Main()
    {
      Quadraticrootsqr = new Quadraticroots();
       qr.read();
       qr.compute();
      }
    }
```

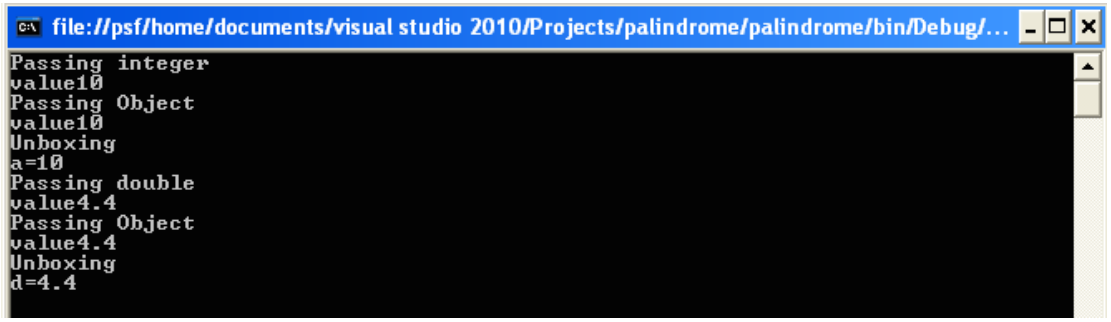
OUTPUT:



2 b) Write a C# program to demonstrate Boxing and unBoxing.

```
using System;
class demo
{
    static void box(object obj)
   {
       Console.WriteLine("value" + obj);
   }
   public static void Main()
   {
      Object o;
      int a = 10;
      double d = 4.4;
      o = a; //boxing integer
      Console.WriteLine("Passing integer");
      box(a);
      Console.WriteLine("Passing Object");
      box(o);
      int x = (int)o;//Unboxing
     Console.WriteLine("Unboxing");
     Console.WriteLine("a=" + x);
     o = d; //boxing double
     Console.WriteLine("Passing double");
     box(d);
     Console.WriteLine("Passing Object");
     box(o);
     double dd = (double)o; //Unboxing
     Console.WriteLine("Unboxing");
     Console.WriteLine("d=" + dd);
   }
}
```

OUTPUT:



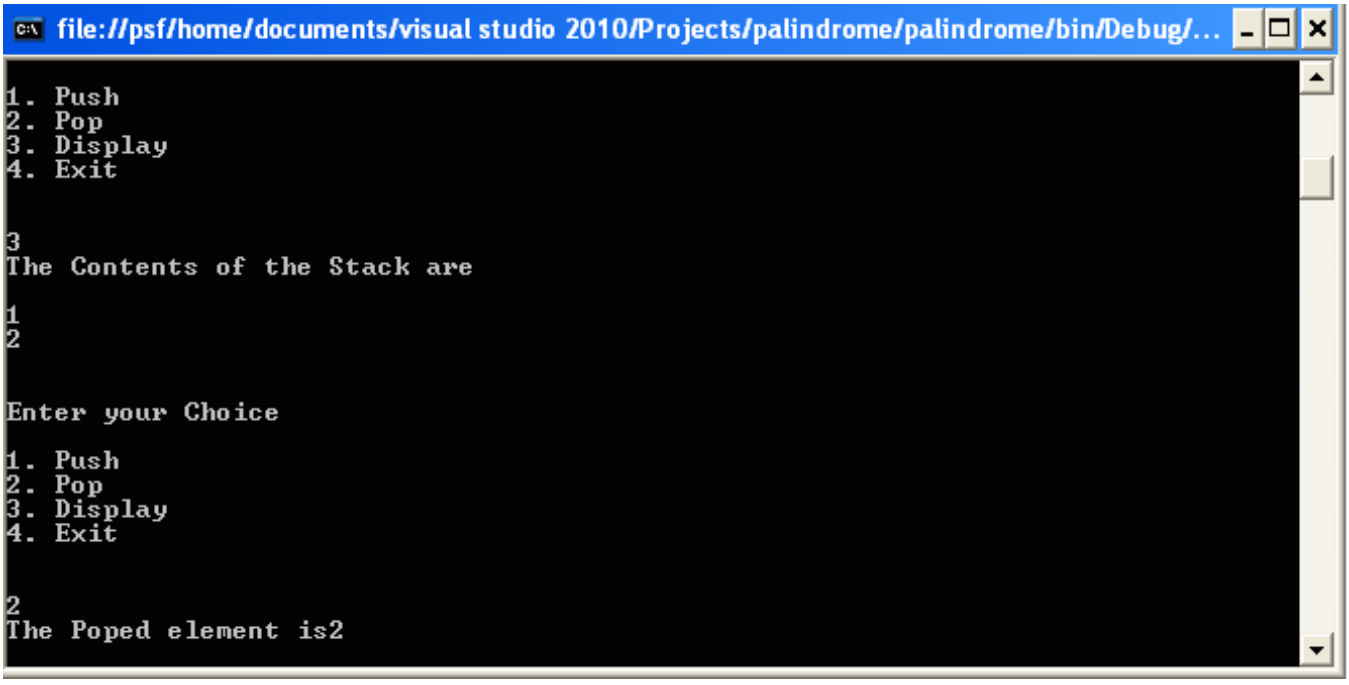
3 a) Write a C# program to implement Stack of integers.

