Experiment1.4

Student Name: Himanshu	**UID:** 20BCS7944
Branch: BE-CSE	**Section/Group:** 905/A
Semester: 6th	**Date of Performance:** 22/03/2023
Subject Name: Competitive Coding-II	**Subject Code:** 20CSP-351

1. Aim:

To demonstrate the concept of hashing problem.

2. Objective:

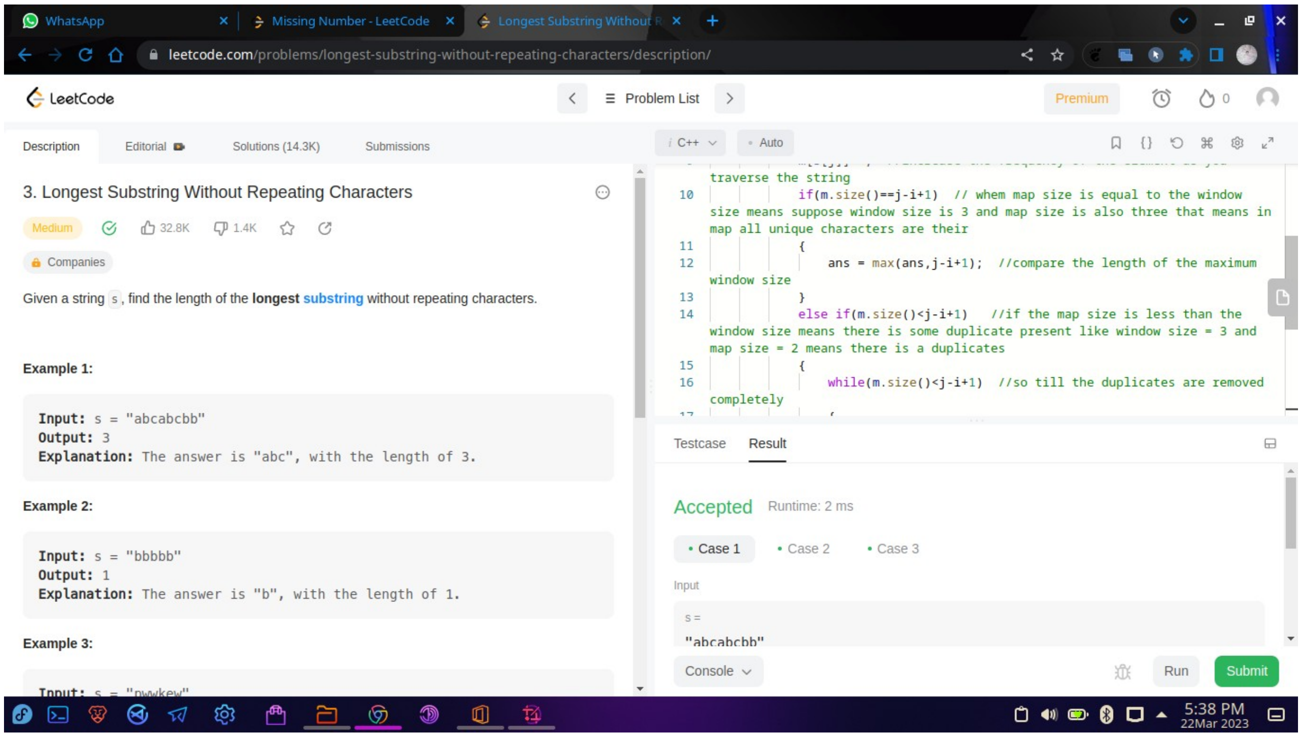
- The objective is to build problem solving capability and to learn the basic concepts of data structures.
- The implementation of missing numbers which shows and brushes up the concept of map and hashing.
- The implementation of longest substring without repeating in which the concept of hashing was introduced.

3. LeetCode code and output:

- **Longest substring withour repeating characters -**

```
class Solution {
public:
int lengthOfLongestSubstring(string s) {
        if(s.length()==0)return 0;
                unordered_map<char,int> m;

int i=0,j=0,ans=INT_MIN;
                while(j<s.length()){
                        m[s[j]]++;

                if(m.size()==j-i+1){
                        ans = max(ans,j-i+1);
                }

                else if(m.size()<j-i+1){
                        while(m.size()<j-i+1){
                                m[s[i]]--;
                if(m[s[i]]==0){
                        m.erase(s[i
            if(m[s[i]]==0){
                        m.erase(s[i])
                                    i++;
                        }
            j++;
            }
return ans;

};
```

OUTPUT:



```
traverse the string
        if(m.size()==j-i+1)  // whem map size is equal to the window size means suppose window size is 3 and map size is also three that means in map all unique characters are their
        {
            ans = max(ans,j-i+1);  //compare the length of the maximum window size
        }
        else if(m.size()<j-i+1)   //if the map size is less than the window size means there is some duplicate present like window size = 3 and map size = 2 means there is a duplicates
        {
            while(m.size()<j-i+1)  //so till the duplicates are removed completely
```

3. Longest Substring Without Repeating Characters

Medium 32.8K 1.4K

Companies

Given a string `s`, find the length of the **longest** substring without repeating characters.

