Assignment NO. 5

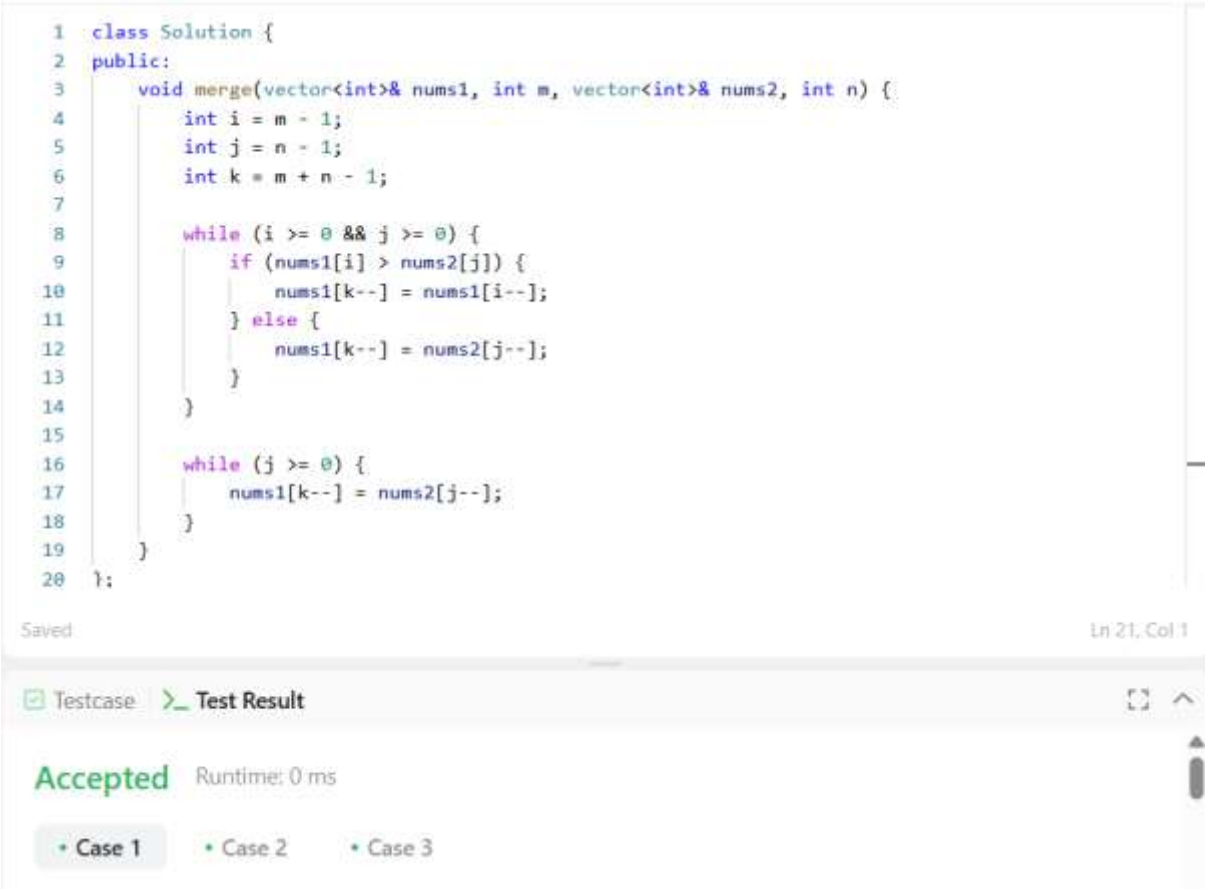
Name: Sahil

UID: 22BCS15237

Section: 608(B)

Subject: AP-2

1. Merge Sorted Array: https://leetcode.com/problems/merge-sorted-array/



```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i = m - 1;
        int j = n - 1;
        int 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--];
        }
    }
};
```

