“**Experiment1.1”**

**Student Name: SUMIT KUMAR UID: 20BCS8226**

**Branch: CSE Section/Group: DM-720 A**

**Semester: 6th Date of Performance: 10-02-23**

**Subject Name: Competitive Coding-II Subject Code: 20CSP-351**

**Aim:**

To implement the concept of Arrays, Queues and Stack and Linked List.

**Objective:**

• The objective is to build problem solving capability and to learn the basic concepts of data structures.

• The implementation of Jump Game-II which shows and brushes up the concept of arrays.

• The implementation of removing the duplicity in the Sorted-List-II.

**Problem 1: “3 Sum”**

<https://leetcode.com/problems/3sum/>

Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.

Notice that the solution set must not contain duplicate triplets.

**Code:**

classSolution {

public:

    vector<vector<int>>threeSum(vector<int>&nums) {

        sort(nums.begin(), nums.end());     // O(nlogn)

        vector<vector<int>>ans;        // SC: O(n)

        for(inti=0;i<=nums.size()-3;i++){          // O(n)

            if(i==0 || (i>0&&nums[i]!=nums[i-1])){

                int lo=i+1, hi=nums.size()-1;

                while(lo < hi){                     // O(n)

                    if(nums[lo]+nums[hi]==0-nums[i]){

                        vector<int> temp;

                        temp.push\_back(nums[i]);

                        temp.push\_back(nums[lo]);

                        temp.push\_back(nums[hi]);

                        ans.push\_back(temp);

                        while(lo<hi &&nums[lo]==nums[lo+1]) lo++;

                        while(lo<hi &&nums[hi]==nums[hi-1]) hi--;

                        lo++; hi--;

                    }

                    elseif(nums[lo]+nums[hi] <0-nums[i]) lo++;

                    else hi--;

                }

            }

        }

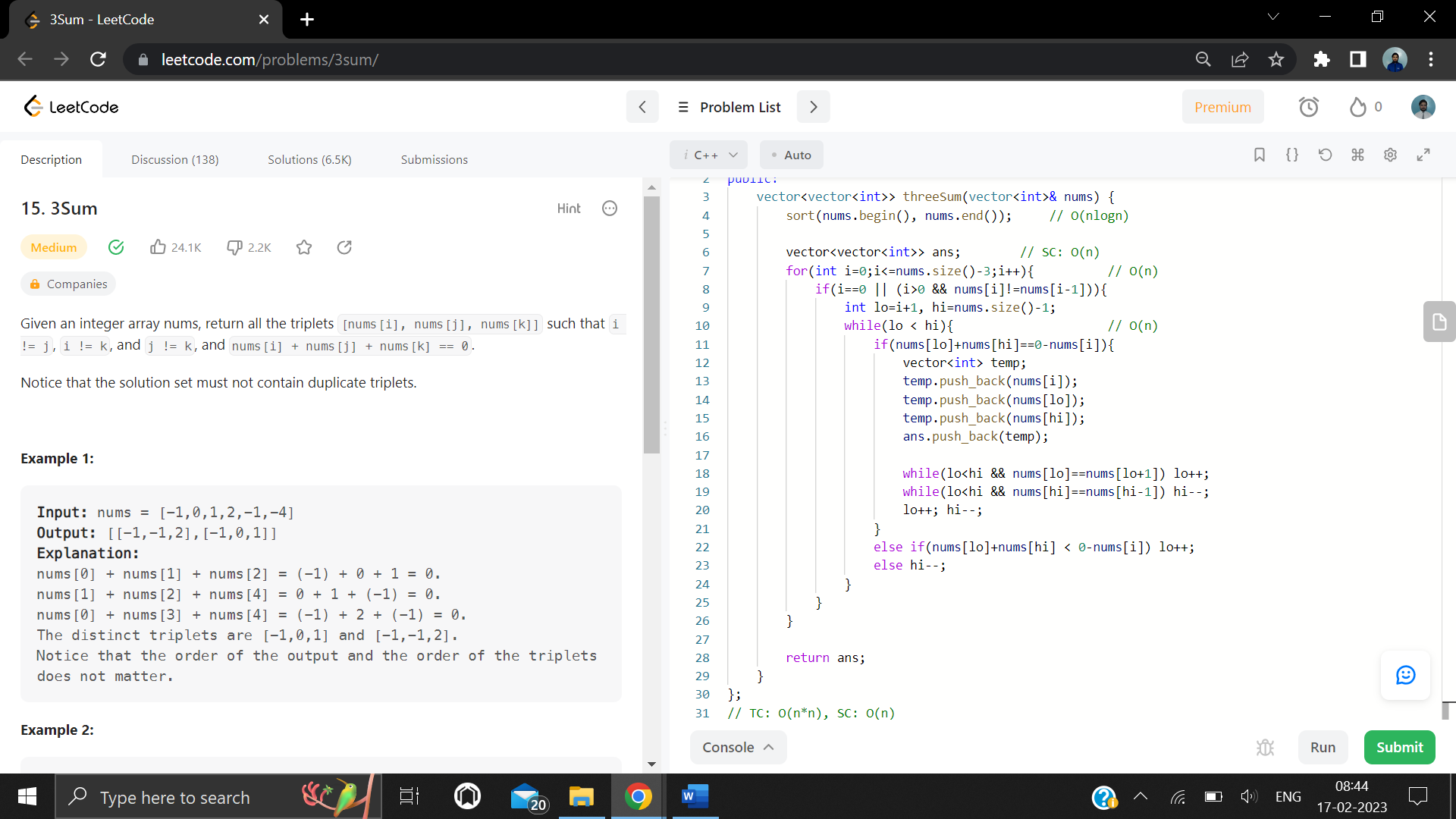
        returnans;

    }

};

// TC: O(n\*n), SC: O(n)

**Output:**



**Problem 2: “Merge Two Sorted Lists”**

<https://leetcode.com/problems/merge-two-sorted-lists/>

You are given the heads of two sorted linked lists list1 and list2.

Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

**Code:**

classSolution {

public:

    ListNode\*mergeTwoLists(ListNode\*list1, ListNode\*list2) {

        if(list1==NULL) return list2;

        if(list2==NULL) return list1;

        if(list1->val>list2->val) swap(list1,list2);

        ListNode\* ans= list1;

        while(list1!=NULL&& list2!=NULL){           // O(n1+n2)

            ListNode\* temp= NULL;

            while(list1!=NULL&&list1->val<= list2->val){

                temp=list1;

                list1= list1->next;

            }

            temp->next= list2;

            swap(list1,list2);

        }

        returnans;

    }

};

// TC: O(n1+n2), SC: O(1)

**Output:**

