Code Jam 2022 - Round 1B

ASeDatAb

Problem

A research consortium has been looking for the best possible database for three years, but they are still having problems. The database stores values as records that hold 8-bit binary strings. Unfortunately, their implementation of the function to set the value of a record is flawed.

Each record of the database is an 8-bit binary string. The bits of the binary string are indexed from 0 to 7 from left to right. When an instruction to set a specific record to a new value V is received, instead of setting the value to V the database does the following:

1. Choose an integer r between 0 and 7, inclusive, and let W be like V but rotated by r to the right. That is, the $((i+r) \bmod 8)$-th bit of W is the i-th bit of V.
2. Replace the current value X of the record with X XOR W. That is, the new value of the record has a 1 as its i-th bit if and only if the i-th bits of X and W are different.
3. Finally, return the number of bits that are 1 in the new value to the user.

Luckily, it turns out that no matter what the initial value is or what rotation values the database chooses, it is always possible to reset the value of a record to have all bits be 0 with no more than 300 uses of this operation. Implement a program to interact with the database that does this.

Input and output

This is an interactive problem. You should make sure you have read the information in the Interactive Problems section of our FAQ.

Initially, your program should read a single line containing an integer $\mathbf{T}$, the number of test cases. Then, $\mathbf{T}$ test cases must be processed.

At the beginning of each test case, the record in the database is set to a value that is not `00000000`. In each test case, your program must process up to 300 exchanges.

The i-th exchange starts with you outputting a single line containing a single 8-bit binary string to be used as the value V for the operation above. Then, the judge program performs the operation as described and sends you a single line containing a single integer $\mathbf{N_i}$ representing the number of bits that are equal to 1 in the updated value of the record.

- If $\mathbf{N_i} = 0$, it means that you have succeeded and you must start the next test case, or finish the program if it was the last one.
- If $\mathbf{N_i} = -1$ it means that this was the 300-th exchange of the test case but the record never got to a value of all zeroes, so the test is failed. No further test cases will be processed.
- If $1 \le \mathbf{N_i} \le 8$, it means that the updated value of the record has $\mathbf{N_i}$ ones and you may proceed to the next exchange to keep trying to make it contain only zeroes.

Your solution is considered correct if and only if you succeed in setting the value of the record to `00000000` for all test cases.

If the judge receives an invalidly formatted or invalid line from your program at any moment, the judge will print a single number -1 and will not print any further output. If you receive a -1, you must finish correctly and without exceeding the time or memory limits to receive a Wrong

Answer judgement. Otherwise, you will receive a judgement informing the exceeded resource or the incorrect termination condition.

Limits

Time limit: 10 seconds.
Memory limit: 1 GB.
$1 \le \mathbf{T} \le 100$.
$-1 \le \mathbf{N_i} \le 8$ for all i.

Test Set 1 (Visible Verdict)

The initial value of the record is chosen uniformly at random from all 8-bit binary strings that are not `00000000`.

Each rotation value is chosen uniformly at random, and independently of all previous choices and interactions.

Test Set 2 (Visible Verdict)

The judge is *adversarial*. This means, among other things, that the judge can change the initial value or rotation values as long as it is consistent with all interactions. The initial value is guaranteed to never be `00000000`.

Testing Tool

You can use this testing tool to test locally or on our platform. To test locally, you will need to run the tool in parallel with your code; you can use our interactive runner for that. For more information, read the instructions in comments in that file, and also check out the Interactive Problems section of the FAQ.

Instructions for the testing tool are included in comments within the tool. We encourage you to add your own test cases. Please be advised that although the testing tool is intended to simulate the judging system, it is **NOT** the real judging system and might behave differently. If your code passes the testing tool but fails the real judge, please check the Coding section of the FAQ to make sure that you are using the same compiler as us.

Download testing tool

Sample Interaction

Judge **Solution**

Number of cases

`1`

Case 1. Record starts at the hidden value `10000000`.

`00110011`

Judge picks $r = 5$ and rotates your given value to get `10011001` then does `10011001` XOR `10000000` obtaining `00011001`, which is the new value of the record.

`3`

`00011001` has 3 ones.

00011001

Judge picks $r = 0$ which leaves your input unrotated. Since it coincides with the current value of the record, this results in the record being `00000000`.

0

Judge informs you that there are no ones in the record, so this case is complete.