Code Jam to I/O for Women 2019 - Code Jam to I/O for Women

War of the Words

Problem

Finally, you are face to face with the leader of the alien robots! You have managed to distract it with its favorite word game, while your fellow Resistance members try to shut down the robots' power supply.

The game uses the robot language, which is made up of 10^5 words: the set of all five-letter strings made up of uppercase English letters between `A` and `J`, inclusive. For example, `AAAAA`, `FGHIJ`, and `BEIGE` are among the valid words.

You know that these words have a rank order, with no ties, such that a higher-ranked word beats any lower-ranked word, with the one exception that the lowest-ranked word beats the highest-ranked word. Unfortunately, you do not know the rank order of the words in the robot language!

The game has the following rules:

- Turn 0: You start by naming a word W_0.
- Turn 1: The robot names a word W_1. If W_1 beats W_0, the robot scores a point.
- This continues; on turn i, the active player is you if i is even, or the robot if i is odd. The active player names a word W_i and scores a point if W_i beats W_{i-1}. (If the two words are the same, no point is scored.) The score is not announced — in particular, you will not know whether each of your words has scored a point!
- This continues for a total of 201 turns.
- Notice that you get the last turn (naming W_{200}), and that at the end of that turn, each player's score is between 0 and 100, inclusive.

Thanks to some great work by the Resistance's spies, you know the strategy that the robot will use for every turn of the game. It only cares about scoring 100 points, so on every turn, it will choose a word uniformly at random from all possible words that will score a point on that turn, and independently of all of its previous word choices. The robot knows the rank order of the language, so it has no trouble choosing words!

If you do not score at least **N** points, the robot will become bored and stop playing with you, so your plan (and the universe) will be doomed!

Input and output

This problem is interactive, which means that the concepts of input and output are different than in standard Code Jam problems. You will interact with a separate process that both provides you with information and evaluates your responses. All information comes into your program via standard input; anything that you need to communicate should be sent via standard output. Remember that many programming languages buffer the output by default, so make sure your output actually goes out (for instance, by flushing the buffer) before blocking to wait for a response. See the FAQ for an explanation of what it means to flush the buffer. Anything your program sends through standard error is ignored, but it might consume some memory and be counted against your memory limit, so do not overflow it.

In addition, sample solutions to a previous Code Jam interactive problem (in various languages) are provided in the Analysis section of this past problem.

Initially, your program should read a single line containing two integers **T** and **N**: the number of test cases, and the number of points you must score in each case. (Notice that the **N** parameter is the same for every test case in a test set.) Then, you need to process **T** test cases.

At the start of a test case, the judge chooses a rank order for the words in the robot language, uniformly at random from all such orders, and independently of any previous choices of rank order.

(However, the judge uses a deterministic seed, so if your program is deterministic, its interaction with the judge will be the same even across submissions for this problem.)

At the start of a test case, you must output one line with a five-letter string of uppercase English letters in the inclusive range `A` through `J`: a valid word W_0 in the robot language. Then you will have 100 exchanges with the judge; the j-th of these (counting starting from 1) proceeds as follows:

1. The judge will output one line with a valid word $\mathbf{W_{2j-1}}$: the robot's word.
2. You output one line with another valid word W_{2j}.

After you send your last word for a test case, your program should terminate if it was the last test case; otherwise, it should output a word to begin the next test case.

If your program outputs something wrong (e.g., gives an invalid word), the judge will send `-1` to your input stream and it will not send any other output after that. If your program continues to wait for the judge after receiving `-1`, your program will time out, resulting in a Time Limit Exceeded error. Notice that it is your responsibility to have your program exit in time to receive a Wrong Answer judgment instead of a Time Limit Exceeded error. As usual, if the total time or memory is exceeded, or your program gets a runtime error, you will receive the appropriate judgment.

You should not send additional information to the judge after processing all test cases. In other words, if your program keeps printing to standard output after the last test case, you will get a Wrong Answer judgment.

Limits

$1 \le \mathbf{T} \le 50$.
Time limit: 40 seconds per test set. (10 seconds per test run.)
Memory limit: 1GB.
$\mathbf{W_i}$ is five characters long and consists only of characters in the set of uppercase letters `ABCDEFGHIJ`, for all odd i. (The robot plays valid words.)

Test set 1 (Visible)

N = 25.

Test set 2 (Visible)

N = 50.

Sample Interaction

Notice that interactive problems do not have sample input and output like other Code Jam problems. Your code will not be run against samples, and the following interaction is just an example.

```
t, n = readline_int_list()   // reads 50, 25 into t, n
// Before the case starts, the judge silently chooses a rank order for the
// 100000 words. Suppose that the word FADED is the lowest-ranked word,
// AHEAD is the highest-ranked word, and CAGED and HIGCJ are ranked
// such that FADED < CAGED < HIGCJ < AHEAD. As the contestant,
// though, we do not know this!
printline AHEAD to stdout    // you name AHEAD as word 0. What luck!
flush stdout
robot_word = readline_str()  // reads FADED into robot_word
printline CAGED to stdout    // you name CAGED as word 2
flush stdout
robot_word = readline_str()  // reads HIGCJ into robot_word
printline GAFFED to stdout   // you name an invalid word as word 4
flush stdout
```

```
robot_word = readline_str()  // reads -1 (judge has decided our solution is
                             //   incorrect)
exit                         // exits to avoid an ambiguous TLE error
```

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