Experiment 1.4

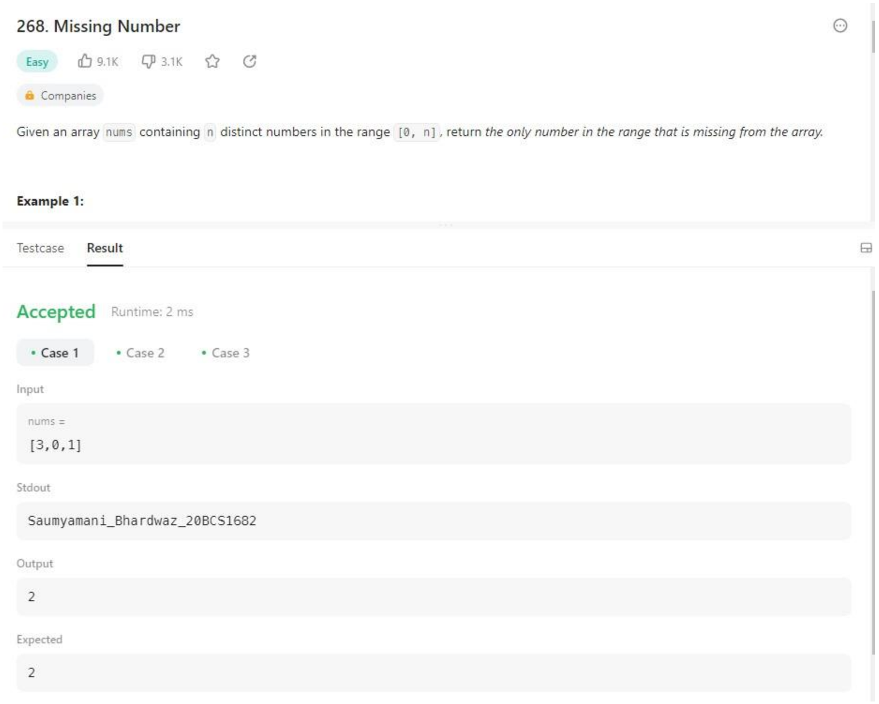
Student Name: Saumyamani Bhardwaz **UID:** 20BCS1682
Branch: BE-CSE **Section/Group:** 701/A
Semester: 6th **Date of Performance:**
Subject Name: Competitive Coding-II **Subject Code:** 20CSP-351

Aim: To demonstrate theconcept of Hashing

1. Missing Number

```
class Solution {
public:
    int missingNumber(vector<int>& nums) {
    cout<<"Saumyamani Bhardwaz_20BCS1682"<<endl;
    sort(nums.begin(),nums.end());
        for(int i=0;i<nums.size();++i)
        {
            if(nums[i]!=i) return i;
        }
        return nums.size();
    }
 };
```

Output:



268. Missing Number

Easy 9.1K 3.1K

Companies

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:

Testcase Result

Accepted Runtime: 2 ms

Case 1 Case 2 Case 3

Input

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

Stdout

```
Saumyamani_Bhardwaz_20BCS1682
```

Output

```
2
```

Expected

```
2
```

2. Word Pattern

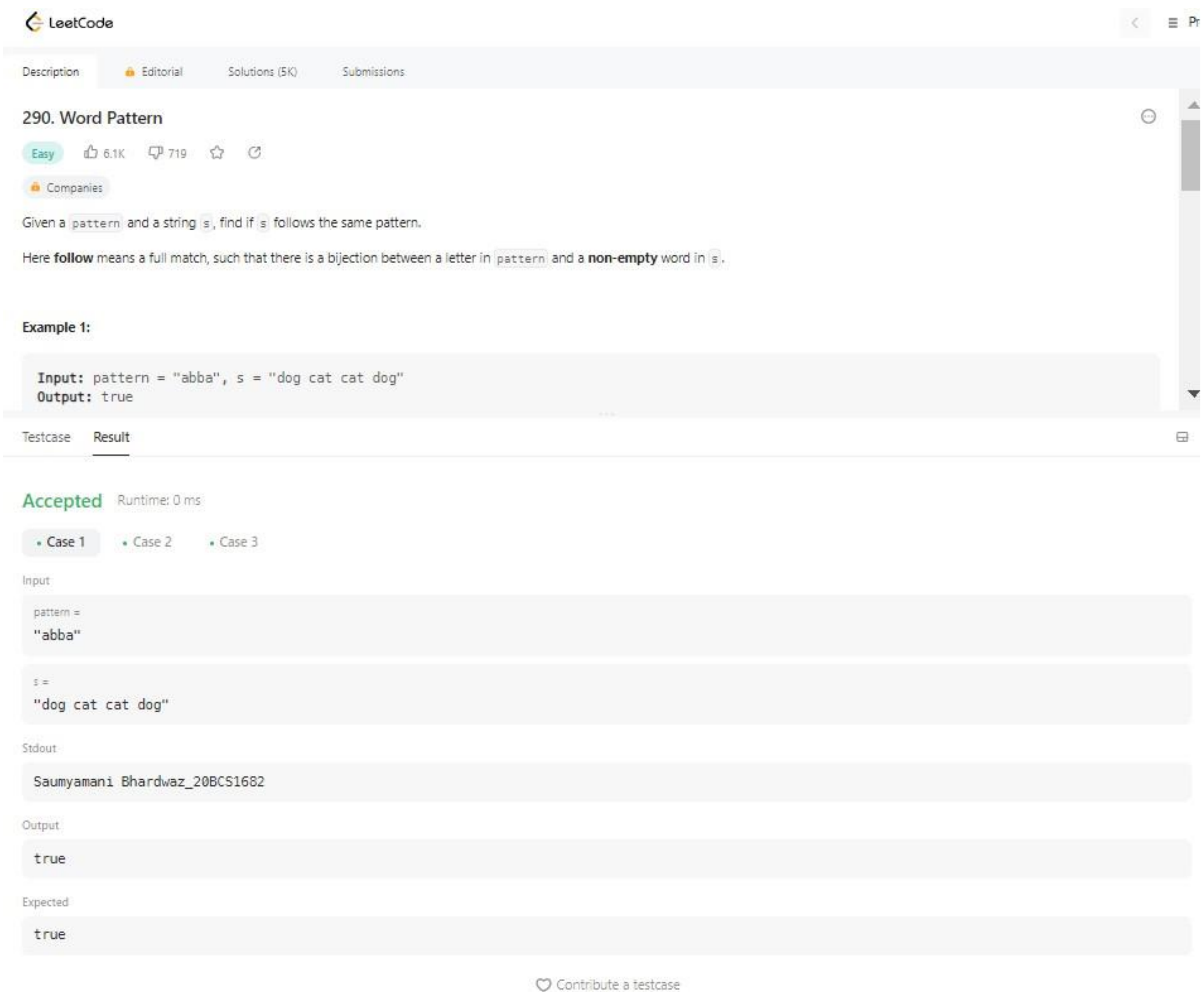
```
class Solution {
public:
    bool wordPattern(string pattern, string s) {
        map<char,string> cnt;
        vector<string> words;
        string t="";
        for(int i=0; i<=s.size(); i++) {
            if(s[i] == '\0') {
                words.push_back(t);
                break;
            }
            else if(s[i] == ' ') {
                words.push_back(t);
                t="";
            }
            else t+=s[i];
        }

        if(words.size() != pattern.size()) return 0;

        map<string, int> vis;
        for(int i=0; i<pattern.size(); i++) {
            if(cnt.find(pattern[i]) != cnt.end()) {
                if(cnt[pattern[i]] != words[i]) return 0;

            }
            else {
                cnt[pattern[i]] = words[i];
                if(vis[words[i]]) return 0;
                vis[words[i]] = 1;
            }
        }
      cout<<"Saumyamani Bhardwaz_20BCS1682"<<endl;
        return 1;
    }
}
```

Output:


LeetCode

Description | Editorial | Solutions (5K) | Submissions

290. Word Pattern

Easy 6.1K 719

Companies

Given a `pattern` and a string `s`, find if `s` follows the same pattern.

Here **follow** means a full match, such that there is a bijection between a letter in `pattern` and a **non-empty** word in `s`.

Example 1:

```
Input: pattern = "abba", s = "dog cat cat dog"
Output: true
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

pattern =
```
"abba"
```

s =
```
"dog cat cat dog"
```

Stdout
```
Saumyamani Bhardwaz_20BCS1682
```

Output
```
true
```

Expected
```
true
```

Contribute a testcase

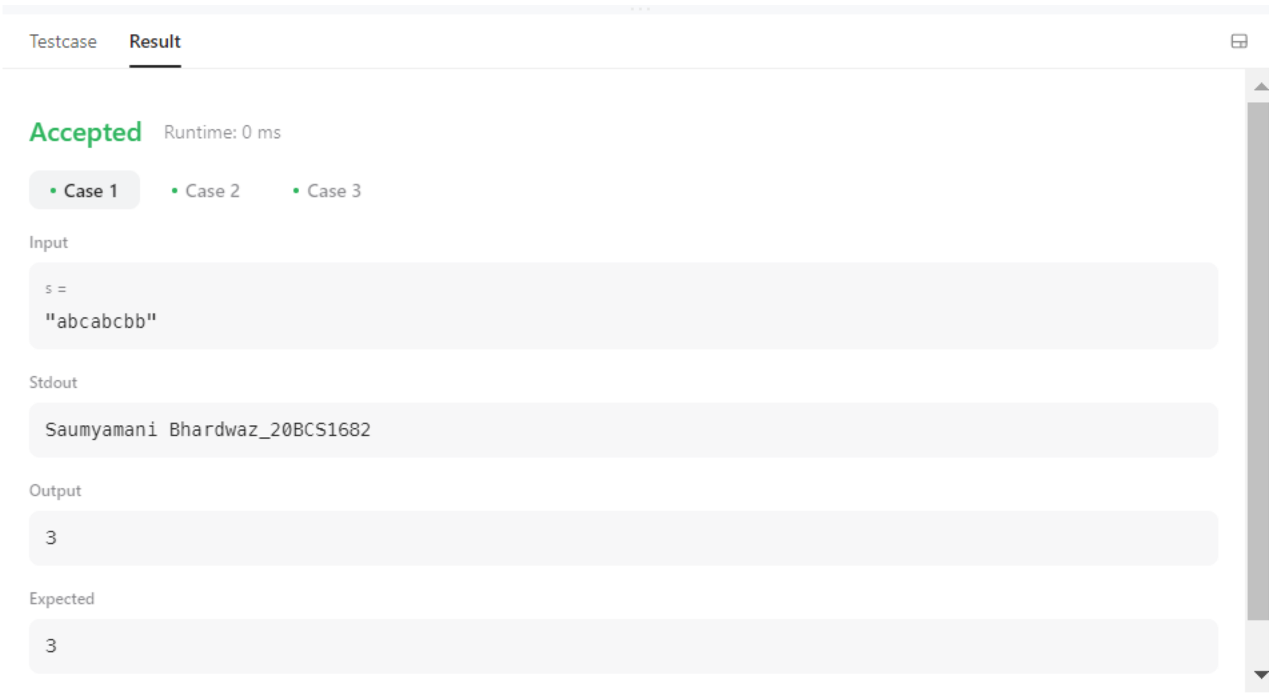
3. Longest Substring Without Repeating Characters

```
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]);
                }
                i++;
              }
            }
            j++;
          }

          return ans;
        }
    };
```

Output:



Testcase Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

```
s =
"abcabcbb"
```

Stdout

```
Saumyamani Bhardwaz_20BCS1682
```

Output

```
3
```

Expected

```
3
```