Experiment- 4

Student Name: Aman Bansal **UID: 22BCS13365**
Branch: BE-CSE **Section/Group:** KPIT-901/B
Semester: 6th **Date of Performance:**14/02/25
Subject Name: AP Lab - 2 **Subject Code:** 22CSP-351

1. **Aim:** Merge sorted array

2. **Objective:**

You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively.

Merge nums1 and nums2 into a single array sorted in non-decreasing order.

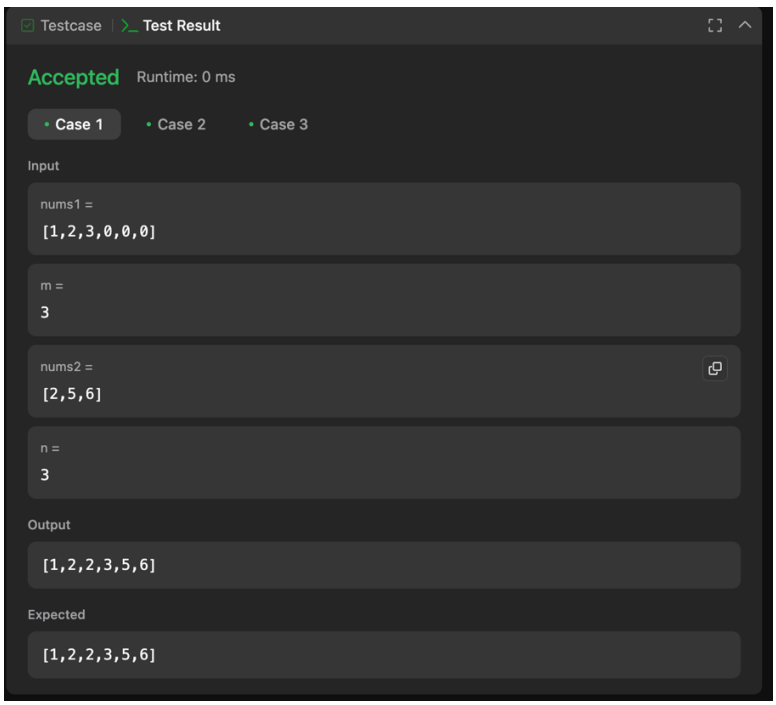
The final sorted array should not be returned by the function, but instead be stored inside the array nums1. To accommodate this, nums1 has a length of m + n, where the first m elements denote the elements that should be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n.

3. **Implementation/Code:**

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        for (int j = 0, i = m; j<n; j++){
```

```
            nums1[i] = nums2[j];
            i++;
        }
        sort(nums1.begin(),nums1.end());
    }
};
```

4.Output :



5.Learning Outcome:

1. **Understand the merging process of two sorted arrays in-place.**
2. **Learn how to efficiently manipulate array indices while merging two lists.**
3. **Implement an optimal merging approach using the two-pointer technique.**
4. **Analyze the time complexity of the merging process, which runs in O(m + n) time.**

QUESTION 2

1. **Aim:** Wiggle sort II.

2. **Objective:**

   Given an integer array nums, reorder it such that nums[0] < nums[1] > nums[2] < nums[3]....

   You may assume the input array always has a valid answer.

   Example 1:

   Input: nums = [1,5,1,1,6,4]
   Output: [1,6,1,5,1,4]
   Explanation: [1,4,1,5,1,6] is also accepted.

   Example 2:

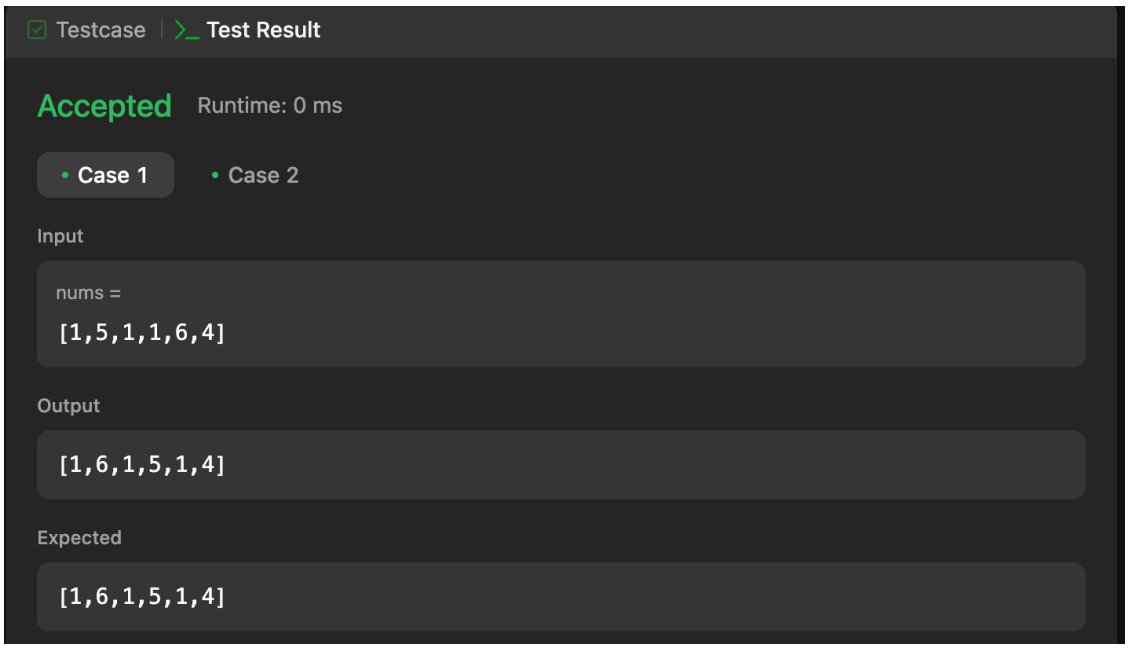
   Input: nums = [1,3,2,2,3,1]
   Output: [2,3,1,3,1,2]

3. **Implementation/Code:**

```
class Solution {
public:
  void wiggleSort(vector<int>& nums) {
   vector<int> sorted(nums);
   sort(sorted.begin(), sorted.end());
```

```
    for (int i=nums.size()-1, j=0, k=i/2+1; i>=0; i--)
        nums[i] = sorted[i&1 ? k++ : j++];
}
};
```

4.Output:



5.Learning Outcome:

- **Understand the concept of Wiggle Sort and its application in array manipulation.**
- **Learn how to rearrange elements such that nums[0] ≤ nums[1] ≥ nums[2] ≤ nums[3] ...**
- **Implement efficient in-place sorting techniques without using extra space.**
- **Analyze the time complexity of O(n) for the optimal linear-time approach.**