1763. Longest Nice Substring

Code:

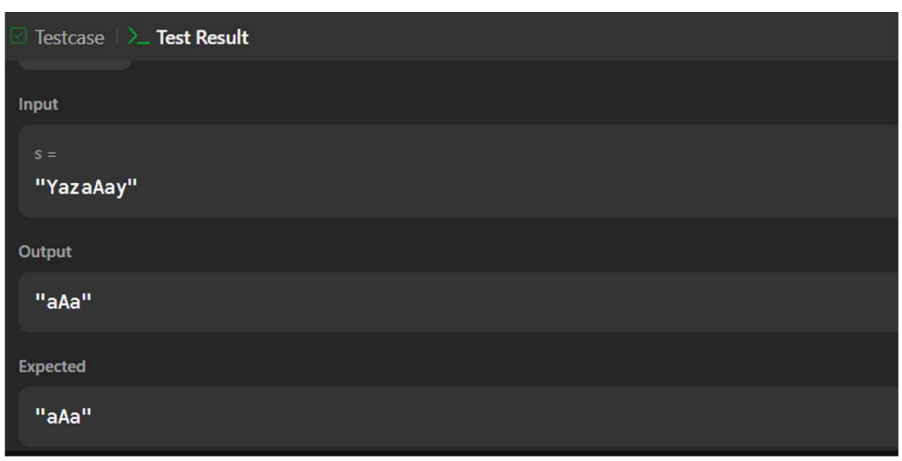
```
class Solution {
public:
   string longestNiceSubstring(string s) {


       string output = "";
      int count = 0;
      for(int i = 0;i<s.length();i++){
         int smallMask=0;
         int largeMask = 0;
         char ch = s[i];
         int chint = 0;
         if(ch>=65 && ch<=90){
            chint = ch-'A';
            largeMask = 1<<chint;
         }
         else{
            chint = ch-'a';
            smallMask = 1<<chint;
         }
         for(int j = i+1;j<s.length();j++){
            ch = s[j];
            if(ch>=65 && ch<=90){
               chint = ch-'A';
               largeMask |= 1<<chint;
            }
            else{
               chint = ch-'a';
               smallMask |= 1<<chint;
            }
```

```
            //checking for nice
            if((smallMask^largeMask) == 0){
                if(count<j-i+1){
                    count = j-i+1;
                    string temp(s.begin()+i,s.begin()+j+1);
                    output = temp;
                }
            }
        }
    }
    return output;

  }
};
```

Output:

