1 **c# read a two number and find minimum & maximum**
```
using System;
namespace FindMinMax{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter two numbers:");
            int a = Convert.ToInt32(Console.ReadLine());
            int b = Convert.ToInt32(Console.ReadLine());
            int min = Math.Min(a, b);
            int max = Math.Max(a, b);
            Console.WriteLine("Minimum: " + min);
            Console.WriteLine("Maximum: " + max);
            Console.ReadLine();
        }   } }
```
2 c# to print number odd &even Number from 1toN
```
using System;
namespace Consoleprogram {
    class Program    {
        static void Main(string[] args)  {
            Console.WriteLine("Enter a number:");
            int n = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Odd numbers from 1 to " + n + " are:");
            for (int i = 1; i <= n; i++)
            {
                if (i % 2 != 0)
                {
                    Console.Write(i + " ");
                }
            }
            Console.WriteLine();
            Console.WriteLine("Even numbers from 1 to " + n + " are:");
            for (int i = 1; i <= n; i++)
            {
                if (i % 2 == 0)
                {
                    Console.Write(i + " ");
                }
            }
            Console.WriteLine();
        }    }   }
```
3 **C# to find positive number from array of integer**
```
using System;
namespace PositiveNumbersExample
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] numbers = new int[] { -1, -2, 3, 4, -5, 6, 7, 8, 9, -10 };
            Console.WriteLine("Positive numbers in the array:");
            for (int i = 0; i < numbers.Length; i++)
            {
                if (numbers[i] > 0)
                {
                    Console.WriteLine(numbers[i]);
                }
            }   }   } }
```
4 **c# to find minimum & maximum in the array**
```
using System;
namespace ArrayMinMax
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] numbers = { 5, 2, 8, 9, 1, 4, 7 };
            int min = numbers[0];
            int max = numbers[0];
            for (int i = 1; i < numbers.Length; i++)
            {
                if (numbers[i] < min)
                {
                    min = numbers[i];
                }
                else if (numbers[i] > max)
                {
                    max = numbers[i];
                }
            }
            Console.WriteLine("The minimum value in the array is: " + min);
            Console.WriteLine("The maximum value in the array is: " + max);
            Console.ReadLine();
        }   } }
```
5 C# program to implement indexer for an integer array
```
using System;
class intValues
{
    private int[] intArray = { 90,89,88,87,86,85,84,83,82,81 };
    public int Size
    {
        get
        {
            return intArray.Length;
        }
    }
    public int this[int index]
    {
        get
        {
            return intArray[index];
        }
        set
        {
            intArray[index] = value;
        }   }  }
class Demo
{
    static void Main()
    {
        intValues vals = new intValues();
        int loop = 0;
        vals[2] = 47;
        vals[4] = 67;
        vals[6] = 74;
        for (loop = 0; loop < vals.Size; loop++)
        {
            Console.Write(vals[loop]+" ");
        }
        Console.WriteLine();
    }
}
```

6 **c# to search an element in an array**

```
using System;
using System.Collections.Generic;
public class GFG {
        public static void Main()
        {
                try {
                        String[] myArr = {"Sun", "Mon", "Tue", "Thu"};
                        Console.WriteLine("Initial Array:");
                        PrintIndexAndValues(myArr);
                        string value = Array.Find(myArr,
                        element => element.StartsWith("S",
                        StringComparison.Ordinal));
                        Console.Write("Element: ");
                        Console.Write("{0}", value);
                }
                catch (ArgumentNullException e) {
                        Console.Write("Exception Thrown: ");
                        Console.Write("{0}", e.GetType(), e.Message);
                }
        }
```

```
            public static void PrintIndexAndValues(String[] myArr)
            {
                        for (int i = 0; i < myArr.Length; i++) {
                                    Console.WriteLine("{0}", myArr[i]);
                        }
                        Console.WriteLine();
            } }
```

7 **write a c# program for constructor and its types**

```
using System;
class Car {
    public string Model;
    public int Year;
    public string Color;
    // Default constructor
    public Car() {
        Model = "Unknown";
        Year = 0;
        Color = "Unknown";
    }
    // Parameterized constructor
    public Car(string model, int year, string color) {
        Model = model;
        Year = year;
        Color = color;
    }
    // Copy constructor
    public Car(Car otherCar) {
        Model = otherCar.Model;
        Year = otherCar.Year;
        Color = otherCar.Color;
    }
    // Static constructor
    static Car() {
        Console.WriteLine("Static constructor called.");
    }
}
class Program {
    static void Main() {
        // Default constructor
        Car car1 = new Car();
        Console.WriteLine($"Model: {car1.Model}, Year: {car1.Year},
 Color: {car1.Color}");
        // Parameterized constructor
        Car car2 = new Car("Toyota", 2022, "Red");
        Console.WriteLine($"Model: {car2.Model}, Year: {car2.Year},
Color: {car2.Color}");
        // Copy constructor
        Car car3 = new Car(car2);
        Console.WriteLine($"Model: {car3.Model}, Year: {car3.Year}, 
Color: {car3.Color}");
    }
}
```

8 write a c# program for abstract class and methods

```
using System;
abstract class Shape
{
    public abstract double GetArea();
    public abstract double GetPerimeter();
}
class Rectangle : Shape
{
    double length;
    double width;
    public Rectangle(double l, double w)
    {
        length = l;
        width = w;
    }
    public override double GetArea()
    {
        return length * width;
    }
    public override double GetPerimeter()
    {
        return 2 * (length + width);
    }
}
class Program
{
    static void Main(string[] args)
    {
        Rectangle r = new Rectangle(5, 10);
        Console.WriteLine("Rectangle Area: " + r.GetArea());
        Console.WriteLine("Rectangle Perimeter: " + r.GetPerimeter());
        Console.ReadLine();
    }
}
```

