ALGO WAR

21 Feb ‘23

Rules

- Write Simple Algorithm/Psedo Code.
- Should NOT write Code; .py .cpp not allowed
- Steps should be clear; points will be given on the basis of the coverage of all border cases
- Don’t use complex functions in STL like - rotate() or accumulate()
- However you can use the available data structures like a = heap(), a = list()
- Dont focus too much on variable naming, getting input.
- 5 mins for each question.

Question 1

Given a number ‘n’ , print “yes” if you can partition the given number into two even numbers, and print the two numbers. If you cannot partition the given number into two even numbers, print “no”.

0 <= n

Question 2

Input is a string from A-Z. If you are encountering a letter for a first time, it contributes 2 points , if not, then only 1 point. Given the string ‘s’. Print the total points of the string.

ABAB

2 + 2 + 1 + 1 = 6

Question 3

string ‘a’ consisting of only 0 and 1. add + and – between all pairs of consecutive positions to make the absolute value of the resulting expression as small as possible.

01101

+-++

0+1-1+0+1 = 1

Question 4

An array ‘a’ is called **ugly** if it contains **at least one element** which is equal to the sum of all elements before it. If the array is not ugly, it is **beautiful**.

[6,3,9,6] is ugly -> 9 = 3+6

[5,5,7] is ugly.

You are given array ‘a’, in increasing order. Reorder array to make it beautiful. If impossible print “NO”

3 3 6 6

Output = 3 6 3 6

10 10

Output = NO

Question 5

You are given a seq a1,a2,...an consisting of integers bw 1 and m. You also have a string of length m consisting of all ‘B’.

1 operation : replace character at ai or (m+1 - ai) with ‘A’. You can replace the character at any position multiple times through the operations. Find the lexicographically smallest string.

M = 5, seq = 1 1 3 1

ABABA

M = 4, seq = 1 3

AABB

SOLUTIONS

Question 1

```
Check whether a number is odd or even (num%2==0)
If (Even && num!=2)
    print(yes)
    print(2,num-2) // or print valid even numbers
Else
    print(no)
```

Question 2

#1:

Use hash map (say count) and traverse through string

```
If count[‘letter’] ==0
    sum+=2
Else if count[‘letter’] >0
    sum+=1
count[‘letter’]++
```

#2: Use an alphabet array (Arr[26]) in order to increment the count of occurence

Question 3

Traverse through the string and maintain a flag that toggles for every alternate occurrence of one

(flag = 0 if no. of 1’s visited is odd and flag = 1 if no. of 1’s visited is even)

Append ‘+’ if number is 0 and next number is either 1 or 0 or if number is 1 and next number is 0

Append '-' to the solution if flag = 1 and make flag = 0

Append '+' to the solution if flag = 0 and make flag = 1

Print the solution string

Question 4

Concept:

If we put the maximum in the array on the first position, then for every element, starting from the third one, the sum of elements before it will be greater than it (since that sum is greater than the maximum value in the array). So, the only element that can make our array ugly is the second element. We need to make sure that it is not equal to the first element.

Let's put the maximum element on the first position, the minimum element on the second position, and then fill the rest of the array arbitrarily. The only case when it fails is when the maximum element is equal to the minimum element — and it's easy to see that if the maximum is equal to the minimum, then the first element of the array will be equal to the second element no matter what, and the array cannot become beautiful.

So, the solution is to check if the maximum is different from the minimum, and if it is so, put them on the first two positions, and the order of remaining elements does not matter. Note that the given array is sorted, so the minimum is the first element, the maximum is the last element.

Question 5

Initialise array of ‘B’s

Traverse the input sequence [1,1,3,1]

Decrement the number to make it the index in the array (say x)

```
x = min(x, m - x - 1);
if (s[x] == 'B') s[x] = 'A';
else s[m - x - 1] = 'A';
```