SET-2

1. **Validate Balanced Brackets**

Write a program that checks if a string containing brackets (), {}, and [] is balanced. A string is balanced if every opening bracket has a corresponding closing bracket in the correct order.

```
using System;
using System.Text.RegularExpressions;

class Program
{
  static void Main()
  {
    string input = Console.ReadLine();
    string pattern = @"\(\)|\[\]|\{}";

    while (Regex.IsMatch(input, pattern))
      input = Regex.Replace(input, pattern, "");

    Console.WriteLine(input.Length == 0);
  }
}
```

2. Find First Non-Repeating Character in a String

Given a string, return the first character that does not repeat. If all characters repeat, return null.

```
using System;
using System.Linq;

class Program
{
  static void Main()
  {
    Console.Write("Enter a string: ");
    string input = Console.ReadLine();

    char? result = input.FirstOrDefault(c => input.Count(x => x == c) == 1);

    if (result != '\0')
      Console.WriteLine($"{result}");
    else
      Console.WriteLine("null");
  }
}
```

3. Merge Two Sorted Arrays Without Duplicates

Write a function that merges two sorted arrays into one sorted array without duplicates.

```
using System;
using System.Linq;
class Program
{
  static void Main()
  {
    string input1 = Console.ReadLine();
    int[] arr1 = string.IsNullOrWhiteSpace(input1)
           ? new int[0]
           : input1.Split().Select(int.Parse).ToArray();


    string input2 = Console.ReadLine();
    int[] arr2 = string.IsNullOrWhiteSpace(input2)
           ? new int[0]
           : input2.Split().Select(int.Parse).ToArray();

    int[] merged = arr1.Concat(arr2).Distinct().OrderBy(x => x).ToArray();
    Console.WriteLine(string.Join(" ", merged));
  }
}
```

4. Count Pairs with Given Sum

Given an array of integers and a target sum, count the number of pairs that add up to the target.

```
using System;
class Program
{
  static void Main()
  {
    // Read array from user
    string[] input = Console.ReadLine().Split();
    int n = input.Length;
    int[] arr = new int[n];
    for (int i = 0; i < n; i++)
      arr[i] = int.Parse(input[i]);
    // Read target sum
    int target = int.Parse(Console.ReadLine());

    int count = 0;
    // Count pairs
    for (int i = 0; i < n; i++)
    {
      for (int j = i + 1; j < n; j++)
      {
        if (arr[i] + arr[j] == target)
          count++;
      }
    }
    Console.WriteLine(count);
}}
```

5. Find Longest Consecutive Sequence in Array
Given an unsorted array of integers, find the length of the longest consecutive elements sequence

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

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

        HashSet<int> set = new HashSet<int>(arr);
        int longest = 0;
        foreach (int num in arr)
        {
            if (!set.Contains(num - 1)) // start of a sequence
            {
                int current = num;
                int length = 1;
                while (set.Contains(current + 1))
                {
                    current++;
                    length++;
                }
                if (length > longest)
                longest = length;
            }
        }
        Console.WriteLine(longest);
    }
}
```

6. Find Majority Element in Array
Problem: Return the element that appears more than ⌊n/2⌋ times. If none, return null.

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

```
            arr[i] = int.Parse(input[i]);
        int majority = 0;
        bool found = false;
        for (int i = 0; i < n; i++)
        {
            int count = 0;
            for (int j = 0; j < n; j++)
            {
                if (arr[i] == arr[j])
                    count++;
            }
            if (count > n / 2)
            {
                majority = arr[i];
                found = true;
                break;
            }
        }
        Console.WriteLine(found ? majority.ToString() : "null");
    }
}
```

<u>7. Find All Subsets of a Set</u>
<u>Problem: Given a set of integers, return all possible subsets.</u>

```
ss Program
{
    static void Main()
    {
        string input = Console.ReadLine();
        int[] arr = string.IsNullOrWhiteSpace(input) ? new int[0] : Array.ConvertAll(input.Split(), int.Parse);

        int n = arr.Length;
        int total = 1 << n; // 2^n subsets

        Console.Write("[");
        for (int i = 0; i < total; i++)
        {
            Console.Write("[");
            bool first = true;
            for (int j = 0; j < n; j++)
            {
                if ((i & (1 << j)) != 0)
                {
                    if (!first) Console.Write(",");
                    Console.Write(arr[j]);
                    first = false;
                }
            }
            Console.Write("]");
            if (i < total - 1) Console.Write(", ");
        }
        Console.WriteLine("]");
    }
}
```

8. Binary Search in Rotated Array
Problem: Search for a target in a rotated sorted array.

```
using System;
public class Program
{
    public static void Main()
    {
        int[] arr = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);
        int target = int.Parse(Console.ReadLine());
        int left = 0, right = arr.Length - 1;
        while (left <= right)
        {
            int mid = (left + right) / 2;
            if (arr[mid] == target) { Console.WriteLine(mid); return; }
            if (arr[left] <= arr[mid])
            {
                if (target >= arr[left] && target < arr[mid]) right = mid - 1;
                else left = mid + 1;
            }
            else
            {
                if (target > arr[mid] && target <= arr[right]) left = mid + 1;
                else right = mid - 1;
            }
        }
        Console.WriteLine(-1);
    }
}
```

9. Sort Array by Frequency
Problem: Sort elements by frequency (highest first). If tie, sort by value

```
using System;
using System.Collections.Generic;
public class Program
{
    public static void Main()
    {
        int[] arr = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);
        Dictionary<int, int> freq = new Dictionary<int, int>();
        foreach (int num in arr)
        {
            if (freq.ContainsKey(num)) freq[num]++;
            else freq[num] = 1;
        }
        Array.Sort(arr, (a, b) =>
        {
            if (freq[b] != freq[a]) return freq[b] - freq[a];
            return a - b;
        });
        Console.WriteLine("[" + string.Join(", ", arr) + "]");
    }
}
```

<u>10. Group Anagrams</u>

<u>Problem: Group words that are anagrams of each other.</u>

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        var words = Console.ReadLine().Split();
        var grouped = words.GroupBy(w => new string(w.OrderBy(c => c).ToArray()))
                           .Select(g => g.ToList())
                           .ToList();

        Console.WriteLine("[" + string.Join(", ", grouped.Select(g => "[" + string.Join(", ", g) + "]")) + "]");
    }
}
```