Experiment 4

Student Name: Suryansh **UID:** 22BCS15110
Branch: BE-CSE **Section/Group:** 22BCS_IOT-638/B
Semester: 6th **Subject Code:** 22CSP-351
Subject Name: AP Lab-II

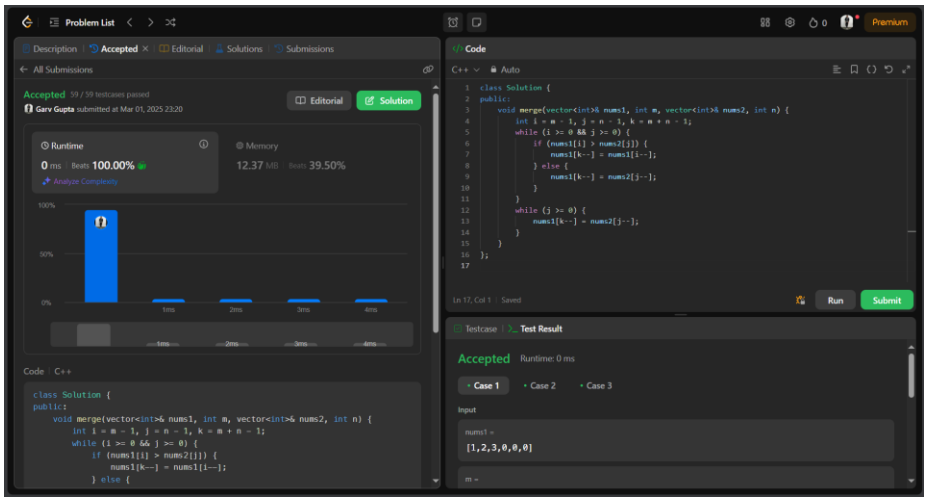
A. Merge Sorted Array

1. **Aim:** 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.

2. Code

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i = m - 1, j = n - 1, k = m + n - 1;
        while (i >= 0 && j >= 0) {
            if (nums1[i] > nums2[j]) {
                nums1[k--] = nums1[i--];
            } else {
                nums1[k--] = nums2[j--];
            }
        }
        while (j >= 0) {
            nums1[k--] = nums2[j--];
        }
    }
};
```

3. Output:



4. **Link: https://leetcode.com/problems/merge-sorted-array/submissions/1560618558/**

B. First Bad Version

1. **Aim:** You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad. Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following ones to be bad. You are given an API bool isBadVersion(version) which returns whether version is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

2. Code

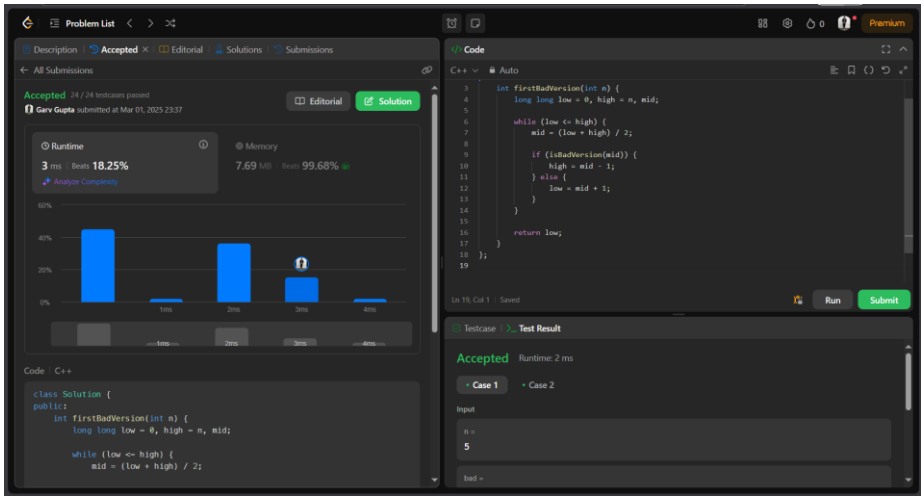
```
class Solution {
public:
    int firstBadVersion(int n) {
        long long low = 0, high = n, mid;

        while (low <= high) {
            mid = (low + high) / 2;

            if (isBadVersion(mid)) {
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }
```

```
            return low;
        }
    };
```

3. **Output:**



4. **Link: https://leetcode.com/problems/first-bad-version/submissions/1560619739/**

C. Sort Colors

1. **Aim:** Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue. We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively. You must solve this problem without using the library's sort function.

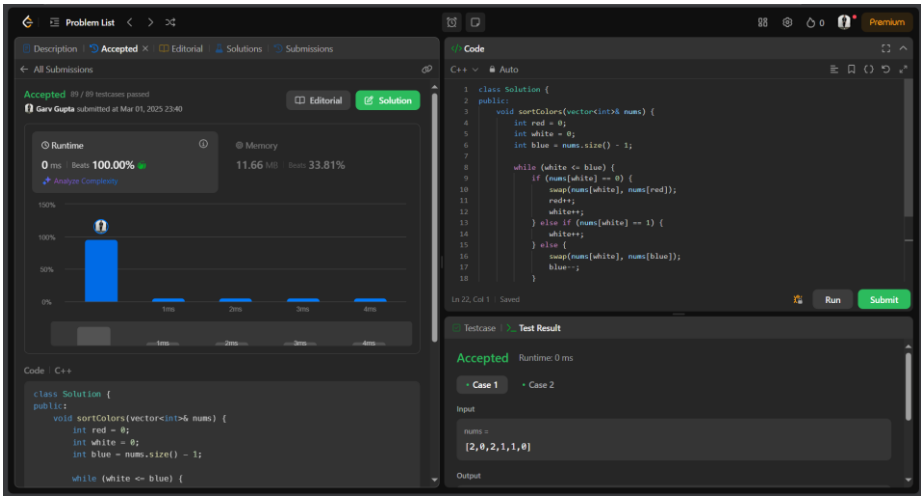
2. **Code**

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int red = 0;
        int white = 0;
        int blue = nums.size() - 1;

        while (white <= blue) {
            if (nums[white] == 0) {
                swap(nums[white], nums[red]);
                red++;
                white++;
```

```
            } else if (nums[white] == 1) {
                white++;
            } else {
                swap(nums[white], nums[blue]);
                blue--;
            }
        }
    }
};
```

3. Output:



4. Link: https://leetcode.com/problems/sort-colors/description/