Name: Devesh

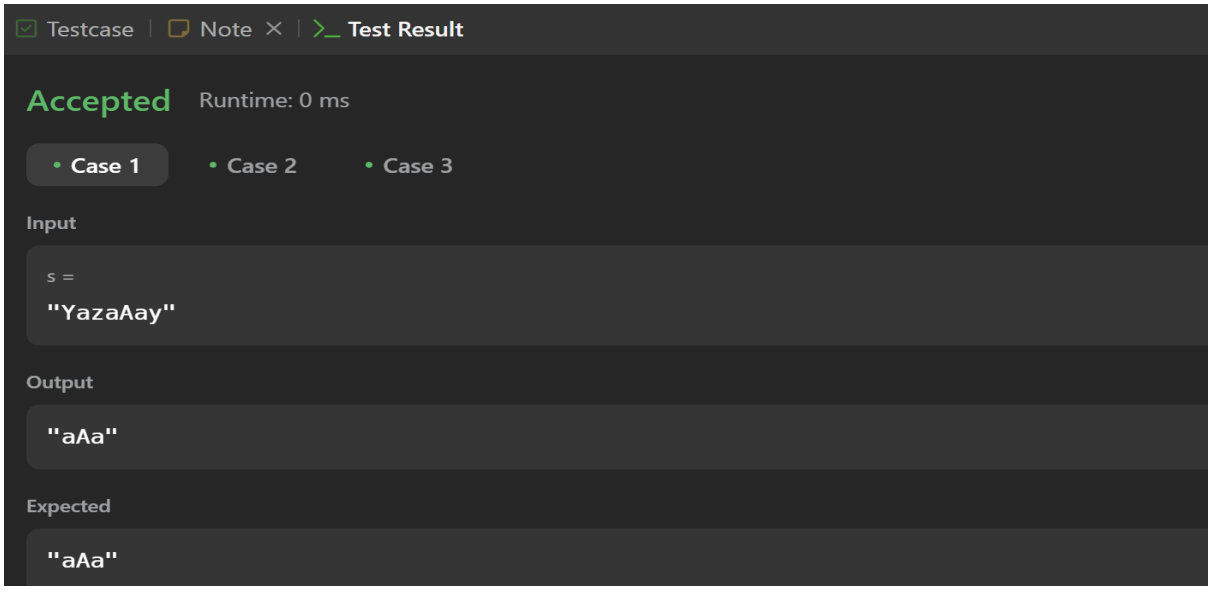
UID: 22BCS16690

Class – 605 -B

Q 1 Longest Nice Substring

```
class Solution {
    public String longestNiceSubstring(String s) {
        Set<Character> charSet = new HashSet<>();
        for (int i = 0; i < s.length(); i++) {
            charSet.add(s.charAt(i));
        }
        for (int i = 0; i < s.length(); i++) {
            if (charSet.contains(Character.toUpperCase(s.charAt(i))) &&
                    charSet.contains(Character.toLowerCase(s.charAt(i)))) {
                continue;
            }
            String s1 = longestNiceSubstring(s.substring(0, i));
            String s2 = longestNiceSubstring(s.substring(i+1));
            return s1.length()>= s2.length() ? s1 : s2;
        }
        return s; } }
```

OUTPUT:



Q 2 Reverse Bits

```
public class Solution {
    public int reverseBits(int n) {
        int result = 0;
        for(int i = 0; i<32; i++){
            int lsb = n & 1;
            int reverse = lsb <<(31-i);
             result = result | reverse;
             n = n >> 1;
        }
        return result;
    }
}
```

