

Ques 1. Longest Nice Substring.**Code:**

```
class Solution {  
public:  
    string longestNiceSubstring(string s) {  
        int n = s.length();  
        if (n < 2) return "";  
        for (int i = 0; i < n; ++i) {  
            char ch = s[i];  
            if (s.find(tolower(ch)) == string::npos || s.find(toupper(ch)) == string::npos) {  
                string left = longestNiceSubstring(s.substr(0, i));  
                string right = longestNiceSubstring(s.substr(i + 1));  
                return left.length() >= right.length() ? left : right;  
            }  
        }  
        return s;  
    }  
};
```

Output:

The screenshot shows a code execution interface with a tab labeled 'Test Result'. The status 'Accepted' is displayed in green, along with 'Runtime: 0 ms'. Below this, three test cases are listed: 'Case 1', 'Case 2', and 'Case 3'. 'Case 1' is selected and highlighted. Under the 'Input' section, the variable 's' is assigned the value '"YazaAay"'. The 'Output' section shows '"aAa"', and the 'Expected' section also shows '"aAa"', indicating a successful match.

```
Testcase | >_ Test Result  
  
Accepted Runtime: 0 ms  
• Case 1 • Case 2 • Case 3  
  
Input  
s =  
"YazaAay"  
  
Output  
"aAa"  
  
Expected  
"aAa"
```

Ques 2. Reverse Bits.

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 & 1);
            n >>= 1;
        }
        return result;
    }
};
```

Output:

☒ Testcase | [> Test Result](#)

Accepted Runtime: 2 ms

• Case 1 • Case 2

Input

n =
00000010100101000001111010011100

Output

964176192 (00111001011110000010100101000000)

Expected

964176192 (00111001011110000010100101000000)

Ques 3. Number of 1 Bits.

Code:

```
class Solution {  
public:  
    int hammingWeight(int n) {  
        int count = 0;  
        while (n) {  
            count += (n & 1); // Add 1 if the last bit is set  
            n >>= 1; // Right shift n to check the next bit  
        }  
        return count;  
    }  
};
```

Output:

☒ Testcase |  Test Result

Accepted Runtime: 0 ms

• Case 1

• Case 2

• Case 3

Input

n =
11

Output

3

Expected

3

Ques 4. Maximum Subarray.

Code:

```
class Solution {  
public:  
    int maxSubArray(vector<int>& nums) {  
        int maxSum = nums[0], currentSum = nums[0];  
        for (int i = 1; i < nums.size(); i++) {  
            currentSum = max(nums[i], currentSum + nums[i]);  
            maxSum = max(maxSum, currentSum);  
        }  
        return maxSum;  
    }  
};
```

Output:

☒ 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

Ques 5. Search a 2-D matrix II.

Code:

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int m = matrix.size(), n = matrix[0].size();
        int row = 0, col = n - 1; // Start from the top-right corner

        while (row < m && col >= 0) {
            if (matrix[row][col] == target) return true; // Found the target
            else if (matrix[row][col] > target) col--; // Move left
            else row++; // Move down
        }

        return false;
    }
};
```

Output:

☒ Testcase | [> Test Result](#)

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

Ques 6. Super Pow.

Code:

```
class Solution {
public:
    const int MOD = 1337;

    int modPow(int a, int b) {
        int result = 1;
        a %= MOD;
        while (b > 0) { a = (a * a) % MOD;
            b /= 2;}
        return result;
    }

    int superPow(int a, vector<int>& b) {
        a %= MOD;
        int result = 1;
        for (int digit : b) {
            result = (modPow(result, 10) * modPow(a, digit)) % MOD; }
        return result;
    }
};
```

Output:

☒ Testcase

☐ Test Result

Accepted Runtime: 0 ms

• Case 1

• Case 2

• Case 3

Input

a =
2

b =
[3]

Output

8

Expected

8