C# Programs

*Q.1 Structure of C#

Write the program of Hello World

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Hello World");
         Console.ReadKey();
      }
   }
}
```

Output: Hello World

*Q.2 Write the program in c# using operator to perform arithmetic operation.

```
namespace ConsoleApplication2 {
class Program {
static void Main(string[] args) {
Console.WriteLine("enter the 1st number"); int a = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter the 2nd number"); int b = Convert.ToInt32(Console.ReadLine()); int c
= a + b;
Console.WriteLine(“the final value is=”+c); Console.ReadKey();
} }
}
```

Output:
enter the 1st number 34
Enter the second number 6
The final value is=40

*Q.3 Write a Console application to demonstrate decision making:

1) Write the program of if-else statement

```
namespace ConsoleApplication2
{
   class Program
   {
      static void Main(string[] args)
      {
         int num = 11;
         if (num % 2 == 0)
         {
            Console.WriteLine("It is even number");
         }
         else
         {
            Console.WriteLine("It is odd number");

         }
         Console.ReadKey();
      }
   }
}
```

Output: It is odd number

2) Write the program of if else-if statement

```
namespace ConsoleApplication3
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Enter the number to check grade :");
         int num = Convert.ToInt32(Console.ReadLine());
         if (num < 0 || num > 100)
         {
            Console.WriteLine("Wrong Number");
         }
         else if (num > 0 && num < 50)
         {
            Console.WriteLine("Fail");
         }
         else if (num > 50 && num < 60)
         {
```

```
            Console.WriteLine("D grade");
        }
        else if (num > 60 && num < 70)
        {
            Console.WriteLine("C grade");
        }
        else if (num > 70 && num < 80)
        {
            Console.WriteLine("B grade");
        }
        else if (num > 80 && num < 90)
        {
            Console.WriteLine("A grade");
        }
        else if (num >= 90 && num <= 100)
        {
            Console.WriteLine("A+ grade");
        }
        Console.ReadKey();
    }
  }
}
```

Output:

Enter the number to check grade :

85

A grade

*Q.4 Write the program using Switch case statement

```
namespace ConsoleApplication4
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Enter the number:");
           int num=Convert.ToInt32(Console.ReadLine());
         switch (num)
         {
            case 10:Console.WriteLine("It is 10");break;
            case 20:Console.WriteLine("It is 20");break;
            case 30:Console.WriteLine("It is 30");break;
            default:Console.WriteLine("Not 10,20 or 30");
               break;
```

```
        }
        Console.ReadKey();
      }
   }
}
```

Output:

Enter the number:

40

Not 10,20 or 30

*Q.5 Looping

1) Write the program of Nested For Loop

```
namespace ConsoleApplication4
{
   class Program
   {
      static void Main(string[] args)
      {
         for (int i = 1; i <= 3; i++)
         {
            for (int j = 1; j <= 3; j++)
            {
               Console.WriteLine(i + " " + j);
            }

         }
            Console.ReadKey();
      }
   }
}
```

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

2) Write the program of Nested While Loop

```
namespace ConsoleApplication4
{
   class Program
   {
      static void Main(string[] args)
      {
         int i = 1;
         while (i <= 3)
         {
            int j = 1;
            while (j <= 3)
            {
               Console.WriteLine(i + " " + j);
               j++;
            }
            i++;

         }
         Console.ReadKey();
      }
   }
}
```

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

3) Write the program of Nested Do-While Loop

```
namespace ConsoleApplication5
{
   class Program
   {
      static void Main(string[] args)
      {
         int i = 1;
         do{
            int j=1;
            do
            {
               Console.WriteLine(i + " " + j);
               j++;
            }while (j <= 3);
            i++;
         }while(i<=3);
            Console.ReadKey();
      }
   }
}
```

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

*Q.6 Arrays

1) Single Dimensional Array

```
namespace ConsoleApplication4
{
   class Program
   {
      static void Main(string[] args)
      {
         int[] arr = new int[5];
         arr[0] = 10;
         arr[2] = 20;
         arr[4] = 30;
         for(int i=0;i<arr.Length;i++)
         {
            Console.WriteLine (arr[i]);

         }
         Console.ReadKey();
      }

   }
}
```

Output:

10

0

20

0

30

2) Multidimensional Array

```
namespace ConsoleApplication4
{
   class Program
   {
      static void Main(string[] args)
      {
         int[,] arr = new int[3,3];
         arr[0,1] = 10;
         arr[1,2] = 20;
         arr[2,0] = 30;
```

```
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)

                {
                    Console.WriteLine(arr[i,j]+" ");

                }
                Console.WriteLine();
            }
            Console.ReadKey();
        }

