Assignment - 6

1. **Find the Majority Element in an Array**

Problem Statement:

Given an array of size n, find the majority element. The majority element is the element that appears more than n/2 times. You may assume that the array always contains a majority element.

Input:

- A single integer n ($1 \le n \le 10^5$) — size of the array.
- An array arr of n integers ($1 \le arr[i] \le 10^9$)

Output:

- A single integer — the majority element.

Input:

```
7

3 3 4 2 3 3 3
```

Output:

```
3
```

Program:

```
import java.util.*;

public class Main {

  public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int count = 0, candidate = 0;

    for (int i = 0; i < n; i++) {

      int num = sc.nextInt();
```

```
            if (count == 0)
            candidate = num;
            count += (num == candidate) ? 1 : -1;
        }
        System.out.println(candidate);
    }
}
```

2. Solve the Maximum Subarray Sum Problem (Kadane's Algorithm)

Problem Statement:

Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Input:

- A single integer n ($1 \le n \le 10^5$) — number of elements.
- An array nums of n integers ($-10^4 \le nums[i] \le 10^4$)

Output:

- A single integer — the maximum subarray sum.

Input:

```
9
-2 1 -3 4 -1 2 1 -5 4
```

Output:

```
6
```

Program:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int first = sc.nextInt();
        int maxSum = first;
        int currentSum = first;
        for (int i = 1; i < n; i++) {
            int num = sc.nextInt();
            currentSum = Math.max(num, currentSum + num);
            maxSum = Math.max(maxSum, currentSum);
        }
        System.out.println(maxSum);
    }
}
```

3. Find the First Non-Repeating Character in a String

Problem Statement:

Given a string s, find the first non-repeating character and return its index. If no non-repeating character exists, return -1.

Input:

- A string s of lowercase English letters ($1 \leq |s| \leq 10^5$)

Output:

- A single integer — index of the first non-repeating character or -1.

Input:

Mountain

Output:

0

Input:

aabb

Output:

-1

Program:

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String s = sc.nextLine().toLowerCase();

        int[] count = new int[26];

        for (int i = 0; i < s.length(); i++) {

            count[s.charAt(i) - 'a']++;

        }

        for (int i = 0; i < s.length(); i++) {

            if (count[s.charAt(i) - 'a'] == 1) {

                System.out.println(i);

                return;

            }

        }

        System.out.println(-1);

    }

}
```

4. **Problem Statement:**

Given two strings s1 and s2, check if s2 is a **rotation** of s1 using only one call to a substring-checking method (or equivalent logic). A rotation means that the characters are shifted in a circular manner.

For example:
s1 = "waterbottle" and s2 = "erbottlewat" → True
s1 = "hello" and s2 = "lohel" → True

Input:

- Two strings s1 and s2 consisting of lowercase or uppercase letters only.
- 1 ≤ |s1|, |s2| ≤ 1000

Output:

- Print True if s2 is a rotation of s1, otherwise False.

Input:

waterbottle

erbottlewat

Output:

True

Input:

hello

lohel

Output:

True

Input:

abc

acb

Output:

False

Program:

```
import java.util.Scanner;
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String s1 = sc.nextLine();
    String s2 = sc.nextLine();
    if (s1.length() == s2.length() CC (s1 + s1).contains(s2)) {
      System.out.println("True");
    } else {
      System.out.println("False");
    }
  }
}
```