

## 1.Validate Brackets

```
using System;
```

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

```
class Program
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        Console.Write("Enter a string with brackets: ");
```

```
        string input = Console.ReadLine();
```

```
        string pattern = @"\";
```

```
        while (Regex.IsMatch(input, pattern))
```

```
        {
```

```
            input = Regex.Replace(input, pattern, "");
```

```
        }
```

```
        if (input.Length == 0)
```

```
Console.WriteLine("True");
```

```
else
```

```
Console.WriteLine("False");
```

```
}
```

```
}
```

## **2.Non-repeating character**

```
using System;
```

```
using System.Linq;
```

```
public class Program
```

```
{
```

```
    public static void Main()
```

```
    {
```

```
        int t = int.Parse(Console.ReadLine());
```

```
        for (int i = 0; i < t; i++)
```

```
        {
```

```
            string s = Console.ReadLine();
```

```
            var ch = s.FirstOrDefault(c => s.Count(x => x == c) == 1);
```

```
            Console.WriteLine(ch == '\0' ? "null" : ch.ToString());
```

```
        }
```

```
    }
```

```
}
```

### **3.Mergeing sorted array**

```
using System;
```

```
using System.Linq;
```

```
public class Program
```

```
{
```

```
    public static void Main()
```

```
    {
```

```
        int[] arr1 = Console.ReadLine().Split().Select(int.Parse).ToArray();
```

```
        int[] arr2 = Console.ReadLine().Split().Select(int.Parse).ToArray();
```

```
        var merged = arr1.Concat(arr2).Distinct().OrderBy(x => x).ToArray();
```

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

```
    }
```

```
}
```

### **4.Target value question**

```
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 target = int.Parse(Console.ReadLine());

int count = 0;
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.Consecutive array**

```

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.Majority element in array**

```

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");
}
```

## 7. Find all subset of a set

using System;

```
class Program
{
    static void Main()
    {
        string input = Console.ReadLine();
        int[] arr = string.IsNullOrEmpty(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. Rotated binary search

```
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**

```
using System;
```

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

```
public class Program
```

```
{  
    public static void Main()  
    {  
        int[] arr = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);  
  
        // Count frequency  
        Dictionary<int, int> freq = new Dictionary<int, int>();  
        foreach (int num in arr)  
        {  
            if (freq.ContainsKey(num)) freq[num]++;  
            else freq[num] = 1;  
        }  
  
        // Sort by frequency (desc), then value (asc)  
        Array.Sort(arr, (a, b) =>  
        {  
            if (freq[b] != freq[a]) return freq[b] - freq[a]; // higher frequency first  
            return a - b; // smaller number first if tie  
        });  
    }  
}
```

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

## 10. Group Anagrams

```
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) + "]")) + "];
```

```
}
```

```
}
```

## 11.Longest Substring

```
using System;
```

```
class Program
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        string s = Console.ReadLine();
```

```
        if (string.IsNullOrEmpty(s))
```

```
        {
```

```
            Console.WriteLine("");
```

```
            return;
```

```
        }
```

```
        int start = 0, maxLength = 1;
```

```
        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 > maxLength)
```

```
                {
```

```
                    start = l;
```

```
                    maxLength = 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 > maxLength)
    {
        start = l;
        maxLength = r - l + 1;
    }
    l--;
    r++;
}
}

```

```

    Console.WriteLine(s.Substring(start, maxLength));
}
}

```

## 12.Finding Missing Range

```
using System;
```

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

```
class Program
```

```
{
```

```
    static void Main()
```

```
{
```

```
    int[] nums = Array.ConvertAll(Console.ReadLine().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

```
using System;
```

```

class Program
{
    static void Main()
    {
        string input = Console.ReadLine();
        input = input.Replace("[", "").Replace("]", "").Replace(";", " ");
        int[] nums = Array.ConvertAll(input.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries),
int.Parse);

        int n = nums.Length;
        if (n == 1) { Console.WriteLine(0); return; }
    }
}

```

```

int peakIndex = -1;

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;
}

Console.WriteLine(peakIndex);
}
}

```

#### **14.kth largest element**

```

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

```

using System;

```

```

class Program

```

```

{

```

```

    static void Main()

```

```

    {

```

```

        int[] rc = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);

```

```

        int rows = rc[0], cols = rc[1];

```

```
int[,] matrix = new int[rows, cols];  
for (int i = 0; i < rows; i++)  
{  
    int[] row = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);  
    for (int j = 0; j < cols; j++)  
        matrix[i, j] = row[j];  
}
```

```
int top = 0, bottom = rows - 1, left = 0, right = cols - 1;
```

```
Console.Write("[");
```

```
bool first = true;
```

```
while (top <= bottom && left <= right)
```

```
{  
    for (int i = left; i <= right; i++) Print(matrix[top, i], ref first);  
    top++;  
  
    for (int i = top; i <= bottom; i++) Print(matrix[i, right], ref first);  
    right--;  
  
    for (int i = right; i >= left; i--) Print(matrix[bottom, i], ref first);  
    bottom--;  
  
    for (int i = bottom; i >= top; i--) Print(matrix[i, left], ref first);  
    left++;  
}
```



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

```
static void Print(int val, ref bool first)  
{  
    if (!first) Console.Write(", ");  
    Console.Write(val);  
    first = false;  
}  
}
```

## 16. Duplicate element in array

```
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

```
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

```
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 sum 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) + "]";
}

```