Experiment-4

Name: Gurpreet Yadav **UID:** 22BET10336

Branch: BE-IT **Section/Group:** 22BET_IOT-702/A
Semester: 6th **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:

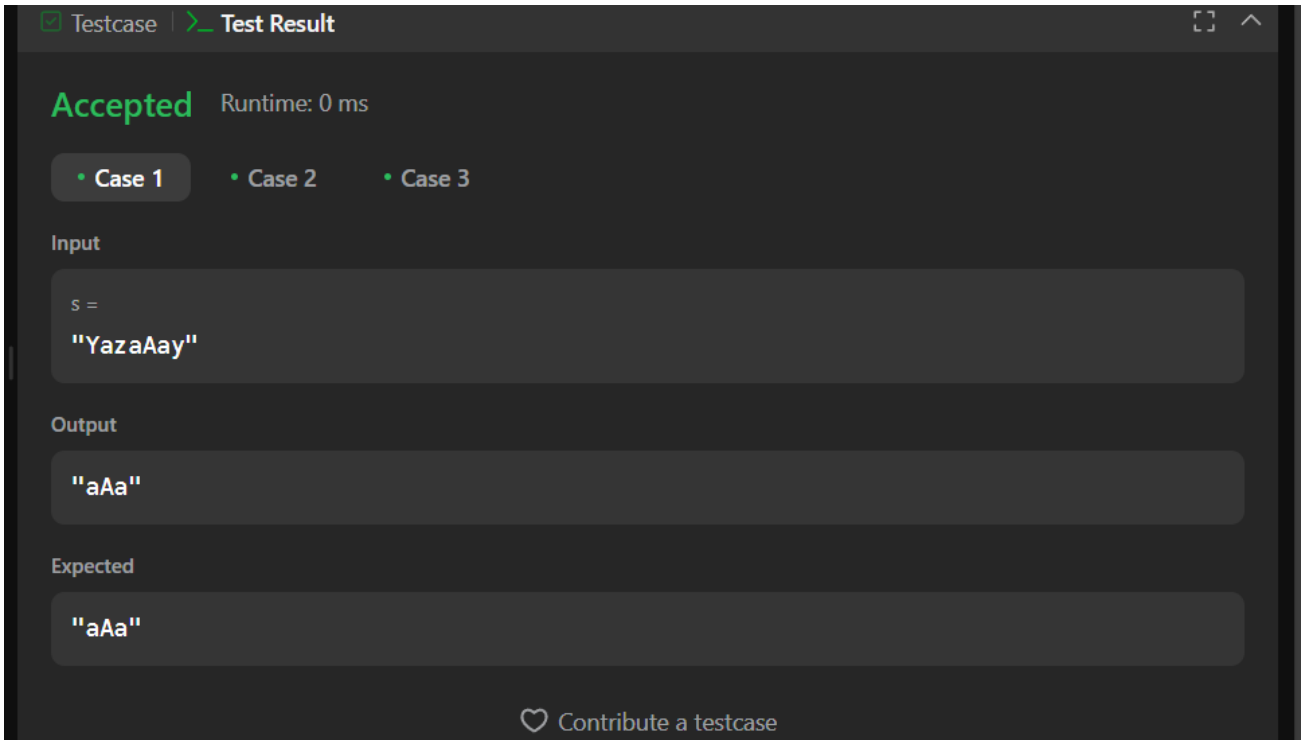

Fig.1.Longest Nice Substring

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;

