“**Experiment1.4”**

**Aim:**

To demonstrate the concept of Hashing.

**Objective:**

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

• The implementation of Missing Number problem brushes up the concept of set / hash.

• The implementation of Longest Duplicate Substring problem brushes up the concept of hash.

**Problem 1: “Missing Number”**

<https://leetcode.com/problems/missing-number/>

Given an array nums containing n distinct numbers in the range [0, n], return the only number in the range that is missing from the array.

**Code:**

class Solution {

public:

    int missingNumber(vector<int>& nums) {

        unordered\_set<int> s;               // SC: O(n)

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

            s.insert(nums[i]);

        }

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

            if(s.find(i)==s.end()) return i;        // O(1) in avg case

        }

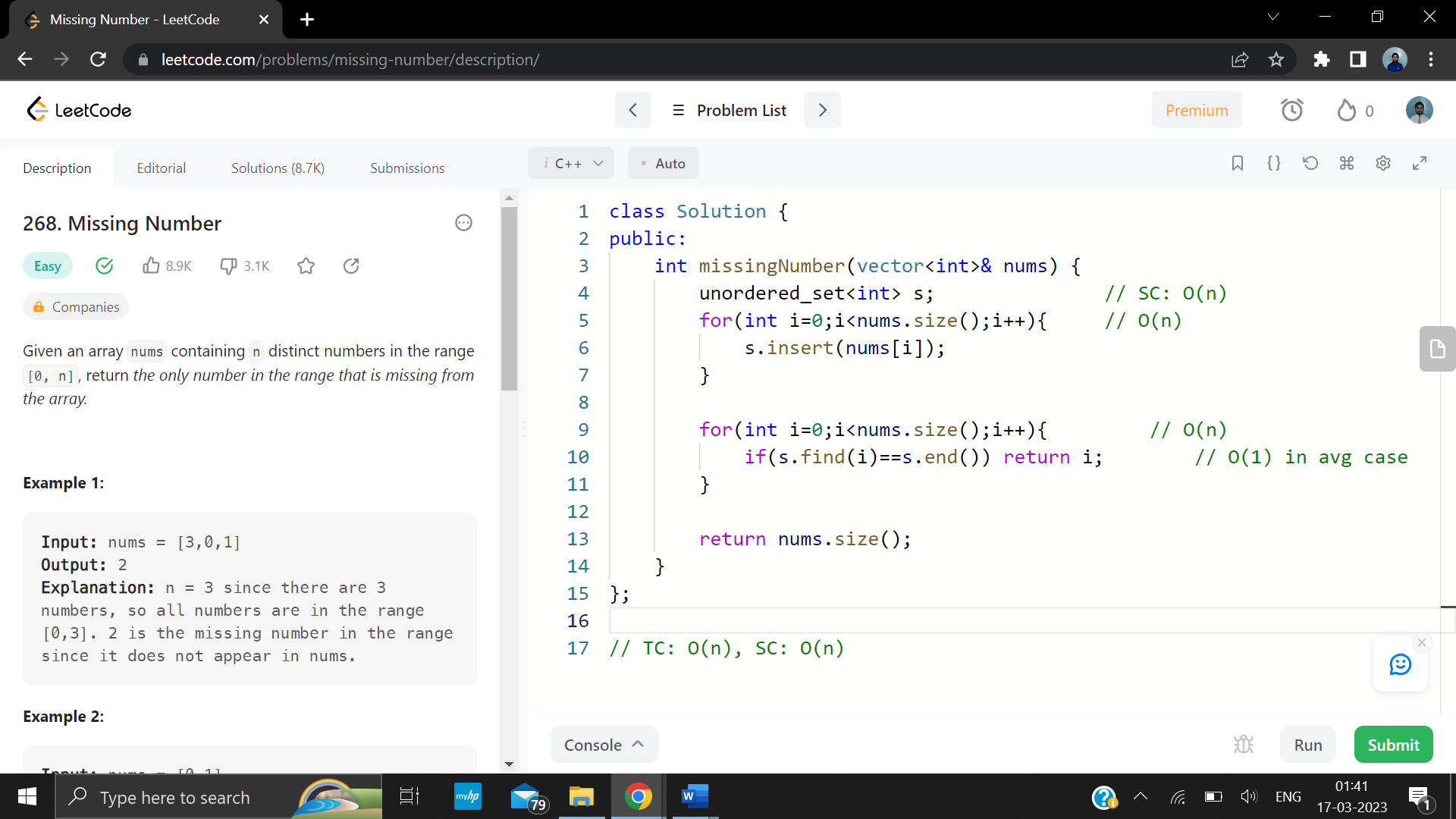
        return nums.size();

    }

};

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

**Output:**



**Problem 2: “Longest Duplicate Substring”**

<https://leetcode.com/problems/longest-duplicate-substring/>

Given a string s, consider all duplicated strings: (contiguous) substrings of s that occur 2 or more times. The occurrences may overlap.

Return any duplicated substring that has the longest possible length. If s does not have a duplicated substring, the answer is “”.

**Code:**

class Solution {

    long long base = 31, MOD=1e11+19;

public:

    string longestDupSubstring(string s) {

        int n = s.size();

        int l=0, r=n;

        int mx=0,idx=0;

        while(l<r){

            int mid = l+ceil((r-l)/(2\*1.0));

            unordered\_set<long long> currHash;

            long long hash=0, d=1;

            bool found = false;

            for(int i=0; i<n; i++){

                hash = (hash\*base + s[i]-'a'+1)%MOD;

                if(i>=mid){

                    hash = (MOD + hash - d\*(s[i-mid]-'a'+1)%MOD)%MOD;

                }else{

                    d = d\*base%MOD;

                }

                if(i>=mid-1){

                    if(currHash.count(hash)){

                        idx = i-mid+1;

                        found = true;

                        break;

                    }

                    currHash.insert(hash);

                }

            }

            if(!found)

                r = mid-1;

            else l = mid;

        }

        string ans = "";

            for(int i = idx; i<(idx+l); i++){

            ans += s[i];

        }

        return ans;

    }};

**Output:**