```
using System;
class stack
{
    int top;
    int[ ] s;
    public stack(int size)
    {
       s = new int[size];
      top = -1;
    }
   public stack( ) { }
   public void pop( )
   {
        if (top == -1)
        {
          Console.WriteLine("No elements to Pop\n");
         return;
        }
        Console.WriteLine("The Poped element is" + s[top]);
        top--;
    }
public void push(int var)
{
        //Console.WriteLine("top = " + top);
         s[++top] = var;
}
public void display()
{
           Console.WriteLine("The Contents of the Stack are\n");
```

```
        if (top == -1)
        {
                Console.WriteLine("No elements to Display\n");
                return;
        }
        for (inti = 0; i<=top; i++)
                Console.WriteLine(s[i]);
   }
}

public class demo
{
  public static void Main( )
   {
       Console.WriteLine("Enter the Size of Stack\n");
       int size = int.Parse(Console.ReadLine( ));
      stack st = new stack(size);
      //st.init();
       int eflag = 0;
      do
      {
        Console.WriteLine("\n\nEnter your Choice\n");
        Console.WriteLine("1. Push");
        Console.WriteLine("2. Pop");
        Console.WriteLine("3. Display");
        Console.WriteLine("4. Exit\n\n");
        int ch = int.Parse(Console.ReadLine());
         switch (ch)
         {
              case 1: Console.WriteLine("Enter a Number to Push\n");
                      int  var = int.Parse(Console.ReadLine());
```

```
                    st.push(var);
                    break;
            case 2: st.pop( );
                    break;
            case 3: st.display( );
                    break;
             case 4: eflag=1;
                    break;
          }
      } while (eflag==0);
      }
}
```

OUTPUT:

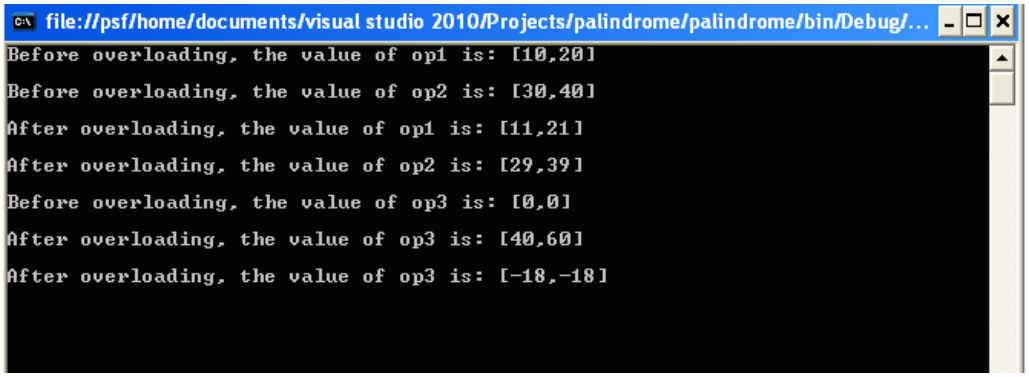


3 b) Write a C# program to demonstrate Operator overloading.

```
using System;
class OLoad
{
        int var1,var2;
        public OLoad(int a, int b)
        {
        var1=a;
        var2=b;
        }
        public OLoad()
        {
        }
        public static OLoad operator ++ (OLoad op1)
        {
                return new OLoad(op1.var1+1,op1.var2+1);
        }
        public static OLoad operator -- (OLoad op1)
        {
                return new OLoad(op1.var1-1,op1.var2-1);
        }
        public static OLoad operator + (OLoad op1,OLoad op2)
        {
                return new OLoad(op1.var1+op2.var1,op1.var2+op2.var2);
        }
        public static OLoad operator - (OLoad op1,OLoad op2)
        {
                return new OLoad(op1.var1-op2.var1,op1.var2-op2.var2);
         }
        public override string ToString()
        {
```

```
            return string.Format("[" + var1 + "," + var2 + "]");
        }
}
class OLoadMain
{
public static void Main()
  {
        OLoad op1=new OLoad(10,20);
        OLoad op2=new OLoad(30,40);
        OLoad op3=new OLoad();
        Console.WriteLine("Before overloading, the value of op1 is: " + op1 + "\n");
        Console.WriteLine("Before overloading, the value of op2 is: " + op2 + "\n");
        op1++;
        op2--;
        Console.WriteLine("After overloading, the value of op1 is: " + op1 + "\n");
        Console.WriteLine("After overloading, the value of op2 is: " + op2 + "\n");
        Console.WriteLine("Before overloading, the value of op3 is: " + op3 + "\n");
        op3=op1+op2;
        Console.WriteLine("After overloading, the value of op3 is: " + op3 + "\n");
        op3=op1-op2;
        Console.WriteLine("After overloading, the value of op3 is: " + op3 + "\n");
  }
}
```

OUTPUT:

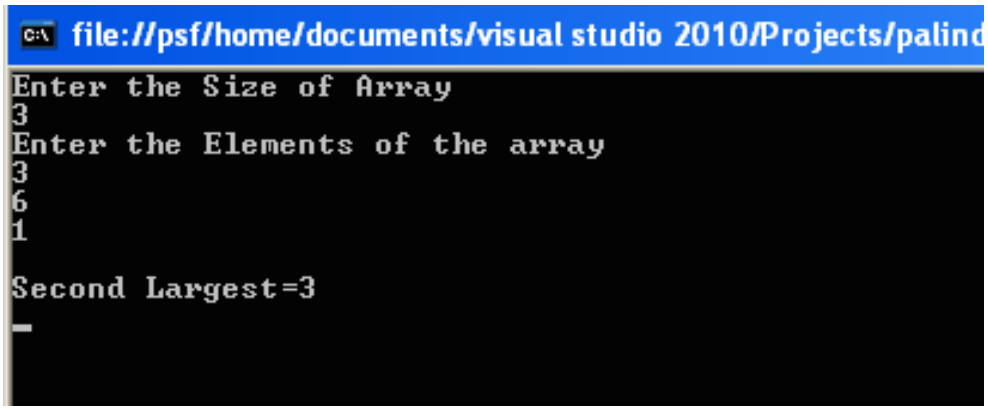


4 a) Write a C# program to find the second largest element in a single dimensional array.