    }
}
```

Output:

0

10

0

0

0

20

30

0

0

3) Jagged Array

```
{
    public class JaggedArrayTest
    {
        public static void Main()
        {
            int[][] arr = new int[2][];// Declare the array

            arr[0] = new int[] { 11, 21, 56, 78 };// Initialize the array
            arr[1] = new int[] { 42, 61, 37, 41, 59, 63 };

            // Traverse array elements
```

```
            for (int i = 0; i < arr.Length; i++)
            {
                for (int j = 0; j < arr[i].Length; j++)
                {
                    System.Console.Write(arr[i][j] + " ");
                }
                System.Console.WriteLine();

            }
            Console.ReadKey();
        }
    }
}
```

Output:

11 21 56 78

42 61 37 41 59 63

*Q.7 Function

1) Function using no parameter and return type.

```
namespace ConsoleApp15
{
    internal class Program
    {
        public void Show()
        {
            Console.WriteLine("This is non parameterized function");
        }
        static void Main(string[] args)
        {
            Program p = new Program();
            p.Show();
            Console.ReadKey();
        }

    }
}
```

Output: This is non parameterized function

This is non parameterized function

2) Function using parameter and return type.

```
namespace ConsoleApp15
{
    internal class Program
    {
        public String Show(String message)
        {
            Console.WriteLine("Inside show function");
            return message;
        }
        static void Main(string[] args)
        {
```

```
            Program p = new Program();
            string message=p.Show("Good Morning");
            Console.WriteLine("Hello "+ message);
            Console.ReadKey();
        }

    }
}
```

Output:

```
Inside show function
Hello Good Morning
```

3)Write down a c# code for function using parameter and return type. Take a input from user and perform arithmetic operation.

```
namespace ConsoleApp20
{
    internal class Program
    {
        int add (int x,int y)
        {
            int z = x + y;
            return z;
        }
        int sub(int x,int y)
        {
            int z = x - y;
            return z;
        }
        float div(float x,float y)
        {
            float z = x / y;
            return z;
        }
        int mul(int x,int y)
        {
            int z = x * y;
            return z;
        }
        static void Main(String[] args)
        {
            Program p = new Program();
            Console.WriteLine("Enter first number");
            int a = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("Enter the second number");
            int b = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine(p.add(a, b));
            Console.WriteLine(p.sub(a, b));
            Console.WriteLine(p.div(a, b));
            Console.WriteLine(p.mul(a, b));
            Console.ReadKey();
        }

    }
}
```

Output:

```
Enter first number
```

```
18
Enter the second number
6
24
12
3
108
```

- Object Oriented Programming in C#

1) Ex.of object Class.

```
namespace ConsoleApp15
{
   internal class Program
   {
      int id;
      string name;
      static void Main(string[] args)
      {
         Program p1 = new Program();
         p1.id = 101;
         p1.name = "Abcxyz";
         Console.WriteLine(p1.id);
         Console.WriteLine(p1.name);
         Console.ReadKey();
      }

   }
}
```

Output:

101

Abcxyz

2) Store a display Employee Information.

```
namespace ConsoleApp16
{
   internal class Employee
   {
      public int id;
      public string name;
      public float salary;
      public void insert(int i, string n, float s)
      {
         id = i;
```

```
            name = n;
            salary = s;
        }
        public void display()

        {
            Console.WriteLine(id + " " + name + " " + salary);
        }
    }

    class TestEmployee
    {
        static void Main(String[] args)
        {
            Employee e1 = new Employee();
            Employee e2 = new Employee();
            e1.insert(101, "ABC", 890000f);
            e2.insert(102, "PQR", 490000f);
            e1.display();
            e2.display();
            Console.ReadKey();
        }

    }
}
```

Output:

```
101 ABC 890000
102 PQR 490000
```

*Q.8 C# Constructor

1) Default Constructor

```
namespace ConsoleApp16
{
    internal class Program
    {
        public Program()
        {
            Console.WriteLine("Default Constructor invoke");
        }
        static void Main(String[] args)
        {
            Program e1 = new Program();
            Program e2 = new Program();
            Console.ReadKey();
        }

    }
}
```

Output:

Default Constructor invoke

Default Constructor invoke

2) Parameterized Constructor

```
namespace ConsoleApp15
{
   internal class Program
   {
      public int id;
      public string name;
      public float salary;
      public Program(int i, string n, float s)
      {
         id = i;
         name = n;
         salary = s;
      }
      public void display()

      {
         Console.WriteLine(id + " " + name + " " + salary);
      }
   }

   class TestProgram
   {
      static void Main(String[] args)
      {
         Program e1 = new Program(101, "abc", 890000f);
         Program e2 = new Program(102, "xyz", 490000f);

         e1.display();
         e2.display();
         Console.ReadKey();
      }

   }
}
```

Output:

101 abc 890000

102 xyz 490000

- **Destructor**

```
namespace ConsoleApp15
{
   internal class Employee
   {
      public Employee()

