



## EXPERIMENT - 4

**Student Name:** Sandeep Kumar

**UID:** 20BCS4885

**Branch:** CSE

**Section/Group:** 603/A

**Semester:** 6<sup>th</sup> semester

**Subject:** Competitive Coding

**Aim:** To demonstrate the concept of Hashing

**Objective:**

- a) Missing number
- b) Longest Duplicate Substring

**Problem 1: Missing number**

**Solution code:**

```
class Solution {
public:
    int missingNumber(vector<int>& nums) {
        int ans=0;
        for(int i=0;i<nums.size();i++)
        {
            ans^=nums[i];
            ans^=i+1;
        }
        return ans;
    }
};
```

**Explanation of code:**

- The given code defines a class Solution with a public method missingNumber which takes a vector of integers nums as input and returns an integer. The method implements a solution for the "missing number" problem which involves finding the missing number in an array of integers from 0 to n, where n is the length of the array.
- The method initializes an integer variable ans to 0. It then uses a loop to iterate over all the elements in the input array nums. For each element nums[i], it XORs it with ans to toggle the bits that are different between the two numbers. It also XORs ans with i+1 to toggle the bits corresponding to the missing number. This effectively XORs all the



integers from 1 to n with nums[0] to nums[n-1], and cancels out all the XORed pairs except for the missing number and the XORed result of all the integers from 1 to n.

- Finally, the method returns the value of ans, which represents the missing number.

Overall, this code appears to be a correct and efficient solution to the "missing number" problem using bitwise XOR operation.

## Output:

The screenshot shows a coding platform interface with a dark theme. At the top, there are tabs for 'Testcase' and 'Result', with 'Result' being the active tab. Below the tabs, the status 'Accepted' is displayed in green, followed by 'Runtime: 0 ms'. There are three tabs for test cases: 'Case 1' (selected), 'Case 2', and 'Case 3'. Under 'Case 1', the 'Input' section shows 'nums = [3, 0, 1]'. The 'Output' section shows '2'. The 'Expected' section also shows '2'. At the bottom of the interface, there is a 'Console' dropdown, a 'Run' button, and a green 'Submit' button. A link to 'Contribute a testcase' is also visible.

## Problem 2: Longest Duplicate Substring

### Input Code:

```
class Solution {
public:
    string longestDupSubstring(string s) {

        // map to store the indices of each character
        unordered_map<char, vector<int>>> hash;
        int n = s.length();

        for(int i=0; i<n; i++)
            hash[s[i]].push_back(i);

        // Max stores the length of the longest duplicate substring found till now
        // index stores the starting position of the substring
```



```
int Max = 0, index = -1;
```

```
for(int i=0; i<n; i++)
```

```
{
```

```
    // Here f stores the current character
```

```
    char f = s[i];
```

```
    // erasing the current index
```

```
    hash[f].erase(hash[f].begin());
```

```
    // after erasing, checking the longest substring possible starting with the
```

```
current character
```

```
    for(int it : hash[f])
```

```
    {
```

```
        // j stores the max length of the duplicate substring found till
```

```
        int j = 0;
```

```
        // incrementing j till index not out of bound and both characters are
```

```
duplicate
```

```
        while(i+j < n and it+j < n and s[i+j] == s[it+j])
```

```
            j++;
```

```
        if(j > Max)
```

```
        {
```

```
            Max = j;
```

```
            index = i;
```

```
        }
```

```
        // this is required when we achieved the longest substring possible
```

```
        if(Max == n-i-1)
```

```
            return s.substr(index, Max);
```

```
    }
```

```
}
```

```
if(Max == 0)
```

```
    return "";
```

```
else
```

```
    return s.substr(index, Max);
```

```
}
```

```
};
```



## Explanation of Code:

- The given code defines a class Solution with a public method longestDupSubstring which takes a string s as input and returns a string. The method implements a solution to find the longest duplicate substring in the input string s.
- The method uses an unordered map hash to store the indices of each character in the input string s. It then initializes two integer variables Max and index to 0 and -1 respectively. These variables will be used to store the length of the longest duplicate substring found till now and its starting index.
- The method then loops through each character of the input string s. For each character f, it erases the current index from the hash map hash[f]. It then checks all the remaining indices for the current character f in the hash map hash[f] and finds the length of the longest duplicate substring possible starting with the current character f. It does this by incrementing an integer variable j until either the index is out of bounds or the characters at the current position are no longer duplicate.
- If the length j of the longest duplicate substring found for the current character f is greater than the value stored in Max, then the values of Max and index are updated to store the new maximum length and starting index respectively.
- Finally, the method returns the longest duplicate substring found in the input string s by using the substr function with the starting index and the length of the substring stored in Max.

Overall, this code appears to be a correct and efficient solution to finding the longest duplicate substring in the input string using an unordered map to store the indices of each character.

## Output:

The screenshot shows a code execution environment with a dark theme. At the top, there are tabs for 'Testcase' and 'Result', with 'Result' being the active tab. Below the tabs, the status 'Accepted' is displayed in green, followed by 'Runtime: 6 ms'. There are two tabs for test cases, 'Case 1' and 'Case 2', with 'Case 1' being selected. Under the 'Input' section, the variable 's =' is followed by a text box containing '"banana"'. Under the 'Output' section, a text box contains '"ana"'. Under the 'Expected' section, a text box also contains '"ana"'. At the bottom of the interface, there is a 'Console' dropdown menu, a 'Contribute a testcase' link, and two buttons: 'Run' and 'Submit'.