```
using System;
class SLar
{
       int size;
       int[] nums;
       int lar, sec;
       public SLar(int n)
       {
          nums = new int[size = n];
       }
       public void input()
       {
          Console.WriteLine("Enter the Elements of the array");
          for (int i = 0; i < size; i++)
             nums[i] = int.Parse(Console.ReadLine());
       }
       public void second()
      {
          lar = nums[0];
         sec = nums[1];
         for (int i = 0; i< size; i++)
        {
              if (nums[i] >lar)
             {
                 sec = lar;
                 lar = nums[i];
             }
             else if ((nums[i] > sec && nums[i] < lar) || lar == sec)
            {
                 sec = nums[i];
             }
        }
       if (lar == sec)
            Console.Write("All are Equal");
       else
           Console.WriteLine("Second Largest=" + sec);
    }
  }
```

```
class SLarMain
{
      public static void Main()
      {
         Console.WriteLine("Enter the Size of Array");
         SLar s = new SLar(int.Parse(Console.ReadLine()));
         s.input();
         s.second();
         Console.Read();
      }
   }
```

OUTPUT:



4 b) Write a C# program to multiply two matrices using Rectangular arrays.

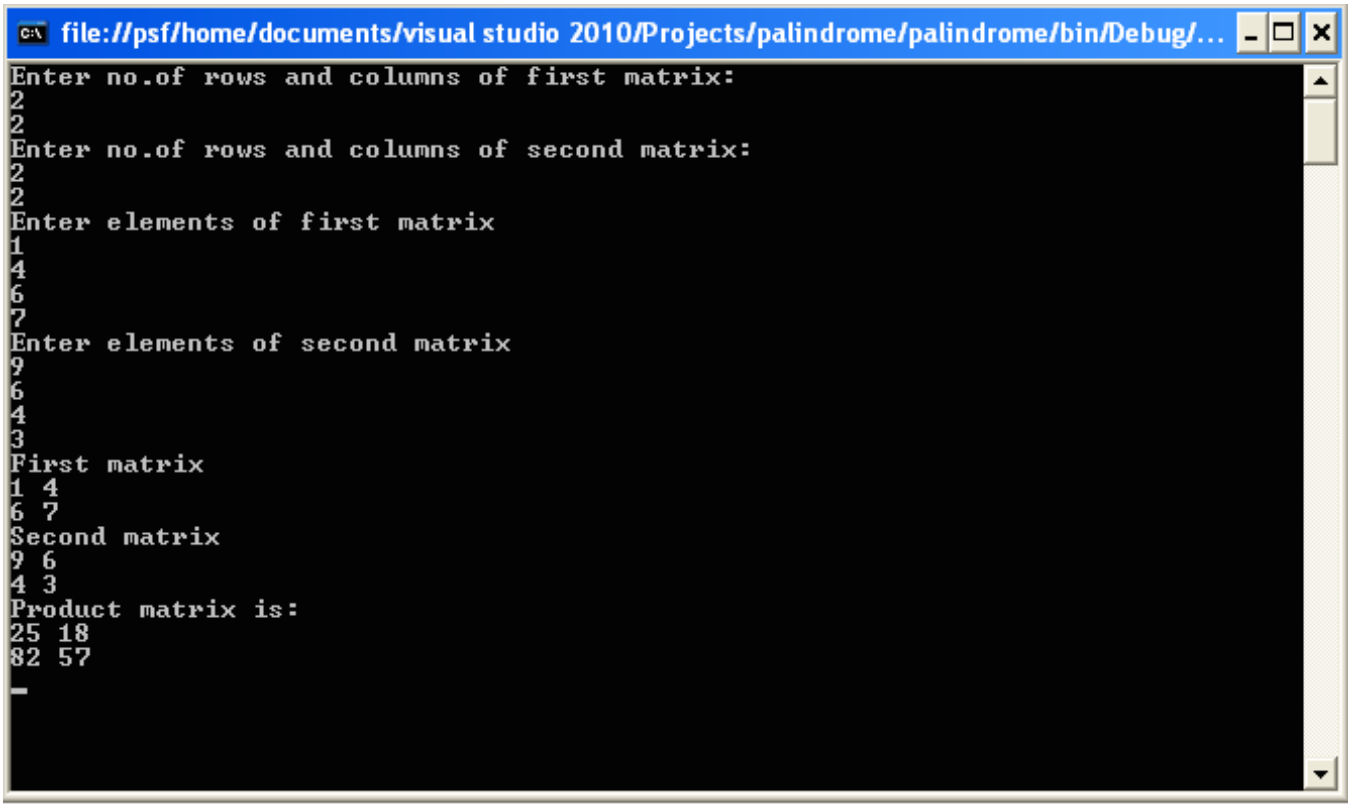
```
using System;
class MatMulti
{
        int r1,r2,c1,c2;
        double[,]a;      double[,]b;        double[,]c;
        publicMatMulti(int r1,int c1,int r2,int c2)
        {
                a=new double[(this.r1=r1),(this.c1=c1)];
                b=new double[(this.r2=r2),(this.c2=c2)];
                c=new double[r1,c2];
        }
```

```
public void Multiply()
{
        if(c1==r2)
        {
                Console.WriteLine("Enter elements of first matrix");
                for(inti=0;i<r1;i++)
                        for(int j=0;j<c1;j++)
                                a[i,j]=double.Parse(Console.ReadLine());

                Console.WriteLine("Enter elements of second matrix");
                for(inti=0;i<r2;i++)
                        for(int j=0;j<c2;j++)
                                 b[i,j]=double.Parse(Console.ReadLine());



                for(inti=0;i<r1;i++)
                {
                        for(int j=0;j<c2;j++)
                        {
                        c[i,j]=0;
                        for(int k=0;k<r2;k++)
                                c[i,j]+=a[i,k]*b[k,j];
                   }
                }
                Console.WriteLine("First matrix");
                for(inti=0;i<r1;i++)
                {
                for(int j=0;j<c1;j++)
                Console.Write(a[i,j]+" ");
                Console.WriteLine();
                }
```

```
                    Console.WriteLine("Second matrix");
                    for(inti=0;i<r2;i++)
                    {
                    for(int j=0;j<c2;j++)
                    Console.Write(b[i,j]+" ");
                    Console.WriteLine();
                    }

                    Console.WriteLine("Product matrix is:");
                    for(inti=0;i<r1;i++)
                    {
                    for(int j=0;j<c2;j++)
                    Console.Write(c[i,j]+" ");
                    Console.WriteLine();
                    }
            }
            else
                    Console.WriteLine("Multiplication is not possible:");
    }
}
class MultiImpl
{
    public static void Main()
    {
            int a,b,c,d;
            Console.WriteLine("Enter no.of rows and columns of first matrix:");
            a=int.Parse(Console.ReadLine());
            b=int.Parse(Console.ReadLine());
            Console.WriteLine("Enter no.of rows and columns of second matrix:");
            c=int.Parse(Console.ReadLine());
```