      {
         Console.WriteLine("Constructor Invoked");
      }
      ~Employee()
```

```
        {
            Console.WriteLine("Destructor Invoked");
        }

    }
    class TestEmployee
    {
        public static void Main(String[] args)
        {
            Employee e1 = new Employee();
            Employee e2 = new Employee();
            Console.ReadKey();
        }
    }


}
```

Output:

Constructor Invoked

Constructor Invoked

*Q.9 Inheritance

1) Single inheritance

```
namespace ConsoleApp17
{
    public class Parent
    {
        public void DisplayParentAB()
        {
            Console.WriteLine("A and B are my parents");
        }
    }
        public class Son:Parent
        {
            public void DisplaySonC()
            {
                Console.WriteLine("I am the son C");
            }
        }
    internal class Program
    {
        static void Main(string[] args)
        {
            Son s=new Son();
            s.DisplaySonC();
            s.DisplayParentAB();
            Console.ReadKey();
        }
    }
}
```

Output:

I am the son C

A and B are my parents

2) Multi level inheritance

```
namespace ConsoleApp17
{
   public class Grandparent
   {
      public void GrandparentAB()
      {
         Console.WriteLine("Constructor called at run-time");
      }
      public void DispalyGrandparentAB()
      {
         Console.WriteLine("A and B are my Grandpatrents");
      }
   }
      public class Parent:Grandparent
      {
         public void DisplayParentCD()
         {
         Console.WriteLine("C and D are my parents");
         }
      }
   public class Child:Parent
   {
      public void DisplayChildZ()
      {
         Console.WriteLine("I am the child Z");
      }
   }
   internal class Program
   {
      static void Main(string[] args)
      {
         Child CD = new Child();
         CD.DisplayChildZ();
         CD.DisplayParentCD();
         CD.DispalyGrandparentAB();
         Console.ReadKey();
      }
   }
}
```

Output:

I am the child Z

C and D are my parents

A and B are my Grandparents

3) Hierarchical inheritance

```
namespace ConsoleApp17
{
   public class Parent
```

```
    {
        public void DisplayParentAB()
        {
            Console.WriteLine("A and B are my parents");
        }
    }
    public class ChildC:Parent
    {
        public void DisplayChildC()
        {
            Console.WriteLine("I am the child C");
        }
    }

        public class ChildD:Parent
        {
            public void DisplayChildD()
            {
            Console.WriteLine("I am the child D");
            }
        }

    internal class Program
    {
        static void Main(string[] args)
        {
            ChildC cc = new ChildC();
            ChildD cd = new ChildD();
            cc.DisplayChildC();
            cc.DisplayParentAB();

            cd.DisplayChildD();
            cd.DisplayParentAB();
            Console.ReadKey();
        }
    }
}
```

Output:

I am the child C

A and B are my parents

I am the child D

A and B are my parents

*Q.10 Polymorphism in C#

1) Method Overloading by changing the numbers of argument

```
namespace ConsoleApp17
{
    internal class Program
    {
        public static int add(int a,int b)
        {
            return a + b;
```

```
        }
        public static int add(int a, int b,int c)
        {
            return a + b + c;
        }
        public class Addition
        {
            static void Main(string[] args)
            {
                Console.WriteLine(Program.add(11, 22));
                Console.WriteLine(Program.add(11, 33, 44));
                Console.ReadKey();
            }
        }
    }
}
```

Output:

33

88

2) Method overloading by changing the datatype of arguments

```
namespace ConsoleApp17
{
    internal class Program
    {
        public static int add(int a,int b)
        {
            return a + b;
        }
        public static float add(float a, float b)
        {
            return a + b;
        }
         class TestMemberOverloading
        {
            static void Main(string[] args)
            {
                Console.WriteLine(Program.add(12,23));
                Console.WriteLine(Program.add(12.4f,12.3f));
                Console.ReadKey();
            }
        }
    }
}
```

Output:

35

24.7

3) Method Overloading

```
namespace ConsoleApp19
{
    public class Shape
```

```
    {
        public virtual void draw()
        {
            Console.WriteLine("Drawing...");
        }
    }
    public class Rectangle:Shape
    {
        public override void draw()
        {
            Console.WriteLine("Drawing rectangle...");
        }

    }
    public class  Circle:Shape
    {
        public override void draw()
        {
            Console.WriteLine("Draw Circle");
        }
    }
    public class TestPolymorphism
    {
        static void Main(string[] args)
        {
            Shape s;
            s = new Shape();
            s.draw();
            s = new Rectangle();
            s.draw();
            s = new Circle();
            s.draw();
            Console.ReadKey();
        }
    }
}
```

Output:

Drawing...

Drawing rectangle...

Draw Circle

*Q.11 Sealed Keywords

1) Sealed Class

