1. **Find the Frequencies in a Sorted Array :**
   Our Task: Given a sorted array, **arr[]** consisting of **N** integers, the task is to find the frequencies of each array element.

Examples:

Input: arr[] = {1, 1, 1, 2, 3, 3, 5, 5, 8, 8, 8, 9, 9, 10}
Output:

Frequency of 1 is: 3
Frequency of 2 is: 1
Frequency of 3 is: 2
Frequency of 5 is: 2
Frequency of 8 is: 3
Frequency of 9 is: 2
Frequency of 10 is: 1

Input: arr[] = {2, 2, 6, 6, 7, 7, 7, 11}
Output:

Frequency of 2 is: 2
Frequency of 6 is: 2
Frequency of 7 is: 3
Frequency of 11 is: 1

2 **To Find Maximum Consecutive 1s :**

Our Task: Given a binary array, find the count of the maximum number of consecutive 1's present in the array.

Examples :

Input: arr[] = {1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1}
Output: 4

Input: arr[] = {0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1}
Output: 1

3. **Maximum Subarray Sum :**

Our Task: Given an array **arr[]**, the task is to find the elements of a contiguous subarray of numbers that has the largest sum.

Examples:

Input: arr = [-2, -3, 4, -1, -2, 1, 5, -3]
Output: [4, -1, -2, 1, 5]
Explanation:
In the above input, the maximum contiguous subarray sum is 7 and the elements of the subarray are [4, -1, -2, 1, 5]

Input: arr = [-2, -5, 6, -2, -3, 1, 5, -6]
Output: [6, -2, -3, 1, 5]
Explanation:
In the above input, the maximum contiguous subarray sum is 7 and the elements of the subarray are [6, -2, -3, 1, 5]

4. **Longest Even Odd Subarray :**

Our Task: Given an **array a[ ] of N** integers, the task is to find the length of the **longest Alternating Even Odd subarray present in the array**.

Input: *a[] = {1, 2, 3, 4, 5, 7, 9}*
Output: *5*
Explanation:
The subarray {1, 2, 3, 4, 5} has alternating even and odd elements.

Input: *a[] = {1, 3, 5}*
Output: *1*
Explanation:
There are only odd numbers, so we can count any one of them.

5, **Majority Element :**

Our Task: Find the majority element in the array. A ***majority element*** in an array A[] of size n is an element that appears more than n/2 times (and hence there is at most one such element).

Examples :

Input : *{3, 3, 4, 2, 4, 4, 2, 4, 4}*
Output : *4*
Explanation: *The frequency of 4 is 5 which is greater than the half of the size of the array size.*

Input : *{3, 3, 4, 2, 4, 4, 2, 4}*
Output : *No Majority Element*
Explanation: *There is no element whose frequency is greater than the half of the size of the array size.*

6. **Sliding Window Technique!!!**

Our Task: Given an array of integers of size 'n'. Our aim is to calculate the **maximum sum of 'k' consecutive elements in the array.**

Examples :

Input : arr[] = {100, 200, 300, 400}
k = 2
Output : 700

Input : arr[] = {1, 4, 2, 10, 23, 3, 1, 0, 20}
k = 4
Output : 39
We get maximum sum by adding subarray {4, 2, 10, 23}
of size 4.

Input : arr[] = {2, 3}
k = 3
Output : Invalid
There is no subarray of size 3 as size of whole
array is 2.

7. Subarray with given Sum !!!

Our Task: Given an array **arr[ ]** of non-negative integers and an integer **sum**, find a subarray that adds to a given **sum**.

Examples:

Input: arr[] = {1, 4, 20, 3, 10, 5}, sum = 33
Output: Sum found between indexes 2 and 4
Explanation: Sum of elements between indices 2 and 4 is 20 + 3 + 10 = 33

Input: arr[] = {1, 4, 0, 0, 3, 10, 5}, sum = 7
Output: Sum found between indexes 1 and 4
Explanation: Sum of elements between indices 1 and 4 is 4 + 0 + 0 + 3 = 7

Input: arr[] = {1, 4}, sum = 0
Output: No subarray found
Explanation: There is no subarray with 0 sum

8. In a realm where numbers hold secrets, a captivating challenge awaits, which is, **Finding the Equilibrium Point !!!**

Our Task: Given a sequence **arr[ ]** of size **n**, Write a function ***int equilibrium(int[] arr, int n)*** that returns an equilibrium index (if any) or -1 if no equilibrium index exists.

What is an Equilibrium Point?

The **equilibrium index of an array** is an index such that the sum of elements at lower indexes is equal to the sum of elements at higher indexes.

Examples:

Input*: A[] = {-7, 1, 5, 2, -4, 3, 0}*
Output*: 3 //index of 2*
3 is an equilibrium index, because:
A[0] + A[1] + A[2] = A[4] + A[5] + A[6]

***Input**: A[] = {1, 2, 3}*
***Output**: -1*

9. **Finding Maximum Appearing Element !!!**

Our Task: Given two arrays **L[ ]** and **R[ ]** of size **N** where L[i] and R[i] (**0 ? L[i], R[i] < 10^6**) denotes a range of numbers, the **task is to find the maximum occurred integer in all the ranges**. If more than one such integer exists, print the smallest one.

Examples:

***Input:** L[ ] = {1, 4, 3, 1}, R[ ] = {15, 8, 5, 4}*
***Output:** 4*

Explanation: Overall ranges are: {1,2,3,4,5,6,7,8,9,10,11,12,13,14 15}, {4,5,6,7,8}, {3,4,5}, {1,2,3,4}.

In all these ranges, 4 appears the most times.
***Input:** L[ ] = {1, 5, 9, 13, 21}, R[ ] = {15, 8, 12, 20, 24}*
***Output:** 5*
Explanation: Overall Ranges are: {1,2,3,4,5,6,7,8,9,10,11,12,13,14 15}, {5,6,7,8}, {9,10,11,12}, {13,14,15,16,17,18,19,20},{21,22,23,24}
In these ranges, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 all appear 2 times. The smallest number among all are 5.

10. Reverse array in groups
Given an array **arr** of positive integers. Reverse every sub-array group of size **k.**

Note: If at any instance, k is greater or equal to the array size, then reverse the entire array. You shouldn't return any array, modify the given array in place.

Examples:

Input: k=3, arr= [1, 2, 3, 4, 5]
Output: [3, 2, 1, 5, 4]
Explanation: First group consists of elements 1, 2, 3. Second group consists of 4,5.

Input: k = 5, arr = [5, 6, 8, 9]
Output: [9, 8, 6, 5]
Explnation: Since k is greater than array size, the entire array is reversed.

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(1)