```
            d=int.Parse(Console.ReadLine());
            MatMulti m=new MatMulti(a,b,c,d);
            m.Multiply();
        }
}
```

OUTPUT:

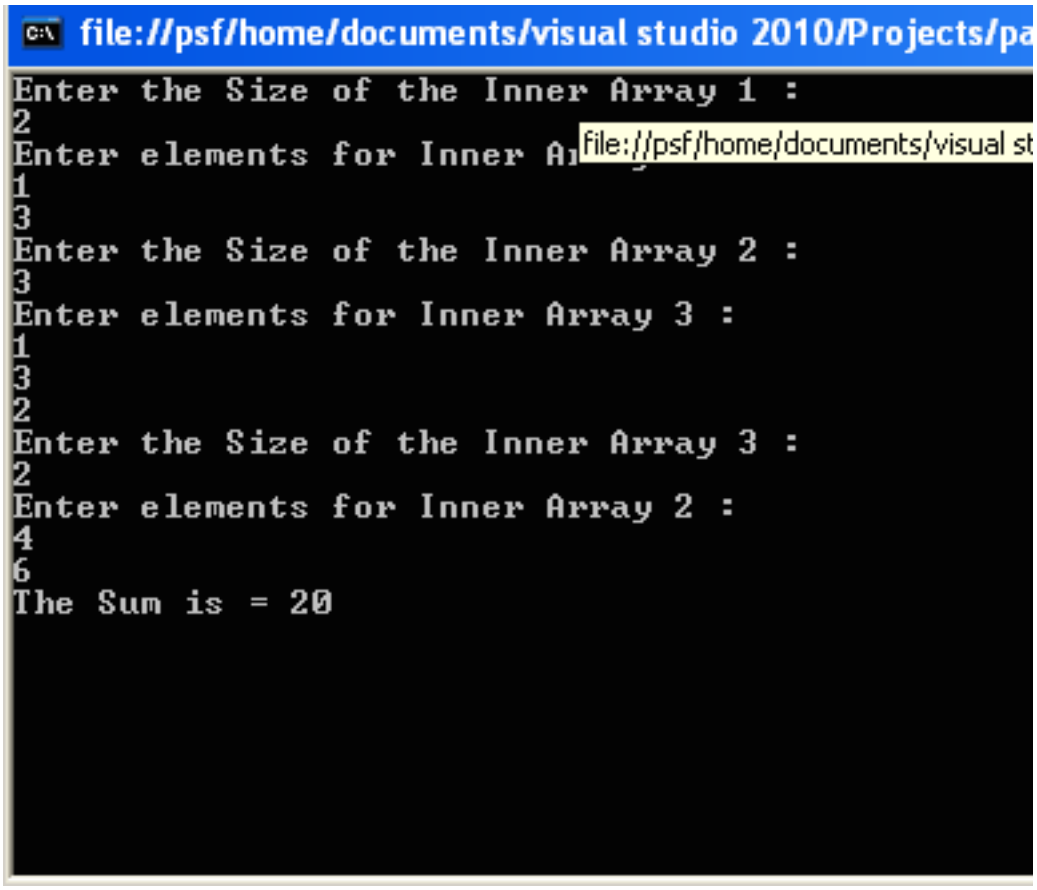


5 a) Write a C# program to find the sum of all the elements present in a jagged array of 3 inner arrays.

```
using System;
public class JaggedArrayDemo
{
        public static void Main()
        {
        int sum = 0;
        int[][] arr = new int[3][];
        arr[0] = new int[3];
        arr[1] = new int[5];
```

```
        arr[2] = new int[2];
        for (inti = 0; i<arr.Length; i++)
        {
                Console.WriteLine("Enter the Size of the Inner Array " + (i + 1) + " : ");
                arr[i] = new int[int.Parse(Console.ReadLine())];

                Console.WriteLine("Enter elements for Inner Array " + arr[i].Length + " : ");
                for (int j = 0; j <arr[i].Length; j++)
                {
                arr[i][j] = int.Parse(Console.ReadLine());
                sum += arr[i][j];
                }
      }
        Console.WriteLine("The Sum is = " + sum);
   }
}
```

OUTPUT:



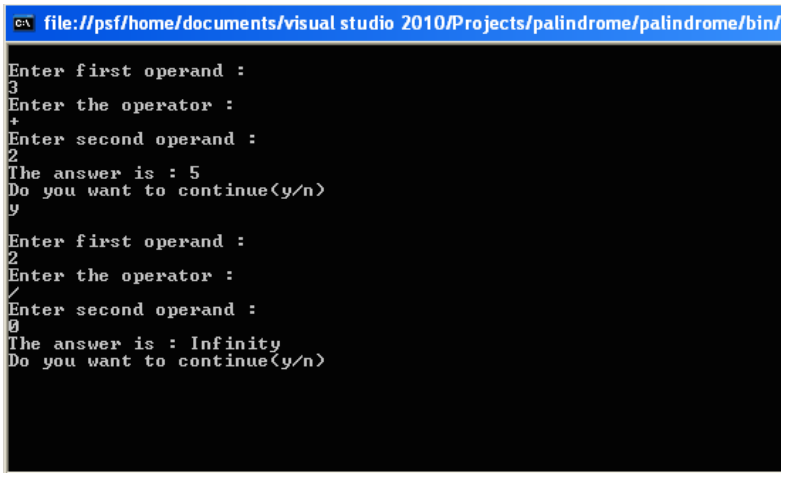
5 b) Design a simple calculator using *switch* Statement in C#.