   }

};
```

4.Output:

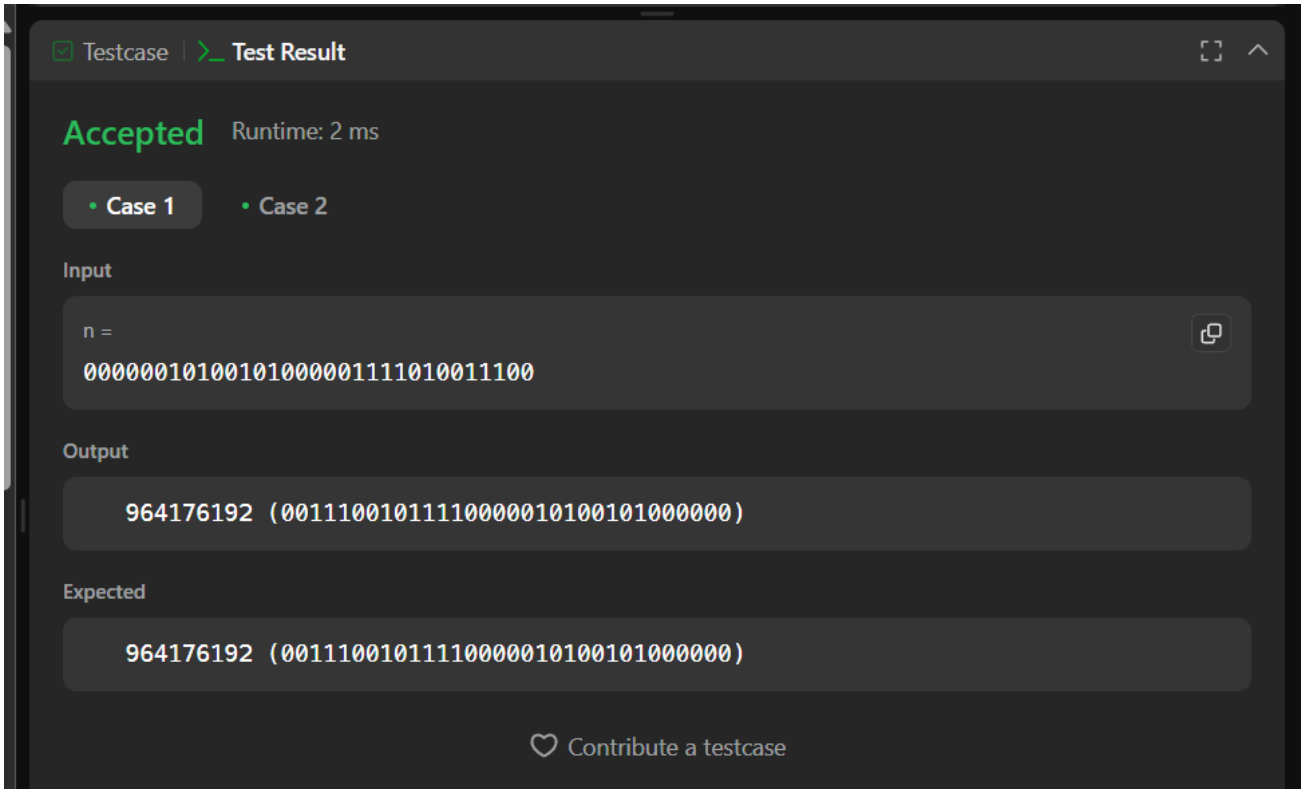

Fig.2:Reverse Bits

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;


 }

};
```

4.Output:

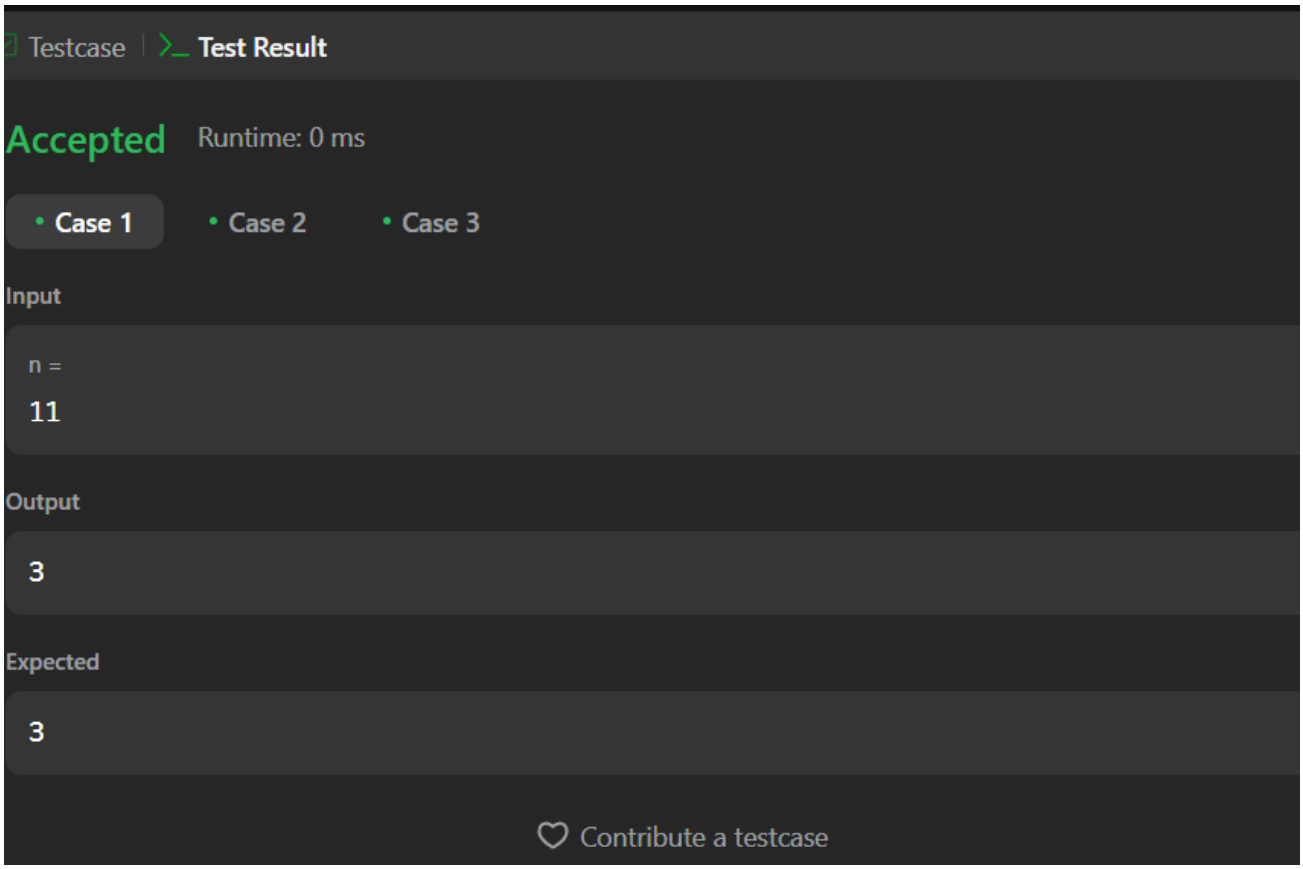

Fig.3:Number of 1 bits

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;
}
 };
```

4.Output:

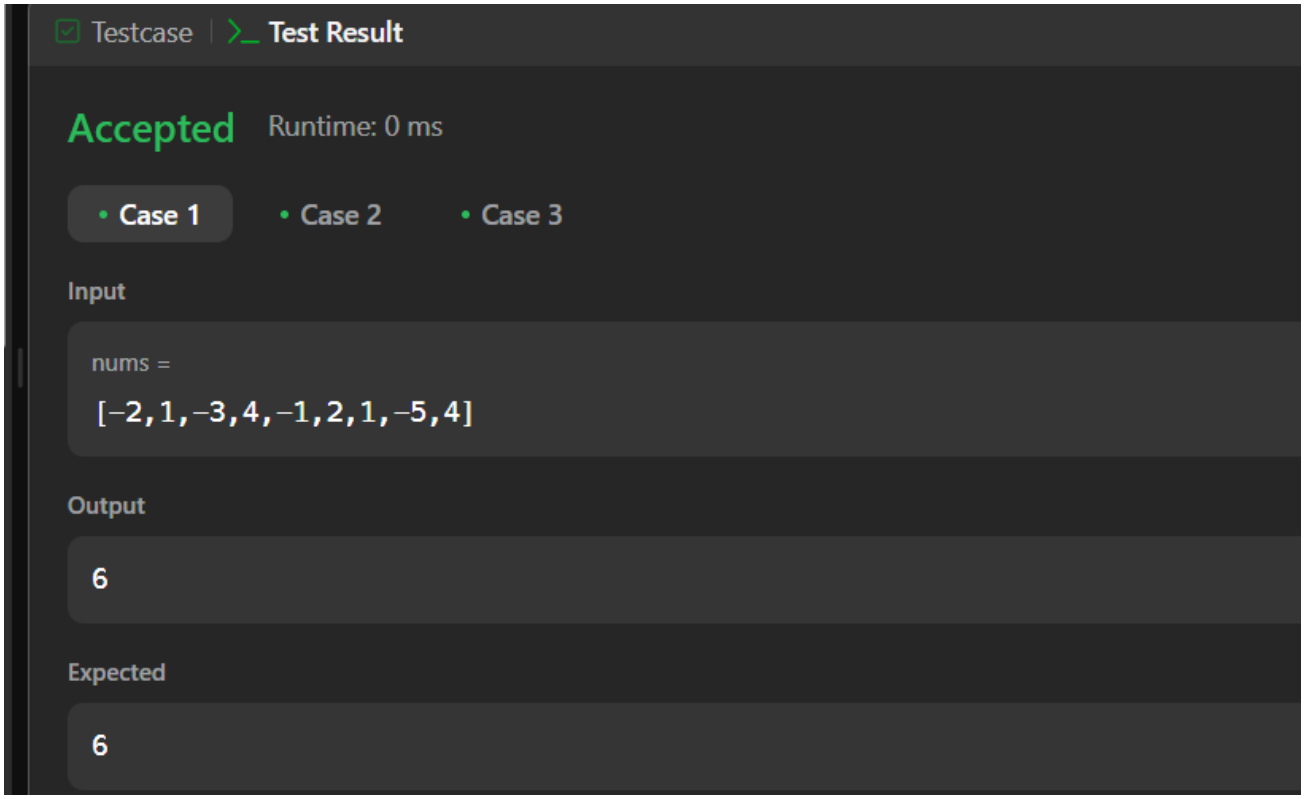

Fig.4:MaxSubarray

Problem-5

1.Aim: Write an efficient algorithm that searches for a value target in an m x 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;
}
};
```

