Code Jam 2019 - Round 2

Pottery Lottery

Problem

The Pottery Palace is going to run a lottery featuring some valuable vases by the artist Cody-Jamal. The lottery works as follows:

- 100 people get to play in the lottery. Each player has a unique number between 1 and 100, and is given a single token with that number.
- There are 20 empty clay vases on a table, numbered 1 through 20. The vases have narrow openings that are large enough to accept a token, but small enough that players cannot look inside to see the contents.
- On the i-th day of the lottery, the player with token number i chooses a vase and puts their token in that vase. Since the vases are all identical (apart from their labels), every player will choose one uniformly at random and independently of all other players' choices.
- On day 100, after player number 100 has inserted their token, the organizers shake the vases to determine how many tokens are inside each one. If there is *exactly* one vase that has fewer tokens than any other vase, then that one is the "winning vase". The organizers then pour out all of the tokens in that vase, and every player whose number is written on one of those poured-out tokens wins a vase! If multiple vases have the same minimal amount of tokens, nobody wins anything.

You have been hired to test the security of the lottery, and you will participate in some trial runs. The company will always assign you the number 100 — that is, you replace player 100.

You have found some ways to tamper with the lottery at night, but security is tight, so you can only do so much! Specifically, after each of the first 99 days of the lottery, you may do exactly *one* of the following:

- forge a token with the player number of your choice (between 1 and 100, inclusive), and add it to a vase of your choice. You are a very good forger: if there is a winning vase, any forged tokens in that vase will cause the players with those numbers to win (with one exception; see below).
- use a special camera to see the numbers on all of the tokens in one vase of your choice

You may perform different actions on different nights, and you may choose dynamically: you do not need to decide on all of your actions in advance.

On the 100th day, it is your turn to insert your token into a vase of your choice (you do not need to choose uniformly at random). You cannot perform any other actions on that day.

You know that if there is a winning vase with more than one token for the same player, it will be obvious that cheating has occurred and nobody will win. However, it does not matter if other vases contain more than one token for the same player because the organizers never see those tokens.

Your goal is to be a winner in at least 90% of the test cases.

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 a single integer **T** indicating the number of test cases. Then, you need to process **T** test cases.

At the start of a test case, the judge outputs one line with one integer: the number of the current day. (The judge starts on day 1, and on the i-th day, it prints i.) After your program reads the integer, it should write a line containing two integers V and P, with $1 \le V \le 20$, and $0 \le P \le 100$. The judge will interpret these as follows:

- If $1 \le P \le 100$, you put a token for player P in vase V. The judge does not write anything back as a response.

- If P = 0, you inspect the contents of vase V. The judge writes one line containing integers. The first integer is **N**, the number of tokens in vase V, and then there are **N** more integers: the player numbers on each of the tokens, in non-decreasing order.

Notice that on turn 100, you must put your own token in, so P must be 100.

Remember that on the i-th day, for 1 ≤ i ≤ 99, the judge simulates the action of the i-th player, as described in the statement. This happens before your own action on that day.

After you send your move for turn 100, your program should terminate if it was the last test case; otherwise, it should start reading data for the next test case. (Notice that the judge does not tell you whether you got each case correct or incorrect.) The judge will only check whether you have enough correct answers after you have attempted all **T** test cases, so you should not stop early! For example, if you answer the first 225 out of 250 cases correctly and then exit, or provide malformed input, your solution will not be considered correct.

If your program outputs something illegal (e.g., gives an invalid value for P or V, or tries to inspect a vase on turn 100), the judge will send one line containing `-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 memory is exceeded, or your program gets a runtime error, you will receive the appropriate judgment.

Test set 1 (Visible)

T = 250.
Time limit (for the entire test set): 40 seconds.
Memory limit: 1GB.

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

```
t = readline_int()            // reads 250 into t
curr_day = readline_int()     // reads 1 (day 1)
printline 8 100 to stdout     // puts a token for player 100 into vase 8
flush stdout
curr_day = readline_int()     // reads 2 (day 2)
printline 8 99 to stdout      // puts a token for player 99 into vase 8
flush stdout
curr_day = readline_int()     // reads 3 (day 3)
printline 8 100 to stdout     // puts a token for player 100 into vase 8
flush stdout
curr_day = readline_int()     // reads 4 (day 4)
printline 20 7 to stdout      // puts a token for player 7 into vase 20
flush stdout
curr_day = readline_int()     // reads 5 (day 5)
printline 8 0 to stdout       // inspects vase 8
flush stdout
tokens = readline_int_list() // reads 5 2 5 99 100 100 (players 2 and 5
```

```
                                //   happen to have chosen vase 8)
curr_day = readline_int()       // reads 6 (day 6)
printline 8 101 to stdout       // tries to add a token with a bad player number
flush stdout
curr_day = readline_int()       // reads -1 (judge has decided our solution is
                                //   incorrect)
exit                            // exits to avoid an ambiguous TLE error
```