```
using System;
namespace SimpleCalc
{
    class Calc
   {
          private float a, b, c;
          char op;
          public Calc(float a, float b, char op)
          {
          this.a = a;
          this.b = b;
          this.op = op;
   }
public void Calculator()
{
       try
          {
            if (op != '+' && op != '-' && op != '*' && op != '/')
            throw new opchck("Operator Invalid");
            else
             {
               switch (op)
                {
                   case '+': c = a + b; break;
                   case '-': c = a - b; break;
                  case '*': c = a * b; break;
                  case '/':
                              try
                              {
                              c = a / b;
```

```
                    }
                    catch (ArithmeticException e)
                    {
                            Console.WriteLine("Denomination zero");
                    } break;
                default: break;
              }
        Console.WriteLine("The answer is : " + c);
           }
        }
        catch (opchck o)
        {
             Console.WriteLine(o);
        }
    }
class opchck : Exception
{
        public opchck(string msg)
           : base(msg)
        {

        }
        public opchck()
        {

        }
    }
  }
```

```
public class Demo
{
        public static void Main(string[] args)
        {
        char ch;
        do
        {
        Console.WriteLine("\nEnter first operand : ");
        float a = float.Parse(Console.ReadLine());
        Console.WriteLine("Enter the operator : ");
        char op = char.Parse(Console.ReadLine());
        Console.WriteLine("Enter second operand : ");
        float b = float.Parse(Console.ReadLine());
        Calc c = new Calc(a, b, op);
        c.Calculator();
        Console.WriteLine("Do you want to continue(y/n)");
        ch = char.Parse(Console.ReadLine());
        } while (ch == 'Y' || ch == 'y');
        //Console.ReadLine();
      }
   }
}
```

OUTPUT:

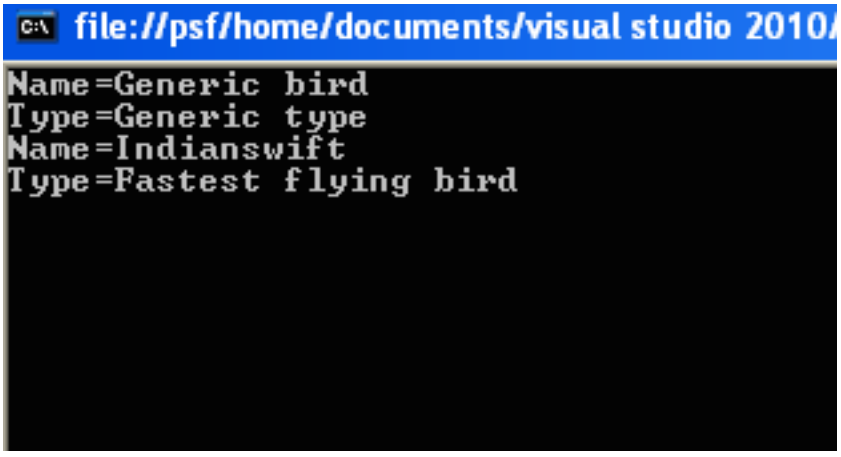


6 a) Demonstrate the use of *virtual* and *override* keyword in C# with a simple Program.

```
using System;
namespace Virtualnride
{
        class Bird
        {
                public string name;
                public string type;
                public virtual void setfun()
                {
                name = "Generic bird";
                type = "Generic type";
                }
                public virtual void display()
                {
                        Console.WriteLine("Name=" + name);
                        Console.WriteLine("Type=" + type);
                }
   }
class FlyingBird : Bird
 {
       public override void setfun()
       {
          //base.setfun();
          name = "Indianswift";
          type = "Fastest flying bird";
       }
```

```
        public override void display()
        {
            Console.WriteLine("Name=" + name);
            Console.WriteLine("Type=" + type);
        }
    }
class Program
    {
            static void Main(string[] args)
        {
            Bird o1 = new Bird();
            o1.setfun();
            o1.display();
             FlyingBird o2 = new FlyingBird();
            o2.setfun();
            o2.display();
            Console.ReadLine();
        }
    }
}
```

OUTPUT:



6 b) Implement Linked Lists in C# using the existing collections name space.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
namespace LinkedList
{
        public class MyLinked
        {
                private ArrayListarr;
                public MyLinked()
                {
                        Console.WriteLine("Linked list created");
                        arr = new ArrayList(0);
                }
                public void insert(int value, intpos)
                {
                        if (checkpos(pos))
                                arr.Insert(pos, value);
                }
                public void add(intval)
                {
                        arr.Add(val);
                }
                public Boolean checkpos(intpos)
                {
                if (arr.Count<pos || pos< 0)
                {
                Console.WriteLine("position should be greater than 0 and in between 1 & " + arr.Count);
                return false;
                }
```

