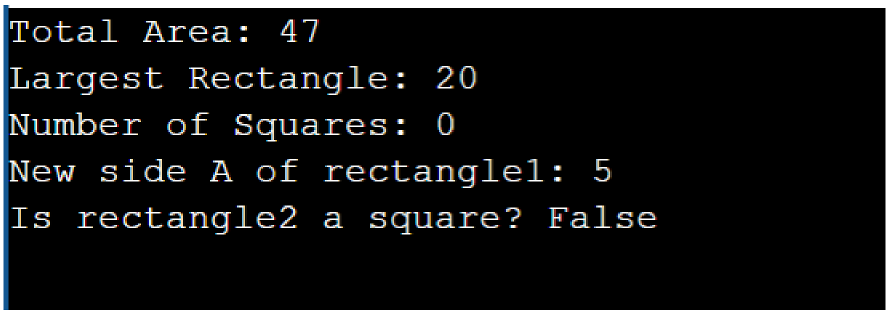
LAB-3

2100030723

```
1. using System;
using System.Linq;

public class Rectangle
{
   // Fields
   private double sideA;
   private double sideB;

   // Constructors
   public Rectangle(double a, double b)
   {
      sideA = a;
      sideB = b;
   }

   public Rectangle(double a)
   {
      sideA = a;
      sideB = 5; // Side B is always equal to 5
   }

   public Rectangle()
   {
      sideA = 4;
      sideB = 3;
   }
```

```
// Methods
public double GetSideA()
{
   return sideA;
}

public double GetSideB()
{
   return sideB;
}

public double Area()
{
   return sideA * sideB;
}

public double Perimeter()
{
   return 2 * (sideA + sideB);
}

public bool IsSquare()
{
   return sideA == sideB;
}

public void ReplaceSides()
{
   double temp = sideA;
   sideA = sideB;
   sideB = temp;
```

```
    }
}

public class ArrayRectangles
{
    private Rectangle[] rectangles;

    public ArrayRectangles(Rectangle[] rects)
    {
        rectangles = rects;
    }

    public double TotalArea()
    {
        double totalArea = 0;
        foreach (var rect in rectangles)
        {
            totalArea += rect.Area();
        }
        return totalArea;
    }

    public Rectangle LargestRectangle()
    {
        Rectangle largest = rectangles[0];
        foreach (var rect in rectangles)
        {
            if (rect.Area() > largest.Area())
            {
                largest = rect;
            }
        }
        return largest;
    }

    public int CountSquares()
    {
        return rectangles.Count(rect => rect.IsSquare());
    }

    // You can add more methods here according to your requirements
}

class Program
{
    static void Main(string[] args)
    {
        Rectangle rectangle1 = new Rectangle(4, 5);
        Rectangle rectangle2 = new Rectangle(3);
        Rectangle rectangle3 = new Rectangle();

        Rectangle[] rectangles = { rectangle1, rectangle2, rectangle3 };

        ArrayRectangles arrayRectangles = new ArrayRectangles(rectangles);

        Console.WriteLine("Total Area: " + arrayRectangles.TotalArea());
        Console.WriteLine("Largest Rectangle: " + arrayRectangles.LargestRectangle().Area());
        Console.WriteLine("Number of Squares: " + arrayRectangles.CountSquares());

        // Example of replacing sides
        rectangle1.ReplaceSides();
        Console.WriteLine("New side A of rectangle1: " + rectangle1.GetSideA());
        // Example of checking if a rectangle is a square
        Console.WriteLine("Is rectangle2 a square? " + rectangle2.IsSquare());
    }
}
```



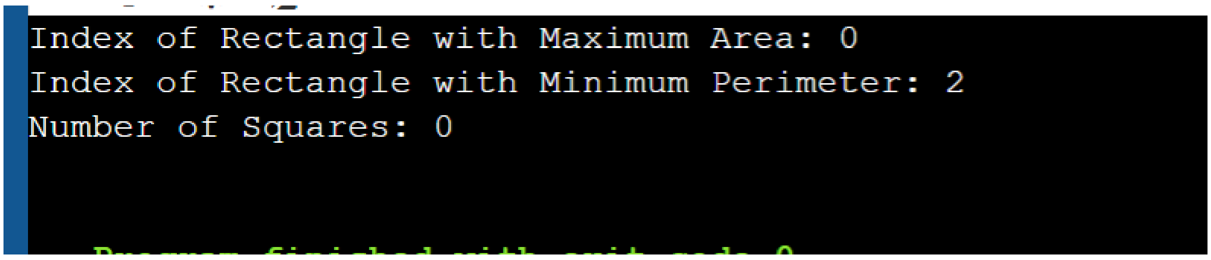
```
Total Area: 47
Largest Rectangle: 20
Number of Squares: 0
New side A of rectangle1: 5
Is rectangle2 a square? False
```

Task2

```
using System;
using System.Linq;

public class Rectangle
{
    // Fields
    private double sideA;
    private double sideB;

    // Constructors
    public Rectangle(double a, double b)
    {
        sideA = a;
        sideB = b;
    }

    public Rectangle(double a)
    {
        sideA = a;
        sideB = 5; // Side B is always equal to 5
    }

    public Rectangle()
    {
        sideA = 4;
        sideB = 3;
    }

    // Methods
    public double GetSideA()
    {
        return sideA;
    }

    public double GetSideB()
    {
        return sideB;
    }

    public double Area()
    {
        return sideA * sideB;
    }

    public double Perimeter()
    {
        return 2 * (sideA + sideB);
    }

    public bool IsSquare()
```

```
    {
        return sideA == sideB;
    }

    public void ReplaceSides()
    {
        double temp = sideA;
        sideA = sideB;
        sideB = temp;
    }
}

public class ArrayRectangles
{
    private Rectangle[] rectangleArray;

    public ArrayRectangles(int n)
    {
        rectangleArray = new Rectangle[n];
    }

    public ArrayRectangles(params Rectangle[] rectangles)
    {
        rectangleArray = rectangles;
    }

    public bool AddRectangle(Rectangle rectangle)
    {
        for (int i = 0; i < rectangleArray.Length; i++)
        {
            if (rectangleArray[i] == null)
        {
            rectangleArray[i] = rectangle;
            return true;
        }
    }
    return false;
}

public int NumberMaxArea()
{
    double maxArea = double.MinValue;
    int index = -1;
    for (int i = 0; i < rectangleArray.Length; i++)
    {
        if (rectangleArray[i] != null && rectangleArray[i].Area() > maxArea)
        {
            maxArea = rectangleArray[i].Area();
            index = i;
        }
    }
    return index;
}

public int NumberMinPerimeter()
{
    double minPerimeter = double.MaxValue;
    int index = -1;
    for (int i = 0; i < rectangleArray.Length; i++)
    {
        if (rectangleArray[i] != null && rectangleArray[i].Perimeter() < minPerimeter)
        {
                minPerimeter = rectangleArray[i].Perimeter();
                index = i;
            }
        }
        return index;
    }

    public int NumberSquare()
    {
        return rectangleArray.Count(rect => rect != null && rect.IsSquare());
    }
}

class Program
{
    static void Main(string[] args)
    {
        ArrayRectangles arrayRectangles = new ArrayRectangles(5);

        Rectangle rectangle1 = new Rectangle(4, 5);
        Rectangle rectangle2 = new Rectangle(3);
        Rectangle rectangle3 = new Rectangle();

        arrayRectangles.AddRectangle(rectangle1);
        arrayRectangles.AddRectangle(rectangle2);
        arrayRectangles.AddRectangle(rectangle3);

        Console.WriteLine("Index of Rectangle with Maximum Area: " +
arrayRectangles.NumberMaxArea());
        Console.WriteLine("Index of Rectangle with Minimum Perimeter: " +
arrayRectangles.NumberMinPerimeter());
        Console.WriteLine("Number of Squares: " + arrayRectangles.NumberSquare());
```

```
  }
}
```



```
Index of Rectangle with Maximum Area: 0
Index of Rectangle with Minimum Perimeter: 2
Number of Squares: 0


   ...Program finished with exit code 0
```