```
namespace ConsoleApp20
{
    sealed public class Animal
    {
        public void eat()
        {
            Console.WriteLine("Eating....");
        }
    }
    public class Dog:Animal
    {
```

```
        public void sound()
        {
            Console.WriteLine("Dog Sound");
        }
    }
    public class TestSealed
    {
        static void Main(string[] args)
        {
            Dog d = new Dog();
            d.eat();
            d.sound();
            Console.ReadKey();
        }
    }
}
```

Output:

Error:Because of sealed class

2) Sealed Method

```
{
    public class Animal
    {
        public virtual void eat()
        {
            Console.WriteLine("eating...");
        }
        public virtual void run()
        {
            Console.WriteLine("running...");
        }

    }
    public class Dog : Animal
    {
        public override void eat()
        {
            Console.WriteLine("eating bread...");
        }
        public sealed override void run()
        {
            Console.WriteLine("running very fast...");
        }
    }
    public class BabyDog : Dog
    {
        public override void eat()
        {
            Console.WriteLine("eating biscuits...");
        }
        public override void run()
        {
            Console.WriteLine("running slowly...");
        }
    }
```

```
    public class TestSealed
    {
        public static void Main()
        {
            BabyDog d = new BabyDog();
            d.eat();
            d.run();
            Console.ReadKey();
        }
    }
}
```

Output:

Error:BabyDog.run(); cannot override inherited member

- Copy Constructor

```
namespace ConsoleApp20
{
    class Sample
    {
        public string param1, param2;
        public Sample(string x, string y)
        {
            param1 = x;
            param2 = y;
        }
        public Sample(Sample obj)      // Copy Constructor
        {
            param1 = obj.param1;
            param2 = obj.param2;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Sample obj = new Sample("Welcome", "c#dotnet");  // Create instance to class Sample
            Sample obj1 = new Sample(obj); // Here obj details will copied to obj1
            Console.WriteLine(obj1.param1 + " to " + obj1.param2);
            Console.ReadLine();
        }
    }
}
```

Output: Welcome to c#dotnet

- Abstract Class

```
namespace ConsoleApplication20

{
    public abstract class Bank
    {
        public abstract void withdraw();
    }
        public class YesBank : Bank
        {
```

```
        public override void withdraw()
        {
            Console.WriteLine("Withdrawing cash from YesBank");
        }

    }
    public class NoBank : Bank
    {
        public override void withdraw()
        {
            Console.WriteLine("Withdrawing cash from Nobank");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Bank b = new YesBank();
            b.withdraw();
            b = new NoBank();
            b.withdraw();
            Console.ReadKey();
        }
    }
}
```

Output:

Withdrawing cash from YesBank

Withdrawing cash from Nobank

*Exception Handling:

```
{
    internal class Program
    {
        static void Main(string[] args)
        { try
            {
                int a = 10;
                int b = 0;
                int x = a / b;
            }
            catch (Exception e) { Console.WriteLine(e); }
            Console.WriteLine("rest of the code");
            Console.ReadKey();
        }
    }
}
```

Output:

System.DivideByZeroException: Attempted to divide by zero.

at ConsoleApp22.Program.Main(String[] args) in C:\Users\bhave\source\repos\ConsoleApp22\ConsoleApp22\Program.cs:line 17

rest of the code

*Finally block

```
namespace ConsoleApp22
{
   internal class Program
   {
      static void Main(string[] args)
      { try
         {
            int a = 10;
            int b = 0;
            int x = a / b;
         }
         catch (Exception e)
         { Console.WriteLine(e); }
         finally
         {
            Console.WriteLine("Finally block is executed");
         }
         {
            Console.WriteLine("Rest of the Code");
         }
         Console.ReadKey();
      }
   }
}
```

Output:

System.DivideByZeroException: Attempted to divide by zero.

at ConsoleApp22.Program.Main(String[] args) in C:\Users\bhave\source\repos\ConsoleApp22\ConsoleApp22\Program.cs:line 17

Finally block is executed

Rest of the Code

*Multiple Catch Block

```
namespace ConsoleApp22
{
   internal class Program
   {
      static void Main(string[] args)
         {
         int a, b, c;

         Console.WriteLine("Enter any two numbers");
         try
         {
            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
```

```
            c = a / b;
            Console.WriteLine("c value=" + c);
        }
        catch (DivideByZeroException Ze)

        {
            Console.WriteLine("Second number should not be zero");
        }
        catch (FormatException Fe)
        {
            Console.WriteLine("Enter only integer number");
        }
        Console.ReadKey();
    }
  }
}
```

Output:

Enter any two numbers

18

virat

Enter only integer number