# Experiment 5

**Student Name: Ananya Akhouri**

**Branch: CSE**

**UID: 22BCS10289**

**Section/Group:IOT-641-A**

**Semester: 6th DOP:11/02/2025**

**Subject: Java Lab Subject Code: 22CSH-359**

**Aim:** Merge sort array

**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**.**

# 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++;

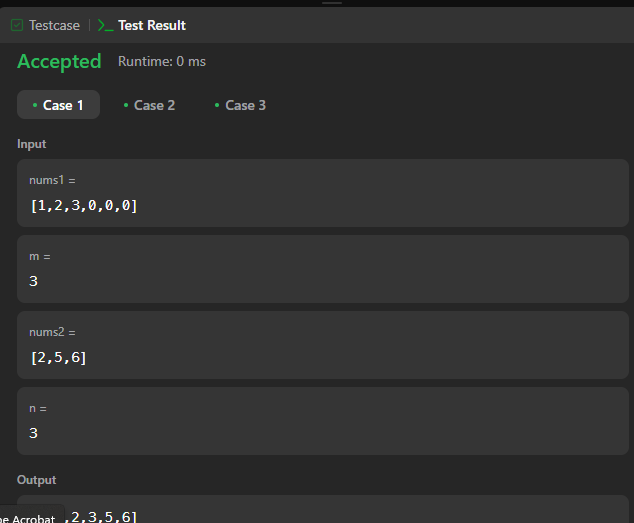
}

sort(nums1.begin(),nums1.end());

}

};

# Output:



**Question 2**

**Aim:** Merge Interval

**Objective:** Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input**.**

**Code:**class Solution {

private:

void fast(){

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

cout.tie(NULL);

}

public:

vector<vector<int>> merge(vector<vector<int>>& intervals) {

fast();

sort(intervals.begin(), intervals.end());

vector<vector<int>> res;

for(auto num: intervals){

if(res.empty() || res.back()[1] < num[0]){

res.push\_back(num);

}

else{

res.back()[1]=max(res.back()[1], num[1]);

}

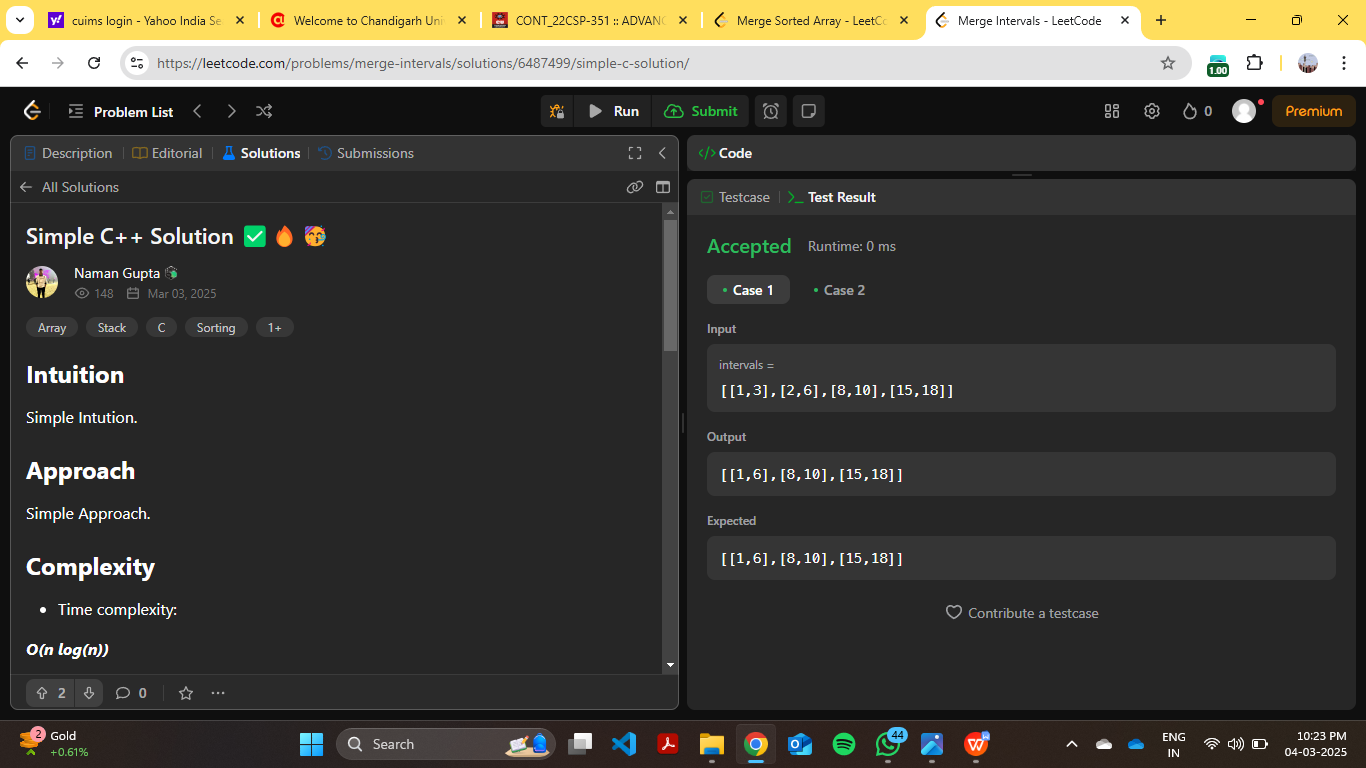
}

return res;

}

};

**Output:**

****

**Question 3**

**Aim:** Sort Colour

**Objective:**

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.

**Code:**

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]);

low++;

mid++;

}

else if(nums[mid] == 1){

mid++;

}

else{

swap(nums[mid], nums[high]);

high--;

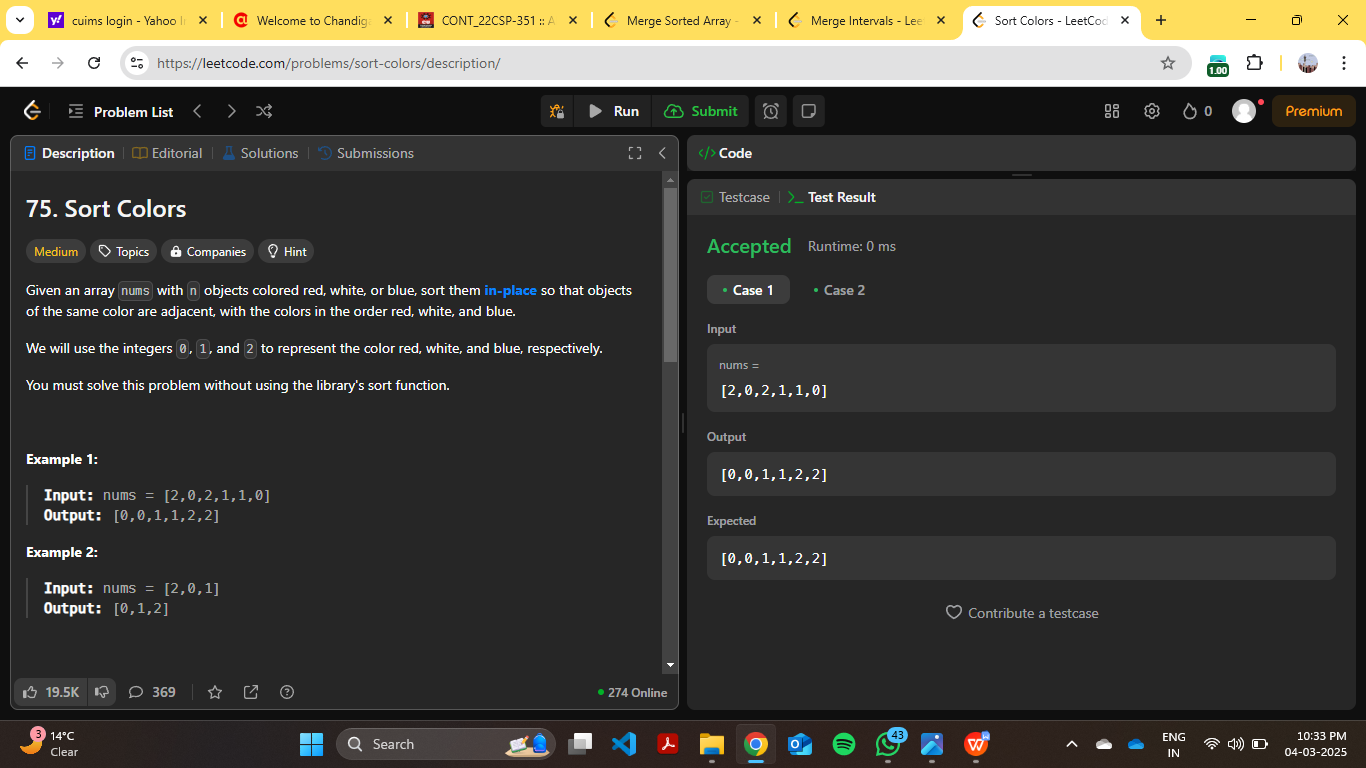
}

}

}

};

**Output:**

****