```
// 1. Program to Check whether a number is Palindrome or not.

using System;
using System.Linq;

class NumPalindrome
{
    static void Main(string[] args)
    {
        Console.Write("Enter a number: ");
        var num = Console.ReadLine();

        var rev = new string(num.Reverse().ToArray());

        if (num == rev)
            Console.WriteLine("{0} is a Palindrome", num);
        else
            Console.WriteLine("{0} is Not a Palindrome", num);


        Console.ReadKey();
    }
}
```

```
// 2. Program to demonstrate Command line arguments Processing.

using System;

class CmdLineArgs
{
    static void Main(string[] args)
    {
        int argc = args.Length;

        if (argc == 0)
        {
            Console.WriteLine("No Arguments!");
        }
        else
        {
            Console.WriteLine("Number of Arguments: " + argc);

            for (int i = 0; i < argc; i++)
            {
                Console.WriteLine("Argument {0} : {1}", i + 1, args[i]);
            }
        }

        Console.ReadKey();
    }
}
```

```
// 3. Program to find the roots of Quadratic Equation.

using System;

class QuadraticEqn
{
    static void Main(string[] args)
    {
        double a, b, c;

        Console.Write("Enter a: ");
        a = double.Parse(Console.ReadLine());
        Console.Write("Enter b: ");
        b = double.Parse(Console.ReadLine());
        Console.Write("Enter c: ");
        c = double.Parse(Console.ReadLine());

        double discriminant = (b * b) - (4 * a * c);

        if (a == 0)
        {
            Console.WriteLine("This is Linear Equation.");
        }
        else if (discriminant < 0)
        {
            Console.WriteLine("Roots are Imaginary");

            double root1 = (-b) / (2 * a);
            double root2 = (Math.Sqrt(Math.Abs(discriminant))) / (2 * a);

            Console.WriteLine("Root 1: " + root1);
            Console.WriteLine("Root 2: " + root2);
        }

        else if (discriminant == 0)
        {
            Console.WriteLine("Roots are Real and Equal");

            double root = (-b) / (2 * a);
            Console.WriteLine("Root: " + root);
        }

        else if (discriminant > 0)
        {
            Console.WriteLine("Roots are Real and Unequal");

            double root1 = (-b + Math.Sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.Sqrt(discriminant)) / (2 * a);

            Console.WriteLine("Root 1: " + root1);
            Console.WriteLine("Root 2: " + root2);
        }

        Console.ReadKey();
    }
}
```

```
// 4. Program to demonstrate Boxing and unBoxing.

using System;

class BoxUnbox
{
    static void Main(string[] args)
    {
        Console.WriteLine("Boxing...");

        int intVar = 100;
        object objVar = intVar;

        Console.WriteLine("Value of 'intVar' = " + intVar);
        Console.WriteLine("Value of 'objVar' = " + objVar);

        Console.WriteLine();

        Console.WriteLine("UnBoxing...");

        int anotherIntVar = (int)objVar;
        Console.WriteLine("Value of 'anotherIntVar' = " + anotherIntVar);


        Console.ReadKey();
    }
}
```

```
// 5. Program to implement Stack operations.

using System;

public class MyStack<T>
{
    int SIZE;
    int top;
    T[] st;

    public MyStack(int n = 10)
    {
        SIZE = n > 0 ? n : 10;
        top = -1;

        st = new T[SIZE];
    }

    public void push(T value)
    {
        if (top == SIZE - 1)
        {
            Console.WriteLine("Stack Overflow.");
            return;
        }

        st[++top] = value;
        Console.WriteLine(value + " is pushed.");
    }

    public T pop()
    {
        if (top == -1)
        {
            Console.WriteLine("Stack Underflow");
            return default(T);
        }

        T value = st[top--];
        Console.WriteLine(value + " is popped.");

        return value;
    }

    public void display()
    {
        if (top == -1)
        {
            Console.WriteLine("Stack Underflow.");
            return;
        }
```

```
        Console.WriteLine("Stack Contains:");
        for (int i = top; i >= 0; i--)
        {
            Console.WriteLine("[{0}]: {1}", i, st[i].ToString());
        }
    }
}

class StackExample
{
    static void Main(string[] args)
    {
        var intStack = new MyStack<int>(5);
        var stringStack = new MyStack<string>(10);

        intStack.push(10);
        intStack.push(20);
        intStack.push(30);
        intStack.push(40);
        intStack.display();
        intStack.pop();
        intStack.pop();
        intStack.display();

        stringStack.push("abc");
        stringStack.push("def");
        stringStack.push("ijk");
        stringStack.push("xyz");
        stringStack.display();
        stringStack.pop();
        stringStack.pop();
        stringStack.display();

        Console.ReadKey();
    }
}
```

```
// 6. Write a program to demonstrate Operator overloading.