4.Output:

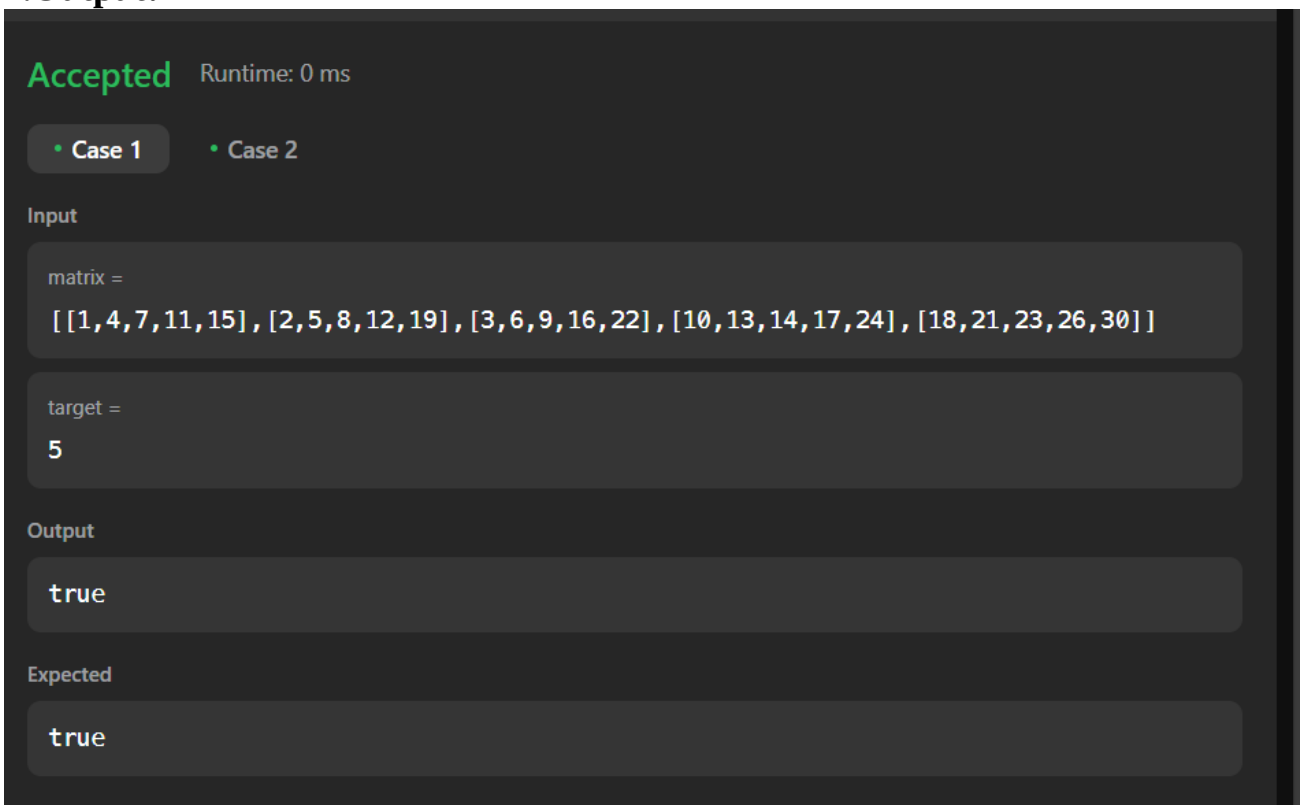

Fig.5:Search 2d matrix