HackWithInfy Previous Problems and some assessment By Intellective Tech

Problem 1:-“Count And Say”

The count-and-say sequence is the sequence of integers with the first five terms as following:

```
1.     1

2.     11

3.     21

4.     1211

5.     111221
```

`1` is read off as `"one 1"` or `11`.
`11` is read off as `"two 1s"` or `21`.
`21` is read off as `"one 2`, then `one 1"` or `1211`.

Given an integer *n* where 1 ≤ *n* ≤ 30, generate the n^{th} term of the count-and-say sequence. You can do so recursively, in other words from the previous member read off the digits, counting the number of digits in groups of the same digit.

Note: Each term of the sequence of integers will be represented as a string.

Example 1:

```
Input: 1

Output: "1"

Explanation: This is the base case.
```

Example 2:

```
Input: 4

Output: "1211"

Explanation: For n = 3 the term was "21" in which we have two groups "2" and
"1", "2" can be read as "12" which means frequency = 1 and value = 2, the same
way "1" is read as "11", so the answer is the concatenation of "12" and "11"
which is "1211".
```

Problem 2:-“Efficient Janitor”

Find the minimum number of groups who's sum of each group is at max 3, and every element must be in a group.
Given an Array like: [1.01, 1.01, 3.0, 2.7, 1.99, 2.3, 1.7]
return the minimum number of groups, in this case it would be 5 groups: (1.01 , 1.99), (1.01, 1.7), (3.0), (2.7), (2.3)
Constraint: all elements are between 1.01-3 inclusive, and each groups sum is at max 3.

Problem 3:-“Division Pair Sum”

You are given an array **ar,** of integers and a positive integer **k,** . Find and print the number of (i,j) pairs,

where **i < j** and **ar[i]+ar[j]** is divisible by **k** .

For example, **ar=[1,2,3,4,5,6]** and **k=5** . Our three pairs meeting the criteria are **[1,4],[2,3]** and **[4,6].**

Function Description

divisibleSumPairs has the following parameter(s):

- *n*: the integer length of array
- *ar*: an array of integers
- *k*: the integer to divide the pair sum by

Input Format

The first line contains space-separated integers, n and k.

The second line contains space-separated n integers describing the values of .ar.

Output Format

Print the number of pairs where and + is evenly divisible by .

Sample Input

6 3

1 3 2 6 1 2

Sample Output

5

Explanation

Here are the 5 valid pairs when k=3 :

- **(0,2) -> ar[0]+ar[2] = 1+2 =3**
- **(0,5) -> ar[0]+ar[5] = 1+2 =3**
- **(1,3) -> ar[1]+ar[3] = 3+6 = 9**
- **(2,4) -> ar[2]+ar[4] = 2+1 =3**
- **(4,5) -> ar[4]+ar[5] = 1+2 =3**

Problem 4:-“Playing with primes”

Given an intger N.
N<=1e50
Output the no. of pairs (x,y) such that

1. 0<=x<=n
2. 0<=y<=n
3. F(x) +F(y) =Prime no.
   F(x) = sum of digits of x

(x,y) and (y,x) are to be treated as same pair
Input
3
ouput
5
Explaination
5 pairs (0,2) (1,1) (0,3) (2,3)(1,2) give prime no.s.

Problem 5:-“Minimum String Rotation”

A rotation on a string is defined as removing first element and concatenating it at the end

Given a number N and N strings .
Output the minimum no. of rotations on the strings so as to make all the strings equal.
If this is not possible print -1.

Input
4
11234
34112
41123
11234

Output
3

finally all the strings would be 11234
first and last string input needs no rotations.
second needs 2 rotations.
third needs 1 rotation.
So total rotation will be 3.

Problem 6:-“Animatter With Particles”

Given an integer(n) denoting the no. of particles initially
Given an array of sizes of these particles
These particles can go into any number of simulations (possibly none)
In one simulation two particles combines to give another particle with size as the difference between the size of them (possibly 0).
Find the smallest particle that can be formed.

constraints
n<=1000
size<=10e9
Example 1
3
30 10 8
Output
2
Explanation- 10 - 8 is the smallest we can achieve.