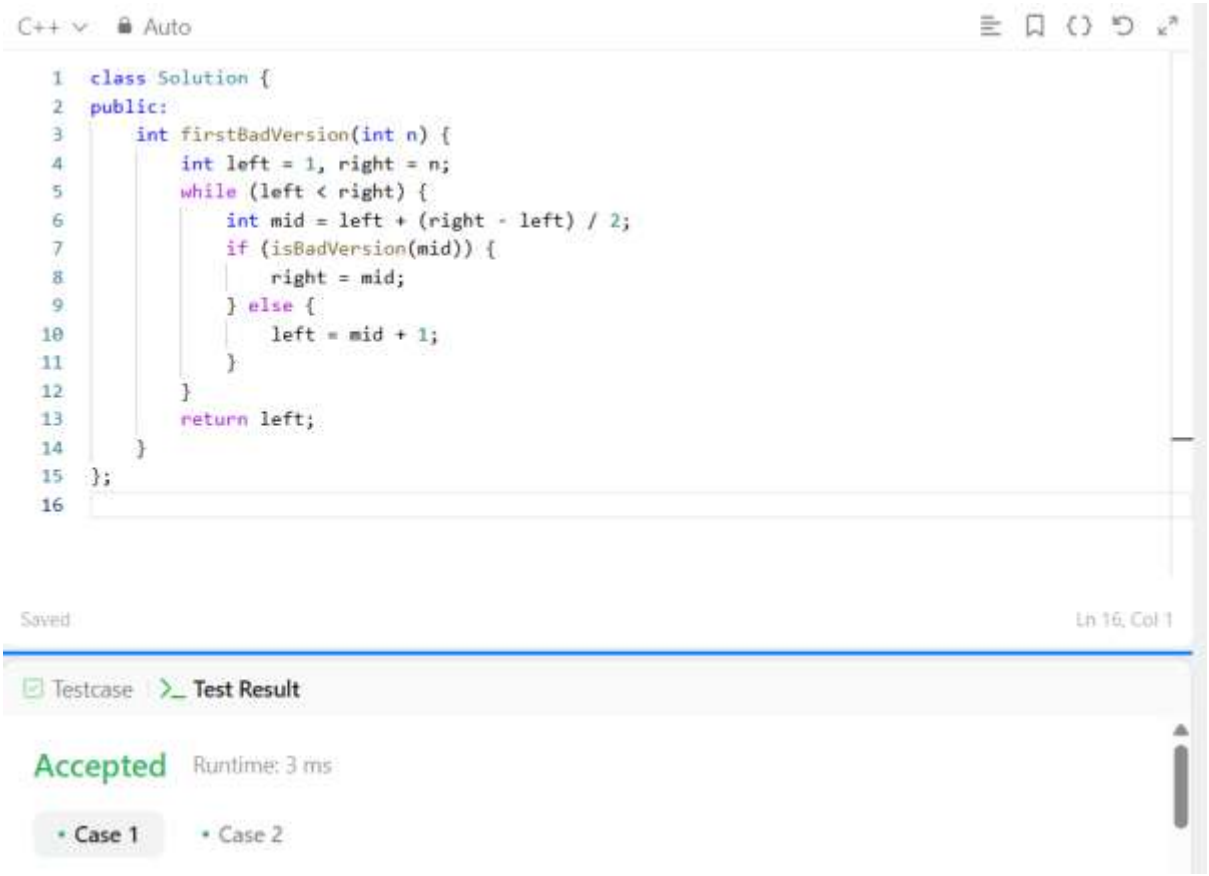
Saved Ln 21, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

2. First Bad Version: https://leetcode.com/problems/first-bad-version/



3. Sort Colors: https://leetcode.com/problems/sort-colors/

```cpp
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low = 0, mid = 0, high = nums.size() - 1;

        while (mid <= high) {
            if (nums[mid] == 0) {
                swap(nums[low++], nums[mid++]);
            } else if (nums[mid] == 1) {
                mid++;
            } else {
                swap(nums[mid], nums[high--]);
            }
        }
    }
};
```