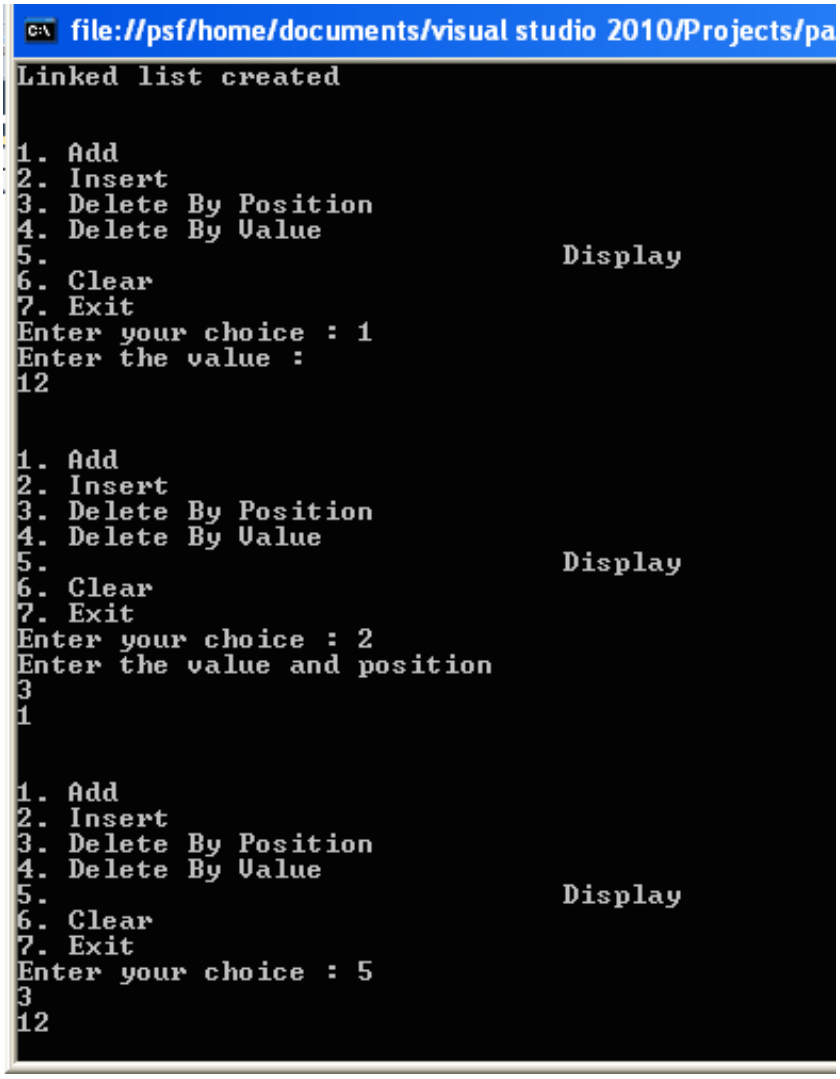
```
else
return true;
}
public Boolean checkLen()
{
        if (arr.Count == 0)
        {
                Console.WriteLine("list empty");
                return false;
        }
        return true;
}

public void remove(intpos)
{
if (pos>= 0 &&pos<arr.Count)
{
        arr.RemoveAt(pos);
}
else
{
        Console.WriteLine("Position is out of array size");
}
}
public void delete(intval)
{
        if (arr.IndexOf(val) >= 0 &&arr.IndexOf(val) <arr.Count)
                arr.Remove(val);
        else
                Console.WriteLine("The element does not present");
}
```

```
        public void show()
        {
            int[] a = new int[arr.Count];
            arr.CopyTo(a);
            if (checkLen())
            foreach (inti in arr)
                Console.WriteLine(i);
        }
        public void clearAll()
        {
            arr.Clear();
        }
    }
public class Demo
{
    public static void Main()
   {
    MyLinked  l = new MyLinked();
    int pos, val;
    while (true)
    {
     Console.Write("\n\n1. Add \n2. Insert\n3. Delete By Position \n4. Delete By Value\n5.
    Display \n6. Clear \n7. Exit \nEnter your choice : ");
       String ch = Console.ReadLine();
    switch (ch)
       {
    case "1":
            Console.WriteLine("Enter the value : ");
            val = int.Parse(Console.ReadLine());
            l.add(val);
            break;
```

```
case "2":
        Console.WriteLine("Enter the value and position ");
        val = int.Parse(Console.ReadLine());
        pos = int.Parse(Console.ReadLine());
        l.insert(val, pos - 1);
        break;
case "3":
        if (l.checkLen())
        {
        Console.WriteLine("Enter the position ");
        pos = int.Parse(Console.ReadLine());
        l.remove(pos - 1);
        }
        break;
case "4":
        if (l.checkLen())
        {
                Console.WriteLine("Enter the value");
                val = int.Parse(Console.ReadLine());
                l.delete(val);
        }
        break;
case "5":
        l.show();
        break;
case "6":
        l.clearAll();
        break;
case "7":
        Environment.Exit(0);
        break;
```

```
        default: break;
           }
        }
     }
  }
}
```

OUTPUT:



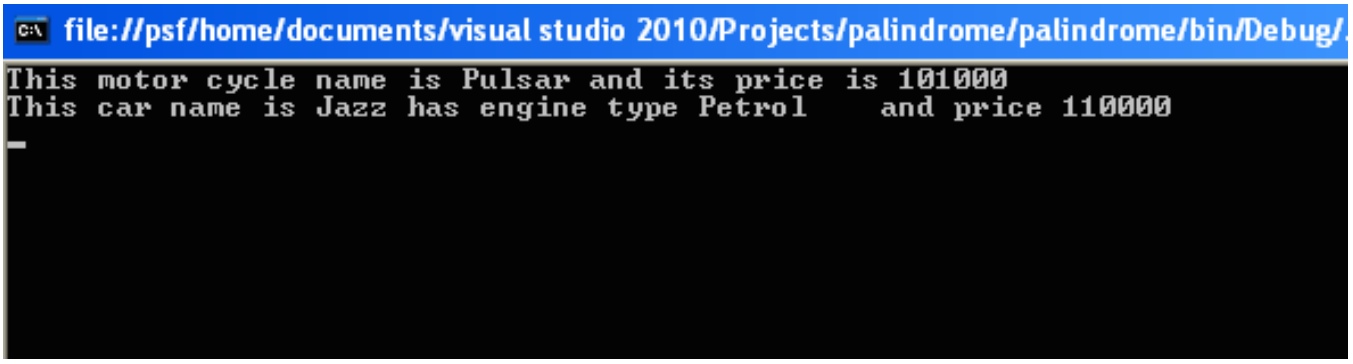
7 a) Write a C# program to demonstrate *abstract* class and *abstract* methods.

```
using System;
namespace Abstract
{
public abstract class Vehicle
{
public string Name;
public int Wheels;
public double Amount;
public abstract void Calculate();
}
public class Motorcycle : Vehicle
{
        public Motorcycle()
    {
        this.Wheels = 2;
    }
        public Motorcycle(string s)
    {
        this.Wheels = 2;
        this.Name = s;
    }
        public override void Calculate()
    {
        this.Amount = 100000 + (500 * this.Wheels);
        Console.WriteLine("This motor cycle name is " + this.Name + " and its price is " +
        this.Amount);
    }
  }
```

```
public class Car : Vehicle
{
        private string EngineType;
        public Car()
    {
        this.Wheels = 4;
    }
        public Car(string s, string t)
    {
        this.Wheels = 4;
        this.Name = s;
        this.EngineType = t;
    }
        public override void Calculate()
    {
        this.Amount = 100000 + (500 * this.Wheels) + 8000;
        Console.WriteLine("This car name is " + this.Name + " has engine type " + this.EngineType + "
        and price " + this.Amount);
    }
}
public class Program
{
        public static void Main(string[] args)
    {
        Vehicle v;
        Motorcycle m = new Motorcycle("Pulsar");
        Car c = new Car("Jazz", "Petrol");
        //m.Calculate();
        //c.Calculate();
        v = m;
        v.Calculate();
```

```
            v = c;
            v.Calculate();
            Console.ReadLine();
        }
    }
}
```

OUTPUT:

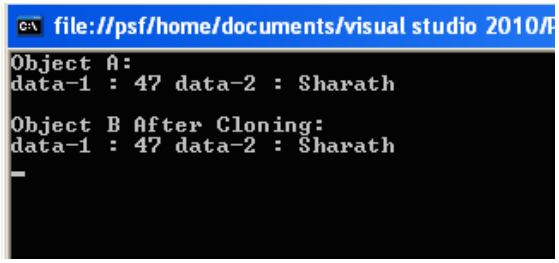


7b) Write a C# program to build a *class* which implements an existing *interface*.

```
using System;
class MyClone : ICloneable
{
    private int data1;
    private String data2;
   public MyClone()
   {
      data1 = 0;
      data2 = "";
   }
   public MyClone(int d1, String d2)
   {
      data1 = d1;
      data2 = d2;
   }
```

```
    public object Clone()
    {
        MyCloneob = new MyClone();
        ob.data1 = data1;
        ob.data2 = data2;
        return ob;
    }

   public override String ToString()
  {
        String str = "data-1 : " + data1 + " data-2 : " + data2;
        return str;
   }
}
class Demo
{
        public static void Main()
        {
                MyClone a = new MyClone(47, "Sharath");
                Console.WriteLine("Object A: \n" + a);
                MyClone b = (MyClone)a.Clone();
                Console.WriteLine("\nObject B After Cloning: \n" + b);
                Console.ReadLine();
    }
}
```

OUTPUT:

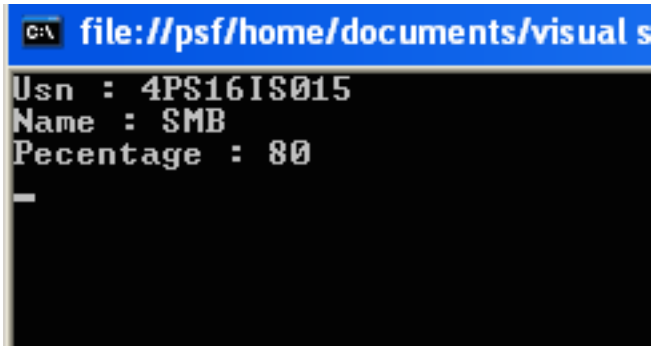


8 a) Write a C# program to illustrate the use of different properties.

```
using System;
class student
{
        private String usn;
        private String Name;
        private double perc;
        public String USN
        {
                get { return usn; }
        }
        public string name
        {
                set { Name = value; }
        }
        public double percentage
        {
                get { return perc; }
                set { perc = value; }
        }
        public student(String usn)
        {
                this.usn = usn;
        }
```

```
        public void show()
        {
                Console.WriteLine("Usn : " + usn + "\nName : " + Name + "\nPecentage : " + perc);
        }
}
public class Demo
{
        public static void Main()
        {
                student s = new student("4PS16IS015");
                 s.name = "SMB";
                s.percentage = 80.0;
                s.show();
    }
}
```

OUTPUT:



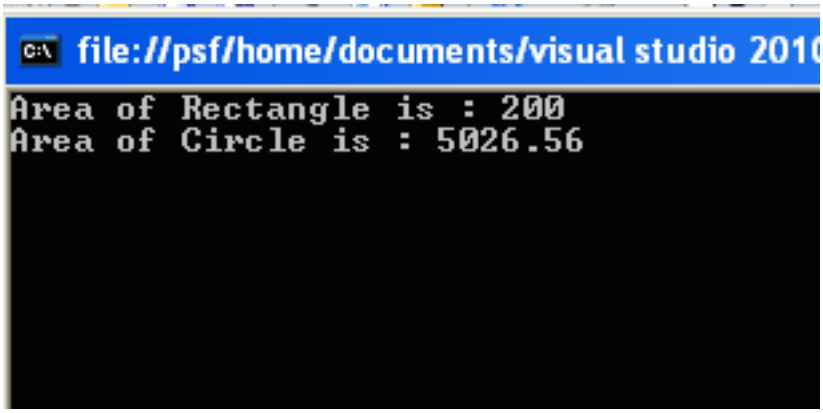
8 b) Demonstrate arrays of interface types with a C# program.

```
using System;
namespace InterfaceArray
{
     public interface IShape
    {
       void Calculate();
        void Display();
     }
    public class Rectangle : IShape
   {
     private double Area;
     private double Length;
     private double Breadth;
          public Rectangle()
          {
          this.Length = 0;
          this.Breadth = 0;
           }
          public Rectangle(double l, double b)
           {
          this.Length = l;
          this.Breadth = b;
           }
          public void Calculate()
          {
                    this.Area = this.Length * this.Breadth;
          }
```

```
        public void Display()
        {
                Console.WriteLine("Area of Rectangle is : " + this.Area);
        }
    }

public class Circle : IShape
{
private double Area;
private double Radius;
        public Circle()
        {
           this.Radius = 0;
        }
        public Circle(double s)
        {
           this.Radius = s;
        }
        public void Calculate()
        {
           this.Area = 3.1416 * this.Radius * this.Radius;
        }
         public void Display()
        {
           Console.WriteLine("Area of Circle is : " + this.Area);
        }
    }
```

```
public class Program
{
        public static void Main(string[] args)
       {
          IShape[] s = { new Rectangle(10, 20), new Circle(40) };
          for (int i = 0; i<s.Length; i++)
          {
                    s[i].Calculate();
                    s[i].Display();
          }
          Console.ReadLine();
       }
   }
}
```

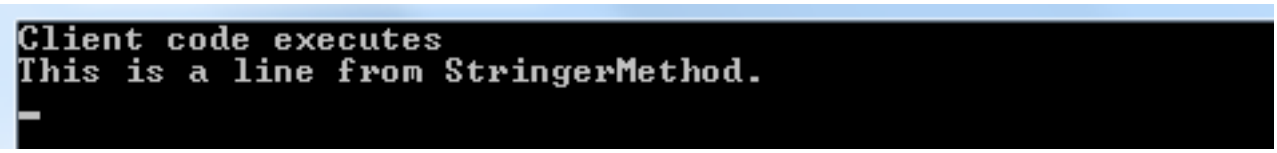
OUTPUT:



9 a) Write a C# program to illustrate the creation of a dll file and then using it in a program.

```
//Stringer.cs
// Assembly building example in the .NET Framework.
using System;
namespace myStringer
{
public class Stringer
   {
      public void StringerMethod()
      {
       System.Console.WriteLine("This is a line from Stringer Method.");
      }
   }
}
//MainClientApp.cs
using System;
using myStringer;
class MainClientApp
{
   // Static method Main is the entry point method.
   public static void Main()
   {
      Stringer myStringInstance = new Stringer();
       Console.WriteLine("Client code executes");
       myStringInstance.StringerMethod();
   }
}
```

OUTPUT:



9 b) Write a C# program to illustrate declaring, instantiating, and using a delegate.