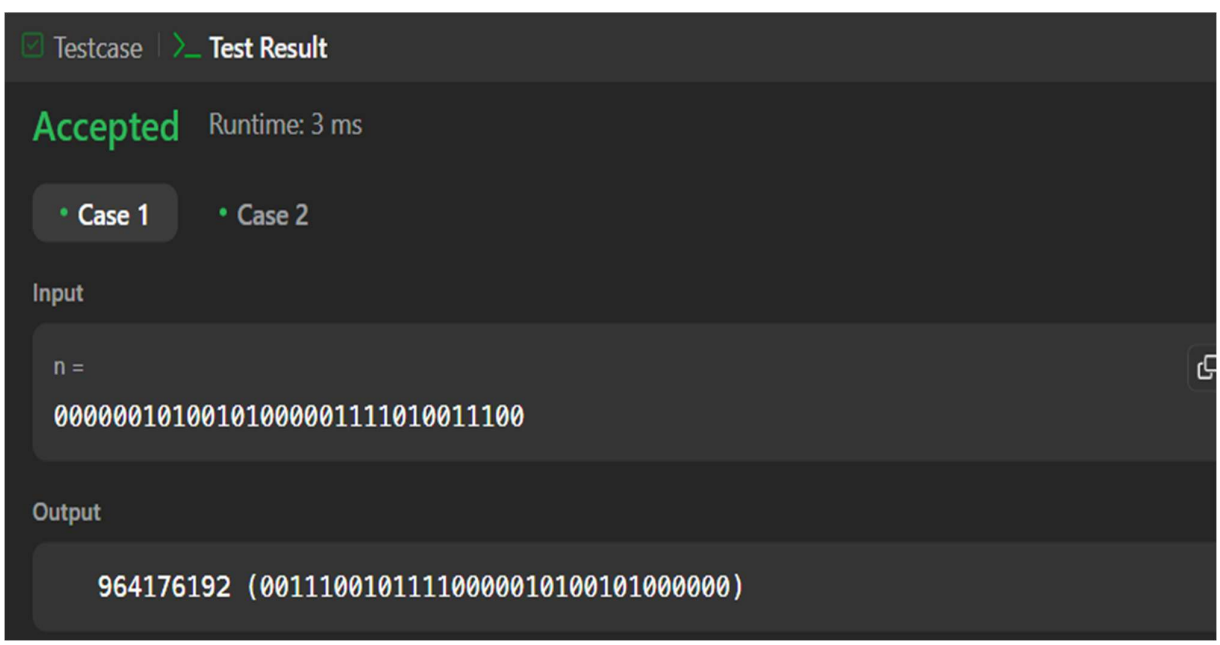


190.Reverse Bits

Code:

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

Output:



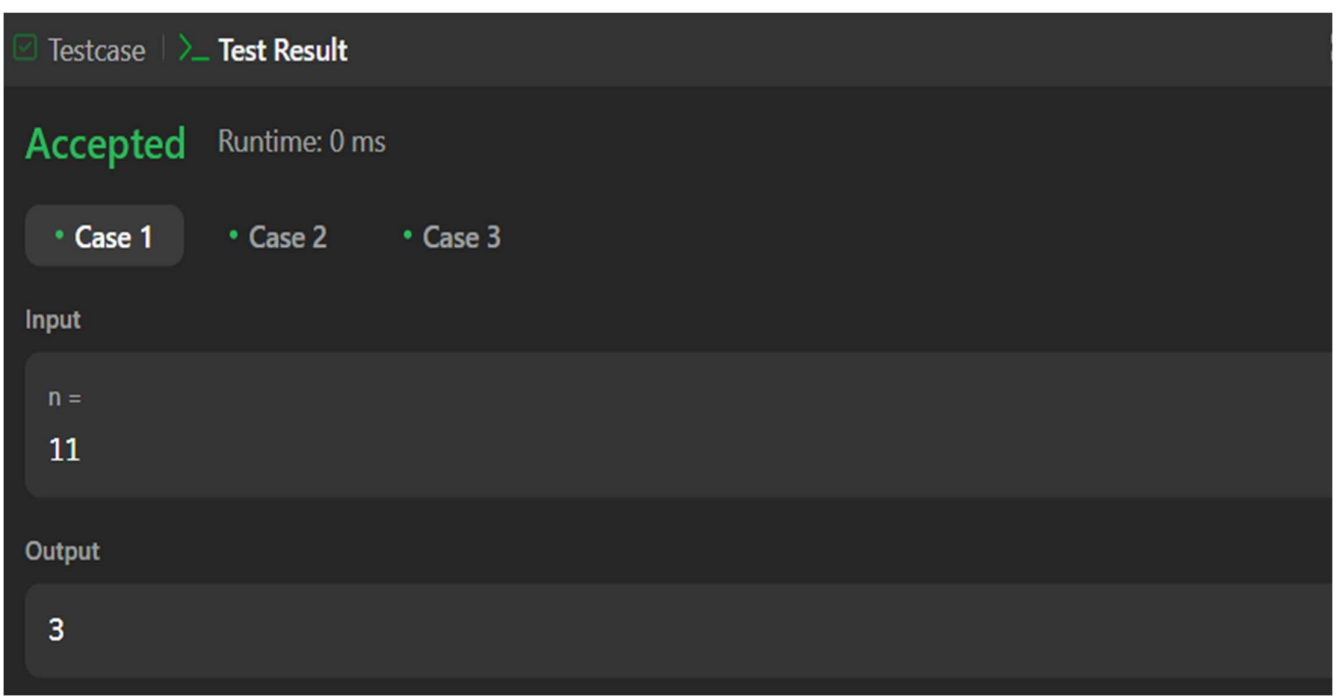
191.Number of 1 Bits

Code:

```
class Solution {
public:
   int hammingWeight(int n) {


    int count = 0;
      for(int i = 31; i >= 0; i--){
         if(((n >> i) & 1) == 1)
            count++;
      }
      return count;
   }
};
```