Example 2
4
1 2 4 8
Output
1
Explaination
We cannot make another 1 so as to get 0 so smallest without any simulation is 1

Example 3
5
30 27 26 10 6
Output
0
30-26=4
10-6 =4
4-4 =0

Problem 7:-“Crests And Troughs”

Given an array of n elements, find out all the crests and troughs along with their lengths. Find the absolute difference between indexes of longest and shortest crests as well as troughs and display the maximum of both.

- An element is said to be crest ,if both it's previous and next elements are less than the current element.
- An element is said to be trough , if both it's previous and next elements are greater than the current element.

Note: The first and last elements are neither crests nor troughs.

- In case of multiple occurrences of shortest crest/trough consider the left most one as shortest and right most one as longest.

- Print -1 if at least one crest and trough doesn't exist.

Input Format

The first line contain an integer, t denoting the number of testcases.
The first line of each test case contains an integer, n denoting the size of the arr.
The second line of each test case contains n space-separated integers describing the elements of array.

Constraints
4<=n<=10e9.
0<=arr[i]<=n

Output Format

For every test case print the required output in a new line.

Sample Input 1
1
8
3 6 2 8 9 5 10 1

Output
2

Explanation

The crest with maximum length (length --->**10 - 1 =9**) exists at **index 6**.
The crest with minimum length (length ---->**9 - 8 =1**) exists at **index 4.**
The trough with maximum length (length---> **8 - 2=6)** exists at **index 2**.

The trough with minimum length (length ---> 6 - 2 = 4) exists at index 2 (troughs with length 4 exists at two indexes 2 and 5, but take trough at index as shortest trough as it is left most).

Print 2 as (difference between indexes of longest and shortest crests) **6 - 4 > 2 - 2** (differences between indexes of longest and shortest troughs).

Problem 8:-"Sam And Alex Game Play"

Sam and Alex are playing a new game where there are a number of piles, each with any number of stones in it. Players take turns removing stones from any one pile.
The number removed has to be either:
1. an integer multiple of a given number, *k,* where *k > 0.*
2. if there are fewer than *k* stones in a pile, any number can be removed.

Determine who wins the game. Sam always starts, they both play optimally, and the last player to remove a stone wins. If Sam wins, return " *Sam wins the game of n pile(s).* ", where *n* is the number of piles in the input. If Alex wins, return " *Alex wins the game of n pile(s).* ".
For example, a game starts with *n = 3* piles of stones that contain *piles* = ***[3, 5, 7]*** stones, and a constant ***k = 2***. Sam will go first and remove ***1 * k = 1 * 2*** stones from the third pile leaving ***7 - 2 = 5*** stones in that pile.
Player Removes Result Start **[3, 5, 7]** Sam 2 **[3, 5, 5]** Alex 2 **[3, 3, 5]** Sam 2 **[3, 3, 3]** Alex 2 **[1, 3, 3]** Sam 2 **[1, 1, 3]** Alex 2 **[1, 1, 1]** Sam 1 **[0, 1, 1]** Alex 1 **[0, 0, 1]** Sam 1 **[0, 0, 0]**
In this case, Sam wins so " *Sam wins the game of 3 pile(s).* " is returned.
Function Description:
Complete the function *gameOfPiles* in the editor below. The function must return a string that denotes the result of the game.
gameOfPiles has the following parameters:
piles[piles[0], piles[1],…piles[n-1]] : an array of integers, each *piles[i]* (where *0 ≤ i < n*) represents the number of stones in the *ith* pile. *k:* an integer
Constraints
- *1 ≤ n ≤ 105*
- *1 ≤ piles[i] ≤ 1000*
- *1 ≤ k ≤ 1000*

Input Format For Custom Testing Sample Case 0
Sample Input For Custom Testing
2 2 2 1
Sample Output
Alex wins the game of 2 pile(s).
Explanation
There are multiple optimal scenarios for this game. Here is a scenario as an example where *k = 1* .
Sam first removes *1* stone from the first pile, the configuration becomes *[1, 2]* .
Alex then removes *1* stone from the second pile, the configuration becomes *[1, 1]* .
Sam then removes *1* stone from the first pile, the configuration becomes *[0, 1]* .

At last, Alex removes *1* stone from the second pile, the configuration becomes *[0, 0]* . Alex makes the last move so Alex wins.