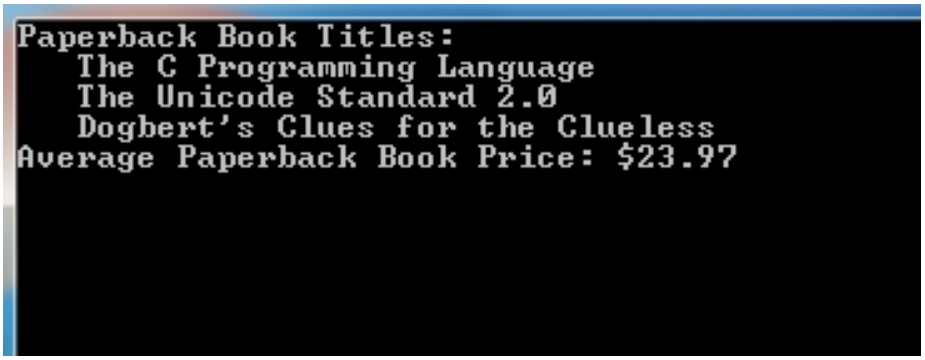
```
using System;
using System.Collections;
namespace Bookstore
{
    // Describes a book in the book list:
public struct Book
{
public string Title;        // Title of the book.
public string Author;       // Author of the book.
public decimal Price;       // Price of the book.
public bool Paperback;      // Is it paperback?
public Book(string title, string author, decimal price, bool    paperBack)
      {
         Title = title;
         Author = author;
         Price = price;
         Paperback = paperBack;
      }
   }
 // Declare a delegate type for processing a book:
public delegate void ProcessBookDelegate(Book book);
   // Maintains a book database.
public class BookDB
   {
      // List of all books in the database:
ArrayList list = new ArrayList();
      // Add a book to the database:
public void AddBook(string title, string author, decimal price, bool paperBack)
      {
         list.Add(new Book(title, author, price, paperBack));
      }
```

```
        // Call a passed-in delegate on each paperback book to process it:
public void ProcessPaperbackBooks(ProcessBookDelegate processBook)
{
            foreach (Book b in list)
            {
            if (b.Paperback)
                  // Calling the delegate:
            processBook(b);
            }
        }
    }
}

// Using the Bookstore classes:
namespace BookTestClient
{
using Bookstore;


 // Class to total and average prices of books:
class PriceTotaller
{
int countBooks = 0;
decimal priceBooks = 0.0m;
internal void AddBookToTotal(Book book)
{
            countBooks += 1;
            priceBooks += book.Price;
}
```

```
internal decimal AveragePrice()
{
        return priceBooks / countBooks;
}
}
   // Class to test the book database:
class TestBookDB
{
      // Print the title of the book.
       static void PrintTitle(Book b)
      {
         System.Console.WriteLine("   {0}", b.Title);
      }
      // Execution starts here.
       static void Main()
      {
         BookDB bookDB = new BookDB();
         // Initialize the database with some books:
         AddBooks(bookDB);
         // Print all the titles of paperbacks:
         System.Console.WriteLine("Paperback Book Titles:");
         // Create a new delegate object associated with the static
         // method Test.PrintTitle:
         bookDB.ProcessPaperbackBooks(PrintTitle);
         // Get the average price of a paperback by using
         // a PriceTotaller object:
         PriceTotaller totaller = new PriceTotaller();
         // Create a new delegate object associated with the nonstatic
         // method AddBookToTotal on the object totaller:
         bookDB.ProcessPaperbackBooks(totaller.AddBookToTotal);
```

```
            System.Console.WriteLine("Average Paperback Book Price: ${0:#.##}",
            totaller.AveragePrice());
            Console.Read();
        }
        // Initialize the book database with some test books:
        static void AddBooks(BookDB bookDB)
        {
          bookDB.AddBook("The C Programming Language", "Brian W. Kernighan and Dennis M.
            Ritchie", 19.95m, true);
            bookDB.AddBook("The Unicode Standard 2.0", "The Unicode Consortium", 39.95m, true);
            bookDB.AddBook("The MS-DOS Encyclopedia", "Ray Duncan", 129.95m, false);
            bookDB.AddBook("Dogbert's Clues for the Clueless", "Scott Adams", 12.00m, true);
        }
    }
}
```

OUTPUT:



Viva Questions:

1. Does C# support multiple-inheritance?
2. Where is a protected class-level variable available?
3. Are private class-level variables inherited?
4. Describe the accessibility modifier "protected internal".
5. Which class is at the top of .NET class hierarchy?
6. What does the term immutable mean?
7. Can you store multiple data types in System.Array?
8. What's the difference between the System.Array.CopyTo() and System.Array.Clone()?
9. How can you sort the elements of the array in descending order?
10. What's the .NET collection class that allows an element to be accessed using a unique key?
11. Will the finally block get executed if an exception has not occurred?
12. What's the C# syntax to catch any possible exception?
13. Can multiple catch blocks be executed for a single try statement?
14. What is the syntax to inherit from a class in C#?
15. Can you prevent your class from being inherited by another class?
16. Can you allow a class to be inherited, but prevent the method from being over-ridden?
17. What's an abstract class?
18. What is an interface class?
19. What is the difference between a Struct and a Class?
20. What's the implicit name of the parameter that gets passed into the set method/property of a class?
21. What does the keyword "virtual" declare for a method or property?
22. How is method overriding different from method overloading?

    Can you declare an override method to be static if the original method is not static?
23. What are the different ways a method can be overloaded?
24. What's the implicit name of the parameter that gets passed into the class' set method?