Example 1:

```
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

Example 2:

```
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

Example 3:

```
Input: s = "pwwkew"
```

Accepted Runtime: 2 ms

Case 1 Case 2 Case 3

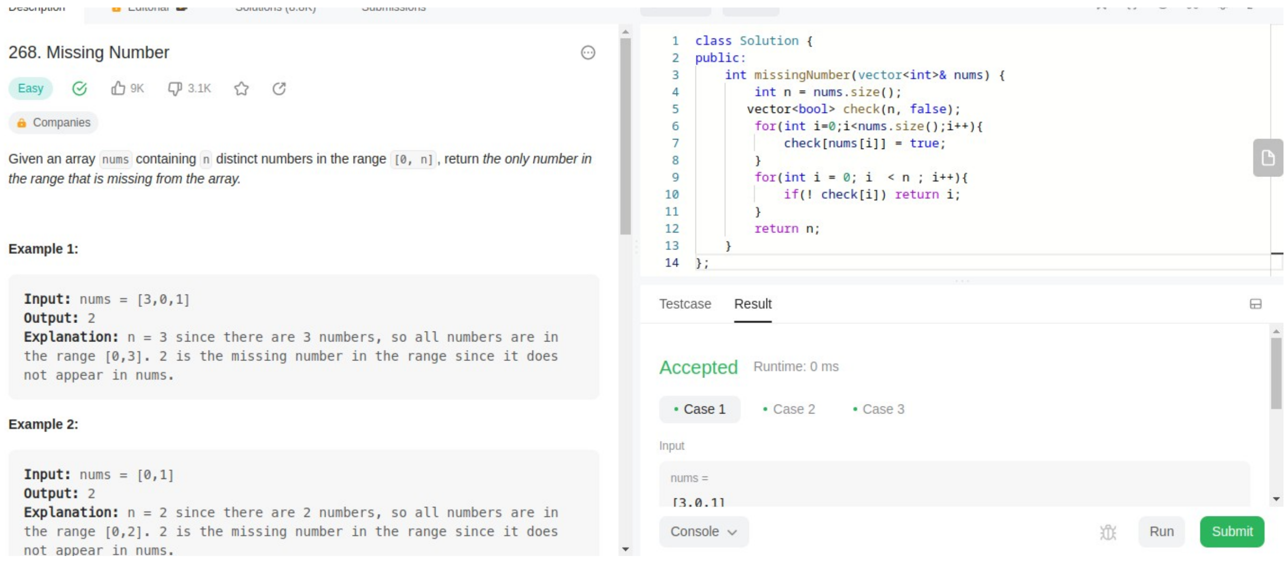
Input

```
s =
"abcabcbb"
```

- **Missing numbers -**

```
class Solution {
public:
        int missingNumber(vector<int>& nums) {
        int n = nums.size();
        vector<bool> check(n, false);
        for(int i=0;i<nums.size();i++){
                check[nums[i]] = true;
        }
for(int i = 0; i < n ; i++){
        if(! check[i]) return i;
}
        return n;
       }
};
```

OUTPUT:



268. 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.*

Example 1:

```
Input: nums = [3,0,1]
Output: 2
Explanation: n = 3 since there are 3 numbers, so all numbers are in
the range [0,3]. 2 is the missing number in the range since it does
not appear in nums.
```

Example 2:

```
Input: nums = [0,1]
Output: 2
Explanation: n = 2 since there are 2 numbers, so all numbers are in
the range [0,2]. 2 is the missing number in the range since it does
not appear in nums.
```

```
class Solution {
public:
    int missingNumber(vector<int>& nums) {
        int n = nums.size();
       vector<bool> check(n, false);
        for(int i=0;i<nums.size();i++){
            check[nums[i]] = true;
        }
        for(int i = 0; i  < n ; i++){
            if(! check[i]) return i;
        }
        return n;
    }
};
```

Accepted Runtime: 0 ms

Input

```
nums =
[3,0,1]
```