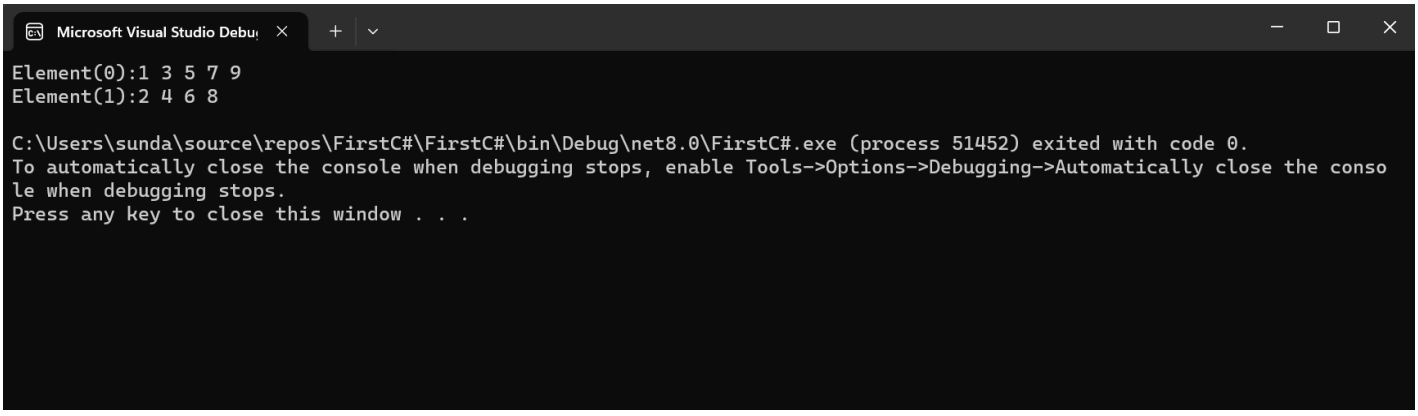
# C# PROGRAMMING

1) Design, Develop and Implement Applications in C# based on the concepts learned.Create a console application to implement jagged array.

Code:

```
using System;

namespace JaggedArray;

class Program
{
    static void Main()
    {
        // Declare and initialize a jagged array with two rows.
        // Note that the second dimension is not specified initially.
        int[][] myArray = new int[2][];

        // Initialize the first row with an array of 5 integers.
        myArray[0] = new int[5] { 1, 3, 5, 7, 9 };

        // Initialize the second row with an array of 4 integers.
        myArray[1] = new int[4] { 2, 4, 6, 8 };

        // Loop through each row of the jagged array.
        for (int row = 0; row < myArray.Length; row++)
        {
            // Print the row number.
            Console.Write("Element({0}):", row);

            // Loop through each element in the current row.
            for (int col = 0; col < myArray[row].Length; col++)
            {
                // Print the current element.
                Console.Write("{0}{1}", myArray[row][col],
                    col == myArray[row].Length - 1 ? "" : " ");
            }

            // Move to the next line after printing all elements of the current row.
            Console.WriteLine();
        }
    }
}
```

Output:



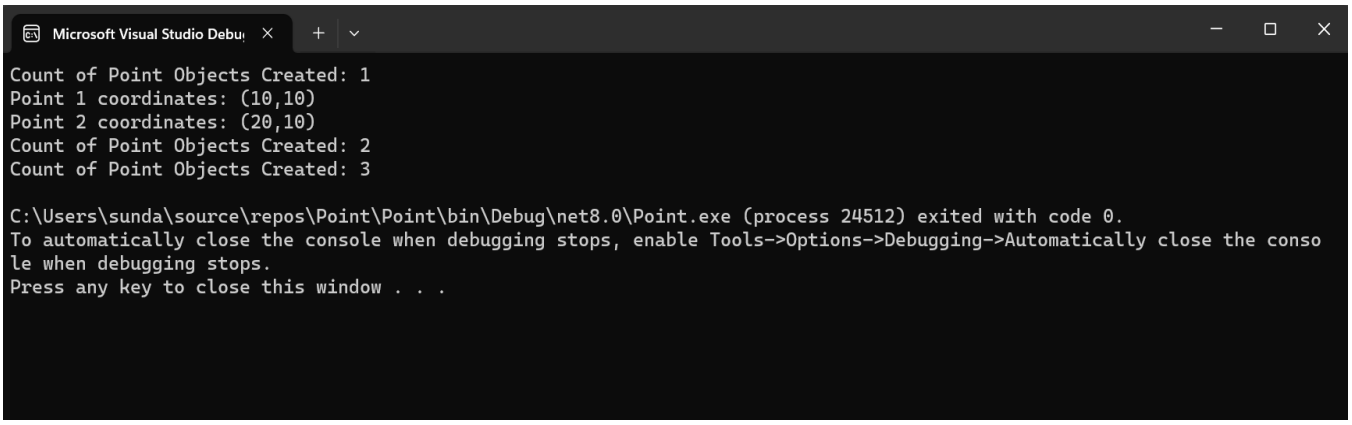
2) Create a Point class with x,y coordinates as its member variables , parameter less constructor, parameterized constructor, static field, static constructor to initialized the static field, static method to get the number of Point objects created and accessor methods to retrieve the member variables. Test the class by creating atleast 3 objects of type Point class and invoking the methods defined in the class.

Code:

```
// Namespace declaration to organize the code.
namespace Point
{

    // Class to represent points in 2D space.
    class Point
    {
        // Data members to store the coordinates of a point.
        private int x;
        private int y;

        // Static variable to keep track of the number of Point objects created.
        private static int countPoint;

        // Static constructor initializes the static variable countPoint to 0.
        static Point()
        {
            countPoint = 0;
        }

        // No-argument constructor initializes the point to default coordinates and increments
count.
        public Point()
        {
            this.x = 10;
            this.y = 10;
            countPoint++;
        }

        // Two-argument constructor allows initializing the point with specific coordinates
and increments count.
        public Point(int x, int y)
        {
            this.x = x;
            this.y = y;
            countPoint++;
        }

        // Getter methods to retrieve the x and y coordinates.
        public int GetX()
        {
            return x;
        }

        public int GetY()
        {
            return y;
        }

        // Static method returns the count of Point objects created.
        public static int GetCount()
        {
            return countPoint;
        }

        // Main method where the program execution begins.
        static void Main()
        {
            // Create a Point object using the parameterless constructor.
            Point point1 = new Point();
```

```
            // Display the count of Point objects created.
            Console.WriteLine("Count of Point Objects Created: {0}", GetCount());

            // Create another Point object using the two-argument constructor.
            Point point2 = new Point(20, 10);

            // Display the coordinates of point1 and point2.
            Console.WriteLine("Point 1 coordinates: ({0},{1})", point1.GetX(), point1.GetY());
            Console.WriteLine("Point 2 coordinates: ({0},{1})", point2.GetX(), point2.GetY());

            // Display the count of Point objects created.
            Console.WriteLine("Count of Point Objects Created: {0}", GetCount());

            // Create a new Point object with different coordinates.
            point2 = new Point(1, 1);

            // Display the count of Point objects created.
            Console.WriteLine("Count of Point Objects Created: {0}", GetCount());
        }
    }
}
```

Output:




3) Modify the program 2 by adding properties and indexers to access the private data members of the Point class.

Code:

```
// Declare a namespace named 'PointIndexer'.
namespace PointIndexer
{
    // Class to represent points with an indexer for accessing/modifying coordinates.
    class PointIndexer
    {
        // Data members to store the coordinates of a point.
        private int x;
        private int y;

        // Indexer for accessing/modifying coordinates.
        public int this[int index]
        {
            get
            {
                switch (index)
                {
                    case 0:
                        return x;
                    case 1:
                        return y;
                    default:
                        return 0;
```

```
            }//end of switch
        }//end of get
        set
        {
            switch (index)
            {
                case 0:
                    this.x = value;
                    break;
                case 1:
                    this.y = value;
                    break;
            }//end of switch
        }//end of set
    }//end of indexer

    // No-argument constructor initializes the point to default coordinates.
    public PointIndexer()
    {
        this.x = 10;
        this.y = 10;
    }

    // Two-argument constructor allows initializing the point with specific coordinates.
    public PointIndexer(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}

// Main method where the program execution begins.
class Program
{
    static void Main()
    {
        // Create PointIndexer objects using both constructors.
        PointIndexer point1 = new PointIndexer();
        PointIndexer point2 = new PointIndexer(20, 10);

        // Display the coordinates of point1 and point2 using the indexer.
        Console.WriteLine("Point 1 coordinates: ({0},{1})", point1[0], point1[1]);
        Console.WriteLine("Point 2 coordinates: ({0},{1})", point2[0], point2[1]);
    }
}
}
```

Output: