



## Experiment-4

**Name:** Gurpreet Yadav

**UID:** 22BET10336

**Branch:** BE-IT

**Section/Group:** 22BET\_IOT-702/A

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

**Date of Performance:** 14/02/25

**Subject Name:** AP LAB-II

**Subject Code:** 22ITP-351

## Problem-1

### 1.Aim:

Given a string s, return the longest substring of s that is nice. If there are multiple, return the substring of the earliest occurrence. If there are none, return an empty string.

### 2.Objective:

- To implement a string s for every letter of the alphabet that s contains, it appears both in uppercase and lowercase.
- To return an empty string.

### 3.Code:

```
class Solution
{
public:
    string longestNiceSubstring(string s) {
        if (s.size() < 2) return "";
        unordered_set<char> st(begin(s), end(s));
        for (int i = 0; i < s.size(); i++) {
            if (st.find((char) toupper(s[i])) == end(st) || st.find((char) tolower(s[i])) == end(st)) {
                string s1 = longestNiceSubstring(s.substr(0, i));
                string s2 = longestNiceSubstring(s.substr(i + 1));
                return s1.size() >= s2.size() ? s1 : s2;
            }
        }
    }
}
```

```
}  
  
return s;  
  
}  
  
};
```

## 4.Output:

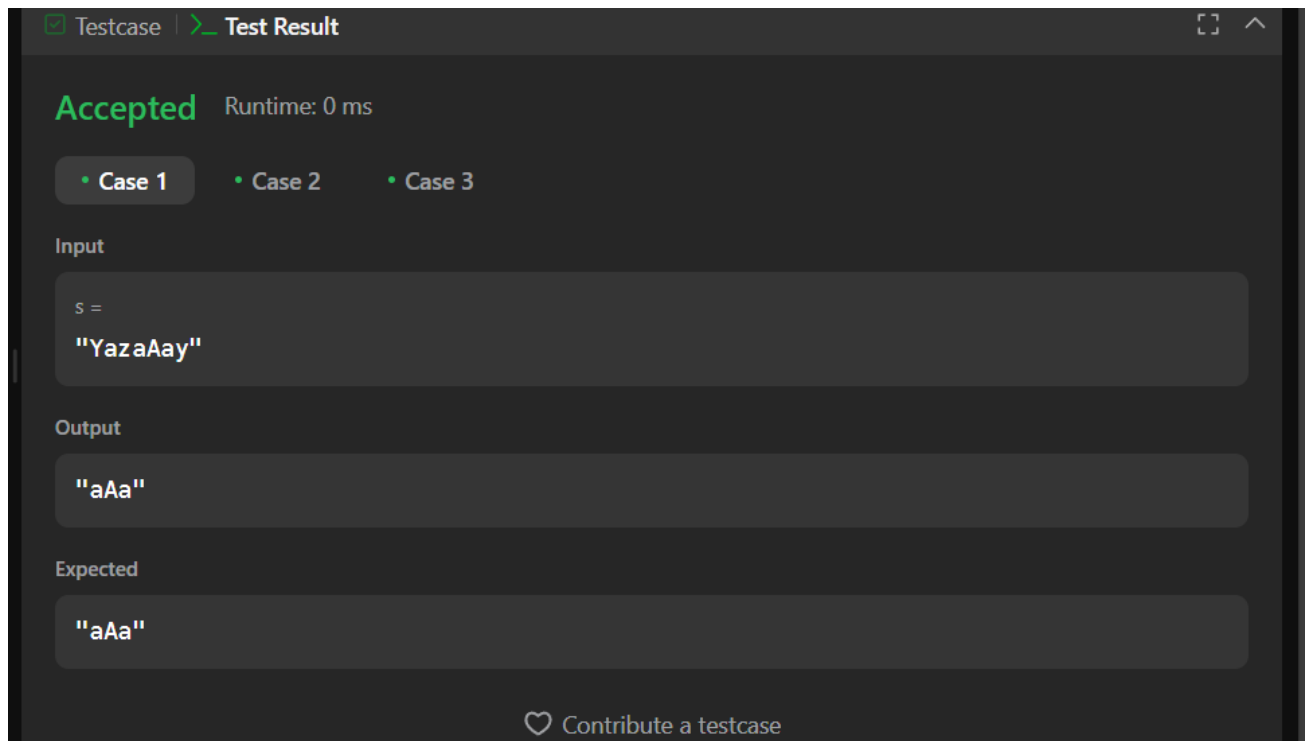


Fig.1.Longest Nice Substring



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Problem-2

### 1.Aim:

Reverse bits of a given 32 bits unsigned integer and they should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.

### 2.Objective:

- There is no unsigned integer type.
- Return the reversed list.

### 3.Code:

```
class Solution {  
  
public:  
  
    uint32_t reverseBits(uint32_t n) {  
  
        uint32_t result= 0;  
  
        for(int i=0; i<32; i++)  
  
            result = (result<<1) + (n>>i &1);  
  
        return result;  
    }  
};
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 4.Output:

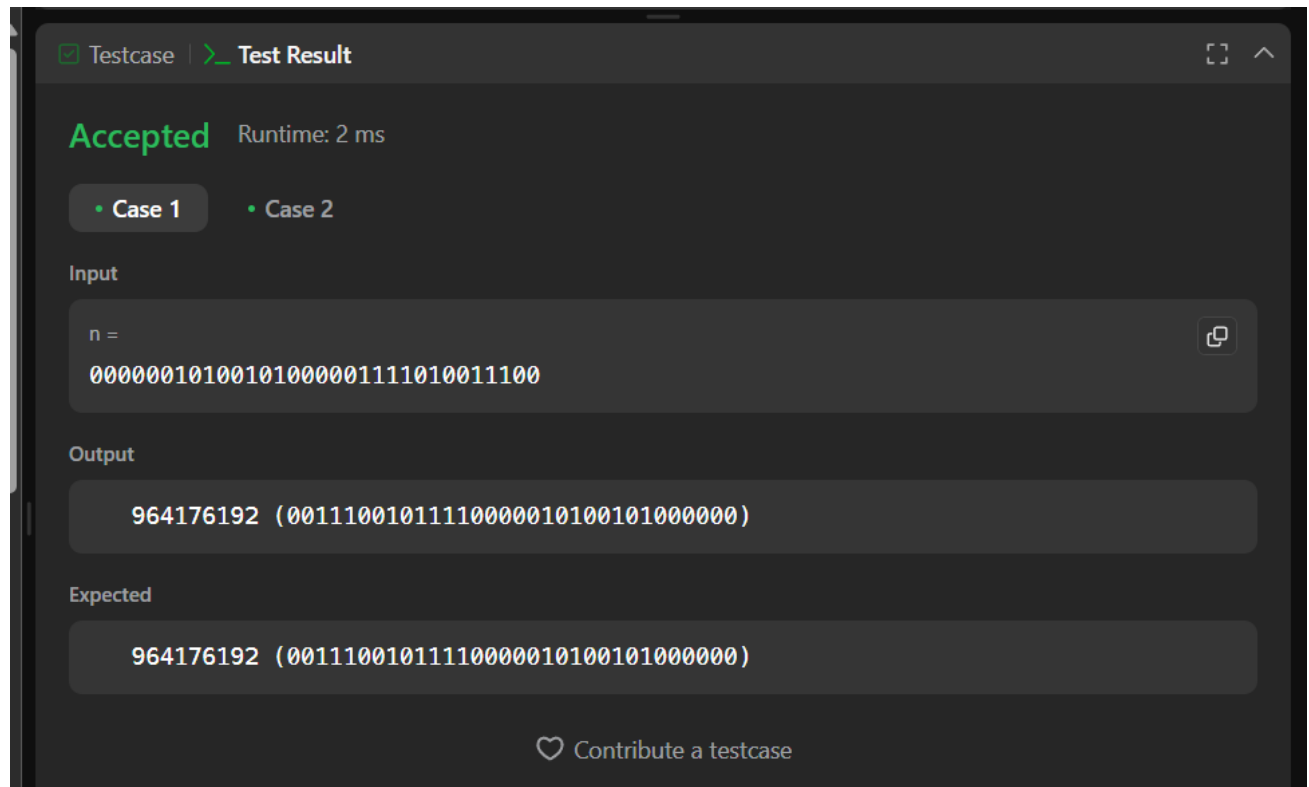


Fig.2:Reverse Bits



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Problem-3

### 1.Aim:

Given a positive integer n, write a function that returns the number of set bits in its binary representation also known as the Hamming Weight.

### 2.Objective:

- To implement a function to print number of 1 bits.
- To return the number of set bits in its binary representation .

### 3.Code:

```
class Solution {  
  
public:  
  
    int hammingWeight(int n) {  
  
        int count = 0;  
  
        while (n ) {  
  
            n = n & (n - 1);  
  
  
            count++;  
  
        }  
  
        return count;  
  
    }  
  
};
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 4.Output:

[Testcase](#) | [Test Result](#)

**Accepted** Runtime: 0 ms

[• Case 1](#) • Case 2 • Case 3

Input

n =  
11

Output

3

Expected

3

[Contribute a testcase](#)

Fig.3: Number of 1 bits



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Problem-4

**1.Aim:** Given an integer array nums, find the subarray with the largest sum, and return *its sum*.

### **2.Objective:**

- To find the subarray with the largest sum
- To return the sum of integer array nums.

### **3.Code:**

```
class Solution {  
  
public:  
  
    int maxSubArray(vector<int>& nums) {  
  
        int maxSum = INT_MIN;  
  
        int currentSum = 0;  
  
        for (int i = 0; i < nums.size(); i++) {  
  
            currentSum += nums[i];  
  
  
            if (currentSum > maxSum) {  
  
                maxSum = currentSum;  
  
            }  
  
  
            if (currentSum < 0) {  
  
                currentSum = 0;  
  
            }  
  
        }  
  
        return maxSum;  
  
    }  
  
};
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 4.Output:

A screenshot of a web-based testing interface for a 'MaxSubarray' problem. The interface has a dark theme. At the top, there are two tabs: 'Testcase' (with a green checkmark) and 'Test Result' (with a green arrow). Below the tabs, the word 'Accepted' is displayed in green, followed by 'Runtime: 0 ms'. There are three buttons labeled 'Case 1', 'Case 2', and 'Case 3', with 'Case 1' being the active one. Under the 'Input' section, the text 'nums =' is followed by an array: '[-2,1,-3,4,-1,2,1,-5,4]'. Under the 'Output' section, the number '6' is displayed. Under the 'Expected' section, the number '6' is also displayed, indicating a successful test case.

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =  
[-2,1,-3,4,-1,2,1,-5,4]

Output

6

Expected

6

Fig.4:MaxSubarray



## Problem-5

**1.Aim:** Write an efficient algorithm that searches for a value target in an  $m \times n$  integer matrix matrix..

## **2.Objective:**

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

## **3.Code:**

```
class Solution {  
  
public:  
  
    bool searchMatrix(vector<vector<int>>& matrix, int target) {  
  
        int m = matrix.size(), n = m ? matrix[0].size() : 0, r = 0, c = n - 1;  
  
        while (r < m && c >= 0) {  
  
            if (matrix[r][c] == target) {  
  
                return true;  
  
            }  
  
            matrix[r][c] > target ? c-- : r++;  
  
        }  
  
        return false;  
  
    }  
  
};
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 4.Output:

**Accepted** Runtime: 0 ms

• Case 1 • Case 2

Input

matrix =  
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]

target =  
5

Output

true

Expected

true

Fig.5:Search 2d matrix