Output:



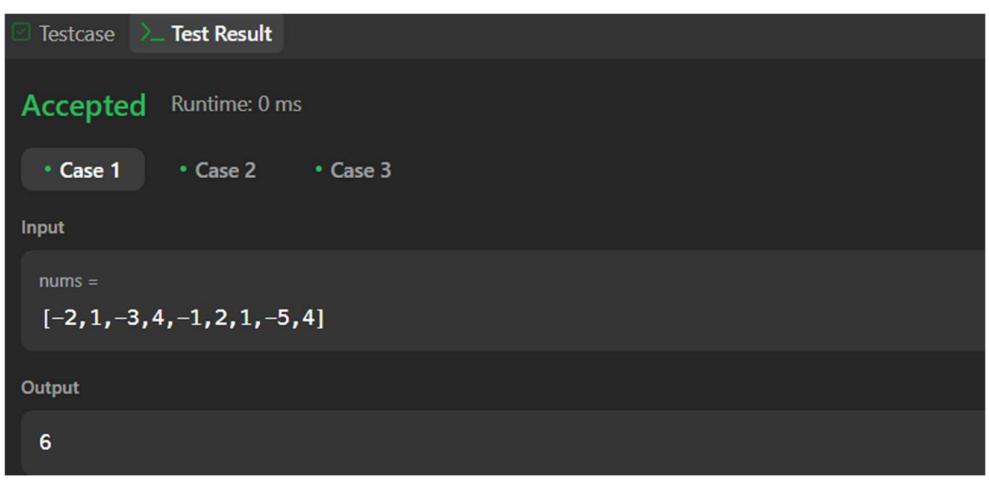
53.Maximum Subarray

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;
   }
};
```

Output:



240.Search a 2D Matrix II

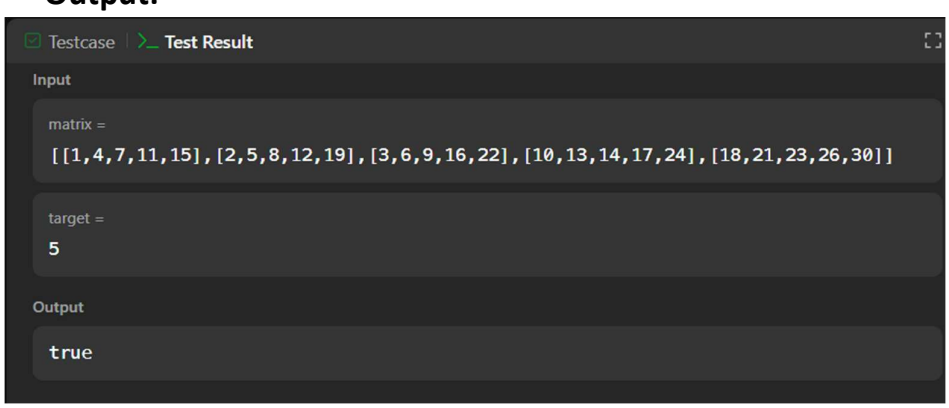
Code:

```
class Solution {
public:
   bool searchMatrix(vector<vector<int>>& matrix, int target) {
 int cols = matrix[0].size() - 1;
      int n = matrix.size() - 1;
      int rows = 0;

      while(rows <= n && cols >= 0){
         int toCompare = matrix[rows][cols];
         if(toCompare > target){
            cols--;
         }else if(toCompare < target){
            rows++;
         }else{
            return true;
         }
      }

      return false;
   }
};
```

Output:



372.Super Pow

Code:

```
class Solution {
private:
    int solve(int base, int power, int mod) {
        int ans = 1;
        while (power > 0) {
            if (power & 1) {
                ans = (ans * base) % mod;
            }
            base = (base * base) % mod;
            power >>= 1;
        }
        return ans;
    }

public:
    int superPow(int a, vector<int>& b) {
        a%=1337;
        int n = b.size();
        int m = 1140;
        int expi = 0;
        for(int i : b){
            expi = (expi*10+i)%m;
        }
        if (expi == 0) {
            expi = m;
        }
        return solve(a,expi,1337);
    }
};
```

Output: