11. Longest Palindromic Substring

Problem: Return the longest palindromic substring in a given string

```
using System;
class Program
{
  static void Main()
  {
    string s = Console.ReadLine();
    if (string.IsNullOrEmpty(s))
    {
      Console.WriteLine("");
      return;
    }
    string longest = "";
    for (int i = 0; i < s.Length; i++)
    {
      // Odd length palindrome
      int l = i, r = i;
      while (l >= 0 && r < s.Length && s[l] == s[r])
      {
        if (r - l + 1 > longest.Length)
          longest = s.Substring(l, r - l + 1);
        l--;
        r++;
      }
      // Even length palindrome
      l = i; r = i + 1;
      while (l >= 0 && r < s.Length && s[l] == s[r])
      {
        if (r - l + 1 > longest.Length)
          longest = s.Substring(l, r - l + 1);
        l--;
        r++;
      }
    }
    Console.WriteLine(longest);  }}
```

12. Find Missing Ranges

Problem: Given a sorted array and a range [lower, upper], find missing ranges

```
using System;
using System.Collections.Generic;
class Program
{
  static void Main()
  {
        string line = Console.ReadLine();
        int[] nums = string.IsNullOrWhiteSpace(line) ? new int[0] : Array.ConvertAll(line.Split(), int.Parse);
        int lower = int.Parse(Console.ReadLine());
        int upper = int.Parse(Console.ReadLine());

        List<string> res = new List<string>();
        int prev = lower - 1;

        for (int i = 0; i <= nums.Length; i++)
        {
            int curr = (i < nums.Length) ? nums[i] : upper + 1;
            if (curr - prev > 1)
            {
                if (curr - prev == 2)
                    res.Add((prev + 1).ToString());
                else
                    res.Add((prev + 1) + "->" + (curr - 1));
            }
            prev = curr;
        }
        Console.WriteLine("[" + string.Join(", ", res) + "]");
    }
}
```

13. Find Peak Element

Problem: Return index of a peak element (greater than neighbors).

```
using System;
class Program
{
    static void Main()
    {
        int[] nums = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);
        int n = nums.Length;
        if (n == 1) { Console.WriteLine(0); return; }
        int peakIndex = 0;
        for (int i = 0; i < n; i++)
        {
            bool leftOk = (i == 0) || (nums[i] > nums[i - 1]);
            bool rightOk = (i == n - 1) || (nums[i] > nums[i + 1]);
            if (leftOk && rightOk)
                peakIndex = i; // keep updating instead of returning
        }
        Console.WriteLine(peakIndex);
    }
}
```

14. Find Kth Largest Element

Problem: Return the kth largest element in an array.

```
using System;

class Program
{
  static void Main()
  {
      string[] input = Console.ReadLine().Split();
      int n = input.Length;
      int[] nums = new int[n];
      for (int i = 0; i < n; i++)
          nums[i] = int.Parse(input[i]);

      int k = int.Parse(Console.ReadLine());


      for (int i = 0; i < n - 1; i++)
      {
          for (int j = i + 1; j < n; j++)
          {
              if (nums[i] > nums[j])
              {
                  int temp = nums[i];
                  nums[i] = nums[j];
                  nums[j] = temp;
              }
          }
      }

      // kth largest is at index n - k
      Console.WriteLine(nums[n - k]);
  }
}
```

15. Spiral Order Matrix Traversal

Problem: Return elements of a matrix in spiral order.

```
using System;
using System.Collections.Generic;
class Program
{
   static void Main()
   {
       Console.Write("Enter number of rows: ");
```

```
        int rows = int.Parse(Console.ReadLine());
        Console.Write("Enter number of columns: ");
        int cols = int.Parse(Console.ReadLine());
        int[,] matrix = new int[rows, cols];
        Console.WriteLine("Enter matrix elements row by row (space separated):");
        for (int i = 0; i < rows; i++)
        {
            string[] input = Console.ReadLine().Split(' ');
            for (int j = 0; j < cols; j++)
                matrix[i, j] = int.Parse(input[j]);
        }
        List<int> result = new List<int>();
        int top = 0, bottom = rows - 1, left = 0, right = cols - 1;
        while (top <= bottom && left <= right)
        {
            // Top row
            for (int i = left; i <= right; i++)
                result.Add(matrix[top, i]);
            top++;
            // Right column
            for (int i = top; i <= bottom; i++)
                result.Add(matrix[i, right]);
            right--;
            // Bottom row
            if (top <= bottom)
            {
                for (int i = right; i >= left; i--)
                    result.Add(matrix[bottom, i]);
                bottom--;
            }

            // Left column
            if (left <= right)
            {
                for (int i = bottom; i >= top; i--)
                    result.Add(matrix[i, left]);
                left++;
            }
        }

        Console.WriteLine("Spiral Order: " + string.Join(",", result));
    }
}
```

16. Find All Duplicates in Array

Problem: Return all elements that appear more than once.

```
using System;
using System.Collections.Generic;

class Program
{
  static void Main()
  {
    int[] nums = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);
    List<int> dup = new List<int>();

    for (int i = 0; i < nums.Length; i++)
    {
      for (int j = i + 1; j < nums.Length; j++)
      {
        if (nums[i] == nums[j] && !dup.Contains(nums[i]))
          dup.Add(nums[i]);
      }
    }

    Console.WriteLine("[" + string.Join(", ", dup) + "]");
  }
}
```

17. Find Longest Common Prefix

Problem: Return the longest common prefix among strings

```
using System;
class Program
{
  static void Main()
  {
    string[] words = Console.ReadLine().Split();
    if (words.Length == 0) { Console.WriteLine(""); return; }
    string prefix = words[0];
    for (int i = 1; i < words.Length; i++)
    {
      while (!words[i].StartsWith(prefix))
      {
        prefix = prefix.Substring(0, prefix.Length - 1);
        if (prefix == "") break;
      }
```

```
        }

        Console.WriteLine(prefix);  }}
```

18. Find All Palindromic Substrings

Problem: Count all palindromic substrings in a string

```
using System;

class Program
{
    static void Main()
    {
        string s = Console.ReadLine();
        int count = 0;

        for (int i = 0; i < s.Length; i++)
            for (int a = i, b = i; a >= 0 && b < s.Length && s[a] == s[b]; a--, b++, count++); // odd
        for (int i = 0; i < s.Length - 1; i++)
            for (int a = i, b = i + 1; a >= 0 && b < s.Length && s[a] == s[b]; a--, b++, count++); // even

        Console.WriteLine(count);
    }
}
```

19. Find Triplets with Zero Sum

Problem: Return all unique triplets that sum to zero.

```
using System;
using System.Collections.Generic;
class Program
{
    static void Main()
    {
        int[] a = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);
        var list = new List<string>();
        for (int i = 0; i < a.Length - 2; i++)
            for (int j = i + 1; j < a.Length - 1; j++)
                for (int k = j + 1; k < a.Length; k++)
                    if (a[i] + a[j] + a[k] == 0)
                    {
                        int[] t = { a[i], a[j], a[k] };
                        Array.Sort(t);
                        string triplet = $"[{t[0]},{t[1]},{t[2]}]";
                        if (!list.Contains(triplet)) list.Add(triplet);
        }

Console.WriteLine("[" + string.Join(", ", list) + "]");  }}
```