MAGIC SESSION

Problem 1:

Given an array containing n integers. The problem is to find the sum of the elements of the contiguous subarray having the smallest(minimum) sum.

Examples:

Input : arr[] = {3, -4, 2, -3, -1, 7, -5}
Output : -6
Subarray is {-4, 2, -3, -1} = -6
Input : arr = {2, 6, 8, 1, 4}
Output : 1

Problem 2:

Given an array of non-negative integers representing a number, implement a function to simulate the carry forward operation that occurs when adding 1 to the number represented by the array. The array represents the digits of the number, where the 0th index is the least significant digit. Your task is to handle the carry forward operation correctly, updating the array accordingly. The function should return the resulting array.

For example, given the input array [1, 9, 9], representing the number 199, the function should return [2, 0, 0], representing the result of adding 1 to 199 with the carry forward properly handled.

Consider edge cases such as when the number has trailing zeros or when the carry forward results in an additional digit. Optimize your solution for efficiency and discuss the time and space complexity of your algorithm.

Problem 3:
Given a string s and an array of string words, determine whether s is a prefix string of words.

A string s is a prefix string of words if s can be made by concatenating the first k strings in words for some positive k no larger than words.length.

Return true if s is a prefix string of words, or false otherwise.

Example 1:

Input: s = "iloveleetcode", words = ["i","love","leetcode","apples"]
Output: true
Explanation:
s can be made by concatenating "i", "love", and "leetcode" together.

Example 2:

Input: s = "iloveleetcode", words = ["apples","i","love","leetcode"]
Output: false
Explanation:
It is impossible to make s using a prefix of arr.

Constraints:

1 <= words.length <= 100
1 <= words[i].length <= 20
1 <= s.length <= 1000
words[i] and s consist of only lowercase English letters.

Problem 4:
Given an array of positive integers nums and a positive integer target, return the minimal length of a
subarray
whose sum is greater than or equal to target. If there is no such subarray, return 0 instead.

Example 1:

Input: target = 7, nums = [2,3,1,2,4,3]
Output: 2
Explanation: The subarray [4,3] has the minimal length under the problem constraint.
Example 2:

Input: target = 4, nums = [1,4,4]
Output: 1
Example 3:

Input: target = 11, nums = [1,1,1,1,1,1,1,1]
Output: 0

Constraints:

1 <= target <= 109
1 <= nums.length <= 105
1 <= nums[i] <= 104

Problem 5

Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N. Find the missing element.

Example 1:

Input:
N = 5
A[] = {1,2,3,5}
Output: 4
Example 2:

Input:
N = 10
A[] = {6,1,2,8,3,4,7,10,5}
Output: 9

Your Task :
You don't need to read input or print anything. Complete the function MissingNumber() that takes array and N as input parameters and returns the value of the missing number.

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

Constraints:
1 ≤ N ≤ 106
1 ≤ A[i] ≤ 106

Problem 6

Given an array A of n positive numbers. The task is to find the first equilibrium point in an array. Equilibrium point in an array is an index (or position) such that the sum of all elements before that index is the same as the sum of elements after it.

Note: Return equilibrium point in 1-based indexing. Return -1 if no such point exists.

Example 1:

Input:
n = 5
A[] = {1,3,5,2,2}
Output:
3
Explanation:
equilibrium point is at position 3 as sum of elements before it (1+3) = sum of elements after it (2+2).
Example 2:

Input:
n = 1
A[] = {1}
Output:
1

Explanation:
Since there's only one element hence it's only the equilibrium point.
Your Task:
The task is to complete the function equilibriumPoint() which takes the array and n as input parameters and returns the point of equilibrium.

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

Constraints:
1 <= n <= 105
1 <= A[i] <= 109

Problem 7

Given an array Arr[] that contains N integers (may be positive, negative or zero). Find the product of the maximum product subarray.

Example 1:

Input:
N = 5
Arr[] = {6, -3, -10, 0, 2}
Output: 180
Explanation: Subarray with maximum product
is [6, -3, -10] which gives the product as 180.
Example 2:

Input:
N = 6
Arr[] = {2, 3, 4, 5, -1, 0}
Output: 120
Explanation: Subarray with maximum product
is [2, 3, 4, 5] which gives the product as 120.

Problem 8

Given an unsorted array arr[] of size N. Rotate the array to the left (counter-clockwise direction) by D steps, where D is a positive integer.

Example 1:

Input:
N = 5, D = 2
arr[] = {1,2,3,4,5}
Output: 3 4 5 1 2
Explanation: 1 2 3 4 5 when rotated
by 2 elements, it becomes 3 4 5 1 2.

Example 2:

Input:
N = 10, D = 3
arr[] = {2,4,6,8,10,12,14,16,18,20}
Output: 8 10 12 14 16 18 20 2 4 6
Explanation: 2 4 6 8 10 12 14 16 18 20
when rotated by 3 elements, it becomes
8 10 12 14 16 18 20 2 4 6.

Example 9

Given a sorted array arr containing n elements with possibly some duplicate, the task is to find the first and last occurrences of an element x in the given array.
Note: If the number x is not found in the array then return both the indices as -1.

Example 1:

Input:
n=9, x=5
arr[] = { 1, 3, 5, 5, 5, 5, 67, 123, 125 }
Output:
2 5
Explanation:
First occurrence of 5 is at index 2 and last occurrence of 5 is at index 5.
Example 2:

Input:
n=9, x=7
arr[] = { 1, 3, 5, 5, 5, 5, 7, 123, 125 }
Output:
6 6
Explanation:
First and last occurrence of 7 is at index 6.

Example 10

You are given an array of size N containing integers. Your task is to find the number of subarrays that can be formed from the given array. A subarray is defined as a contiguous sequence of elements in the array.

Input:

arr = [1, 2, 3]

Output:
6

Explanation:

In this example, the possible subarrays are [1], [2], [3], [1, 2], [2, 3], and [1, 2, 3], so the total count is 6.