TRY SOLVING IT

One-Dimensional Arrays

1. **Find the Second Largest Element in an Array**
   - Write a program to find the second largest element in an array.
   - Example Input: `[12, 35, 1, 10, 34, 1]`
   - Expected Output: `Second largest element is 34.`

2. **Rearrange Array in Alternating Positive and Negative Numbers**
   - Rearrange an array in such a way that positive and negative numbers alternate, while maintaining the relative order of positive and negative numbers.
   - Example Input: `[1, 2, -3, -4, -1, 4]`
   - Expected Output: `Rearranged array: [-3, 1, -4, 2, -1, 4].`

3. **Find the Majority Element in an Array**
   - A majority element in an array is one that appears more than `n/2` times (where `n` is the size of the array). Write a program to find this element, if it exists.
   - Example Input: `[3, 3, 4, 2, 4, 4, 2, 4, 4]`
   - Expected Output: `Majority element is 4.`

Multi-Dimensional Arrays

1. **Transpose of a Matrix**
   - Write a program to find the **transpose** of a matrix. The transpose of a matrix is formed by swapping the rows with columns.
   - Example Input:

```
[
  [1, 2, 3],
  [4, 5, 6]
]
```

   - Expected Output:

```
[
  [1, 4],
  [2, 5],
  [3, 6]
]
```

2. **Search an Element in a Sorted 2D Matrix**
    - Given a 2D matrix where each row is sorted and the first element of each row is greater than the last element of the previous row, write a program to search for an element.
    - Example Input:

```
[
 [1, 3, 5, 7],
 [10, 11, 16, 20],
 [23, 30, 34, 50]
]
```

Target: `3`

    - Expected Output: `Element found at [0, 1].`

3. **Rotate Matrix by 90 Degrees Clockwise**
    - Write a program to rotate a given `n x n` matrix by 90 degrees clockwise.
    - Example Input:

```
[
 [1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]
]
```

    - Expected Output:

```
[
 [7, 4, 1],
 [8, 5, 2],
 [9, 6, 3]
]
```

Strings

1. **Remove Duplicates from a String**
    - Write a program to remove duplicate characters from a string.
    - Example Input: `"geeksforgeeks"`
    - Expected Output: `"geksfor"`.
2. **Find the First Non-Repeating Character in a String**
    - Given a string, find the first non-repeating character and return its index. If it doesn't exist, return -1.
    - Example Input: `"leetcode"`
    - Expected Output: `First non-repeating character is 'l' at index 0.`
3. **Check if Two Strings are Rotations of Each Other**
    - Write a program to check if two strings are **rotations** of each other.
    - Example Input: `"ABCD"` and `"CDAB"`
    - Expected Output: `Strings are rotations of each other.`