9 write a c# program for sealed class and methods

```
using System;
sealed class SealedClass
{
    public void Method1()
    {
        Console.WriteLine("Method1 from SealedClass");
    }
    public void Method2()
    {
        Console.WriteLine("Method2 from SealedClass");
    }
}
class Program
{
    static void Main(string[] args)
    {
        SealedClass obj = new SealedClass();
        obj.Method1();
        obj.Method2();
    }
}
```

10 **write C# program for Binary Operator overloading**

```
using System;
class Vector2D
{
    public double X { get; set; }
    public double Y { get; set; }
    public Vector2D(double x, double y)
    {
        X = x;
        Y = y;
    }
    public static Vector2D operator +(Vector2D v1, Vector2D v2)
    {
        return new Vector2D(v1.X + v2.X, v1.Y + v2.Y);
    }
    public static Vector2D operator -(Vector2D v1, Vector2D v2)
    {
        return new Vector2D(v1.X - v2.X, v1.Y - v2.Y);
    }
    public override string ToString()
    {
        return $"({X}, {Y})";
    }
}
class Program
{
    static void Main(string[] args)
    {
        Vector2D v1 = new Vector2D(1.0, 2.0);
        Vector2D v2 = new Vector2D(3.0, 4.0);
        Vector2D sum = v1 + v2;
        Vector2D difference = v1 - v2;
        Console.WriteLine($"v1 = {v1}");
        Console.WriteLine($"v2 = {v2}");
        Console.WriteLine($"v1 + v2 = {sum}");
        Console.WriteLine($"v1 - v2 = {difference}");
    }
}
```
11write C# program for Delegates
```
using System;
delegate int MyDelegate(int x, int y);
class Program
{
    static int Add(int x, int y)
    {
        return x + y;
    }
    static int Subtract(int x, int y)
    {
        return x - y;
    }
    static void Main(string[] args)
    {
        MyDelegate addDelegate = new MyDelegate(Add);
        MyDelegate subtractDelegate = new MyDelegate(Subtract);
        int result1 = addDelegate(5, 3);
        int result2 = subtractDelegate(5, 3);
        Console.WriteLine("Result of Add: " + result1);
        Console.WriteLine("Result of Subtract: " + result2);
    }
}
```
12 write C# program for Multicast delegates
```
using System;
delegate void MyDelegate(string message);
class Program
{
    static void Main(string[] args)
    {
        MyDelegate del1 = new MyDelegate(Method1);
        MyDelegate del2 = new MyDelegate(Method2);
        MyDelegate multicastDel = del1 + del2;
        multicastDel("Hello, world!");
        multicastDel -= del1;
        multicastDel("Hello again!");
    }
    static void Method1(string message)
    {
        Console.WriteLine("Method1 says: " + message);
    }
    static void Method2(string message)
    {
        Console.WriteLine("Method2 says: " + message);
    }
}
```

13write C# program Exception handling

```
using System;
class Program
{
    static void Main(string[] args)
    {
        try
        {
            // Attempt to divide by zero
            int x = 10;
            int y = 0;
            int result = x / y;
            Console.WriteLine("The result is: " + result);
        }
        catch (DivideByZeroException ex)
        {
            Console.WriteLine("Error: " + ex.Message);
        }
        finally
        {
            Console.WriteLine("This code is always executed, regardless of whether an exception was
thrown or not.");
        }

        Console.WriteLine("Program completed.");
    }
}
```

14 **c# to simple ATM machine**

```
using System;
namespace ATMMachine
{
    class Program
    {
        static void Main(string[] args)
        {
            int balance = 1000; //initial balance
            int withdrawal;
            int deposit;
            int option;
            Console.WriteLine("Welcome to the ATM machine!");
            do
            {
                Console.WriteLine("Please select an option:");
                Console.WriteLine("1. Check Balance");
                Console.WriteLine("2. Withdraw");
                Console.WriteLine("3. Deposit");
                Console.WriteLine("4. Exit");
            option = Convert.ToInt32(Console.ReadLine());
            switch (option)
            {
                case 1:
                    Console.WriteLine("Your current balance is $" + balance);
                    break;
                case 2:
                    Console.WriteLine("Enter the amount you want to withdraw: ");
                    withdrawal = Convert.ToInt32(Console.ReadLine());
                    if (withdrawal > balance)
                    {
                        Console.WriteLine("Insufficient balance");
                    }
                    else
                    {
                        balance = balance - withdrawal;
                        Console.WriteLine("You have withdrawn $" + withdrawal);
                        Console.WriteLine("Your new balance is $" + balance);
                    }
                    break;
                case 3:
                    Console.WriteLine("Enter the amount you want to deposit: ");
                    deposit = Convert.ToInt32(Console.ReadLine());
                    balance = balance + deposit;
                    Console.WriteLine("You have deposited $" + deposit);
                    Console.WriteLine("Your new balance is $" + balance);
                    break;
                case 4:
                    Console.WriteLine("Thank you for using the ATM machine!");
                    break;
                default:
                    Console.WriteLine("Invalid option. Please try again.");
                    break;
            }
        } while (option != 4);
}   }}
```