using System;

class Complex
{
    double x;
    double y;

    public void read()
    {
        Console.Write("Enter real part : ");
        x = double.Parse(Console.ReadLine());
        Console.Write("Enter imaginary part : ");
        y = double.Parse(Console.ReadLine());
    }

    public static Complex operator +(Complex c1, Complex c2)
    {
        Complex c = new Complex();
        c.x = c1.x + c2.x;
        c.y = c1.y + c2.y;
        return c;
    }

    public static Complex operator -(Complex c1, Complex c2)
    {
        Complex c = new Complex();
        c.x = c1.x - c2.x;
        c.y = c1.y - c2.y;
        return c;
    }

    public void Display()
    {
        Console.WriteLine("{0} + i{1}", x, y);
    }
}

class ComplexTest
{

    public static void Main()
    {
        Complex a = new Complex();
        Console.WriteLine("a (Complex number1) : ");
        a.read();

        Complex b = new Complex();
        Console.WriteLine("\n\n b (Complex number2) : ");
        b.read();

        Complex c = a + b;
        Complex d = b - a;
```

```
            Console.Write("a = ");
            a.Display();
            Console.Write("b = ");
            b.Display();
            Console.Write("c = a + b : ");
            c.Display();
            Console.Write("d = b - a : ");
            d.Display();

            Console.ReadKey();
        }
    }
```

```
// 7. Program to find the second largest element in an array.

using System;
using System.Linq;

class SecondLargestElement
{
    static void Main(string[] args)
    {
        Console.Write("Enter the size of the array: ");
        int n = int.Parse(Console.ReadLine());

        int[] arr = new int[n];

        Console.WriteLine("Enter array values:");
        for (int i = 0; i < n; i++)
        {
            arr[i] = int.Parse(Console.ReadLine());
        }

        Console.WriteLine("The Second Largest element is: " + secondLargest(arr));

        Console.ReadKey();
    }

    static int secondLargest(int[] array)
    {
        return array.OrderByDescending(n => n).Distinct().Skip(1).First();
    }
}
```

```
// 8. Program to multiply to matrices using Rectangular arrays.

using System;

class MatrixMultiplication
{
    static void Main(string[] args)
    {
        int m, n;
        int p, q;
        int i, j, k;

        // Read First Matrix
        Console.WriteLine("Enter the number of Rows and Columns of first matrix");
        m = int.Parse(Console.ReadLine());
        n = int.Parse(Console.ReadLine());

        var first = new int[m, n];

        Console.WriteLine("Enter the elements of first matrix");
        for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            first[i, j] = int.Parse(Console.ReadLine());

        // Read Second Matrix
        Console.WriteLine("Enter the number of rows and columns of second matrix");
        p = int.Parse(Console.ReadLine());
        q = int.Parse(Console.ReadLine());

        if (n != p)
        {
            Console.WriteLine("Matrices can't be multiplied.");
            Console.ReadKey();
        }

        var second = new int[p, q];
        var result = new int[m, q];

        Console.WriteLine("Enter the elements of second matrix");
        for (i = 0; i < p; i++)
            for (j = 0; j < q; j++)
                second[i, j] = int.Parse(Console.ReadLine());

        // Multiply
        for (i = 0; i < m; i++)
        {
            for (j = 0; j < q; j++)
            {
                result[i, j] = 0;
                for (k = 0; k < p; k++)
                {
                    result[i, j] += first[i, k] * second[k, j];
                }
            }
        }
```

```
        // Display Result
        Console.WriteLine("Product of entered matrices:");
        for (i = 0; i < m; i++)
        {
            for (j = 0; j < q; j++)
            {
                Console.Write(" {0} ", result[i, j]);
            }
            Console.WriteLine();
        }


        Console.ReadKey();
    }
}
```

```
// 9. Find the sum of all the elements present in a jagged array of 3 inner arrays.

using System;

class Program
{
    static void Main(string[] args)
    {
        int n = 3; //Three Inner Arrays

        int[][] jaggedArray = new int[n][];

        for (int i = 0; i < n; i++)
        {
            Console.WriteLine("\n Enter the size of inner array {0}: ", i + 1);
            jaggedArray[i] = new int[int.Parse(Console.ReadLine())];

            Console.WriteLine("Enter the elements: ", i + 1);
            for (int j = 0; j < jaggedArray[i].Length; j++)
                jaggedArray[i][j] = int.Parse(Console.ReadLine());
        }

        // Calculate Sum
        int sum = 0;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < jaggedArray[i].Length; j++)
            {
                sum += jaggedArray[i][j];
            }
        }

        Console.WriteLine("\n\n Sum of all the three inner arrays is = {0}", sum);

        Console.ReadKey();
    }
}
```

```
// 10. Write a program to reverse a given string.

using System;
using System.Linq;

class ReverseString
{
    static void Main(string[] args)
    {
        Console.Write("Enter the string: ");
        string input = Console.ReadLine();

        string reversed = new string(input.Reverse().ToArray());

        Console.WriteLine("Reversed String is: " + reversed);

        Console.ReadKey();
    }
}
```

```
// 11. Using Try, Catch and Finally blocks Program to demonstrate error handling.

using System;

class ErrorHandling
{
    static void Main(string[] args)
    {
        int a = 10, b = 0;

        try
        {
            Console.Write("Enter a number: ");
            int num = int.Parse(Console.ReadLine());

            int c = a / b;
        }
        catch (DivideByZeroException e)
        {
            Console.WriteLine("Divide By Zero !!!!! : " + e.Message);
        }
        catch (Exception ex) // Catches All the Exceptions
        {
            Console.WriteLine(ex.Message);
        }
        finally
        {
            Console.WriteLine("Finally Block...");
        }

        Console.ReadKey();
    }
}
```