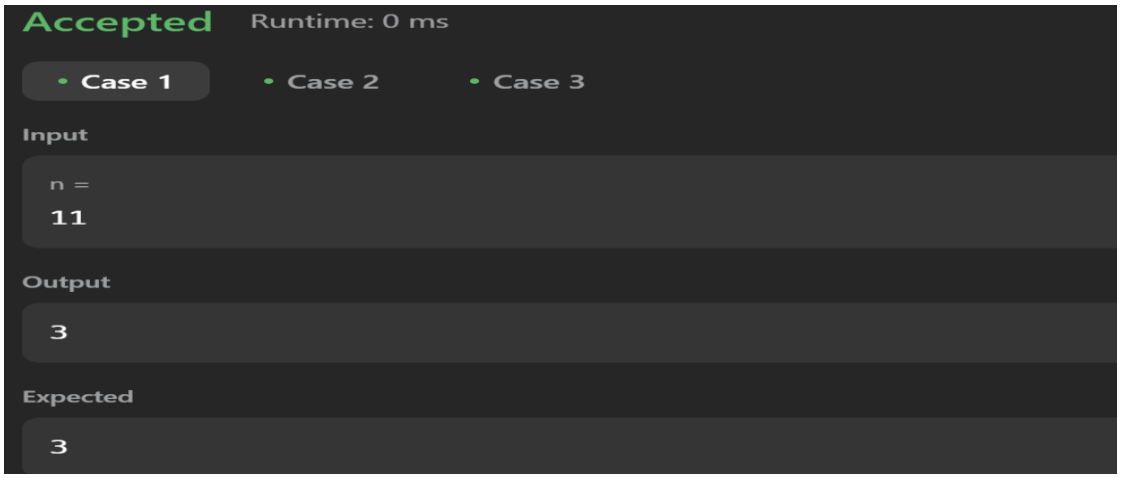
OUTPUT:

Q3 Number of 1 Bits

```
class Solution {
    public int hammingWeight(int n) {
        int count = 0;
        while(n>0){
            if((n & 1) != 0){
                count++;
            }
```

```
            n = n>>1;
        }
        return count;
    }
}
```

OUTPUT:

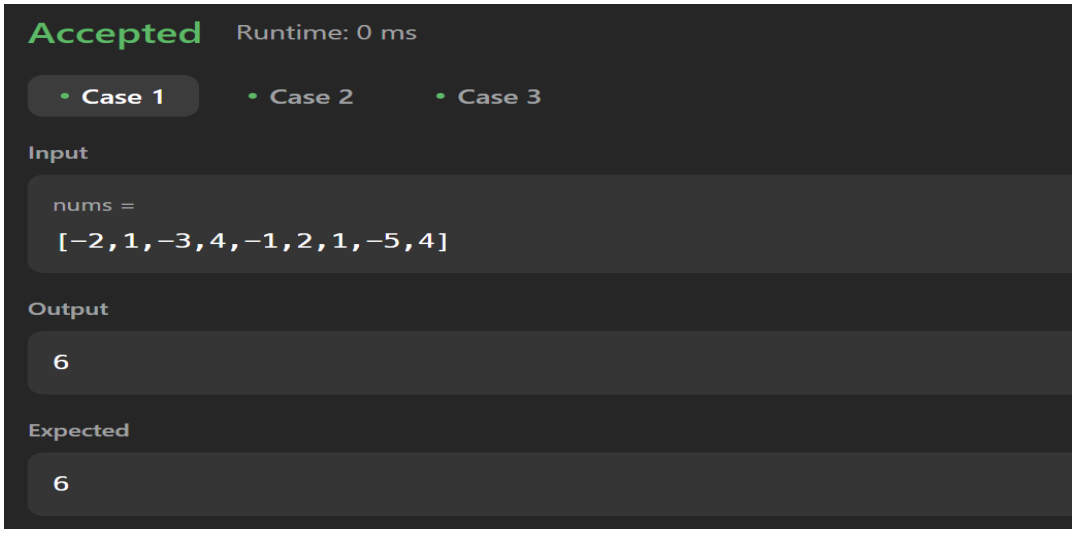


Q 4 Maximum Subarray

```
class Solution {
    public int maxSubArray(int[] nums) {
        int maxSum = Integer.MIN_VALUE;
        int currentSum = 0;

        for (int i = 0; i < nums.length; i++) {
            currentSum += nums[i];

            if (currentSum > maxSum) {
                maxSum = currentSum;
            }

            if (currentSum < 0) {
                currentSum = 0;
            }
        }
```

```
        return maxSum;
    }
}
```

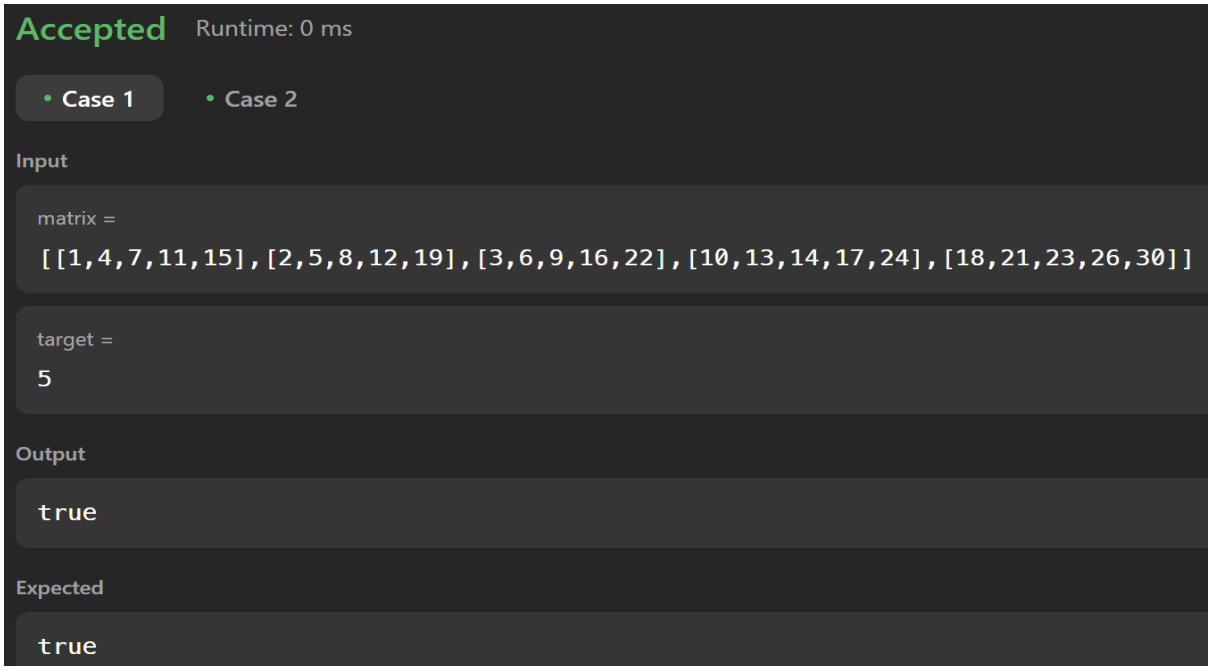
OUTPUT:



Q 5 Search a 2D Matrix II

```
public class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        int r = rows - 1;
        int c = 0;
        while (r >= 0 && c < cols) {
            if (matrix[r][c] > target) {
                r --;
            } else if (matrix[r][c] < target) {
                c ++;
            } else {
                return true;
            }
        }
        return false;
    }
}
```

OUTPUT:



Q 6 Super Pow

```
class Solution {
    public int superPow(int a, int[] b) {
        int num=0;
        for(int i:b){
            num=(num*10+i)%1140;
        }
        return binexpo(a,num,1337);
    }
    public int binexpo(int a, int b, int m){
        a%=m;
        int res=1;
        while(b>0){
            if((b&1)==1)
                res=(res*a)%m;
            a=(a*a)%m;
            b>>=1;
        }
        return res;
```

```
    }
}
```

OUTPUT:


Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
a =
2
b =
[3]
Output
8
Expected
8