Saved

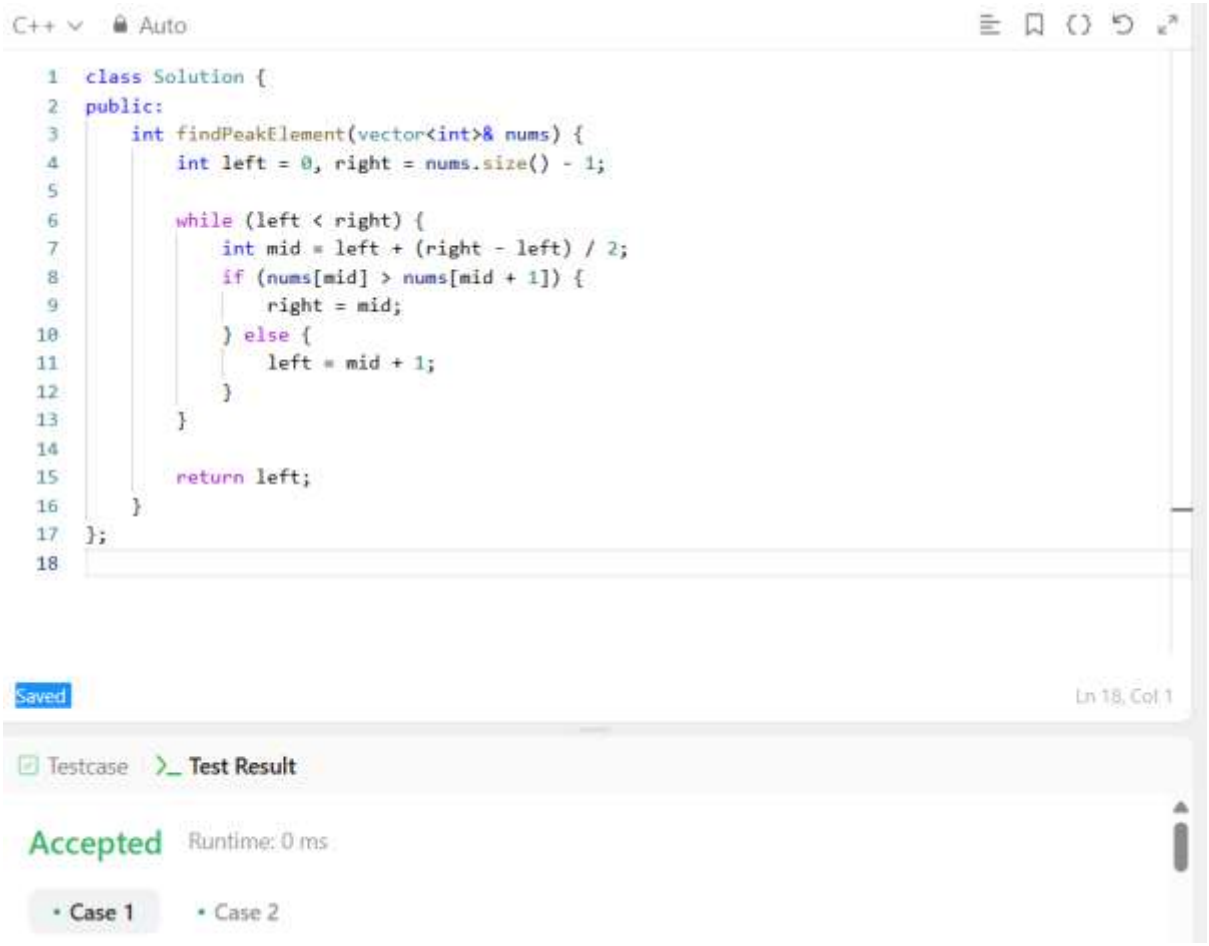
Ln 17, Col 1

Testcase >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

4. Find Peak Element: https://leetcode.com/problems/find-peak-element/



5. Median of Two Sorted Arrays: https://leetcode.com/problems/median-of-two-sorted-arrays/

Code

C++ Auto

```cpp
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int m = nums1.size(), n = nums2.size();
        if (m > n) return findMedianSortedArrays(nums2, nums1);

        int left = 0, right = m, total = (m + n + 1) / 2;

        while (left <= right) {
            int mid1 = left + (right - left) / 2;
            int mid2 = total - mid1;

            int l1 = (mid1 > 0) ? nums1[mid1 - 1] : INT_MIN;
            int l2 = (mid2 > 0) ? nums2[mid2 - 1] : INT_MIN;
            int r1 = (mid1 < m) ? nums1[mid1] : INT_MAX;
            int r2 = (mid2 < n) ? nums2[mid2] : INT_MAX;

            if (l1 <= r2 && l2 <= r1) {
                if ((m + n) % 2 == 0) {
                    return (max(l1, l2) + min(r1, r2)) / 2.0;
                } else {
                    return max(l1, l2);
                }
            } else if (l1 > r2) {
                right = mid1 - 1;
            } else {
                left = mid1 + 1;
            }
        }

        return 0.0;
    }
};

```