Code Jam 2020 - Round 3

Pen Testing

Problem

You have **N** ballpoint pens. You know that each has a distinct integer number of units of ink between 0 and **N**-1, but the pens are given to you in random order, and therefore you do not know which pen is which.

You are about to go on a trip to the South Pole (where there are no pens), and your luggage only has room for two pens, but you know you will need to do a lot of important postcard writing. Specifically, the two pens you choose must have a total of at least **N** ink units.

Your only way to get information about the pens is to choose one and try writing something with it. You will either succeed, in which case the pen will now have one unit of ink less (and is now possibly empty), or fail, which means that the pen already had no ink left. You can repeat this multiple times, with the same pen or different pens.

Eventually, you must select the two pens to take on your trip, and you succeed if the total amount of ink remaining in those two pens is at least **N** units.

You will be given **T** test cases, and you must succeed in at least **C** of them. Note that all test sets in this problem are Visible.

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 three integers **T**, **N**, and **C**: the number of test cases, the number of pens, and the minimum number of test cases you must succeed in. (Note that the value of **N** is the same for all test sets, and is provided as input only for convenience; see the Limits section for more details.)

Then, your program needs to process all **T** test cases at the same time (this is done to reduce the number of roundtrips between your solution and the judging program). The interaction is organized into rounds.

At the beginning of each round, your program must print one line containing **T** integers: the **i**-th integer is the number of the pen you want to try writing with in the **i**-th test case, or 0 if you do not want to write with any pen in this test case in this round. The pens are numbered from 1 to **N**.

Be aware that flushing the output buffer after each one of these integers, instead of only once after printing all **T**, could cause a Time Limit Exceeded error because of the time consumed by the flushing itself.

The judge responds with one line containing **T** integers: the **i**-th integer is the amount of ink spent in the **i**-th test case in this round. It will be equal to 1 if the writing in the **i**-th test case was successful. Otherwise, it will be equal to 0, which could mean that you tried to write in the **i**-th test case but the pen you chose had no ink left, or that you did not try to write in the **i**-th test case at all.

You may participate in at most **N**×(**N**+1)/2 rounds. Note that this is enough to be confident that all pens are empty.

When your program is ready to submit an answer for all test cases, it must print a line containing the number 0 **T** times. This line is not counted towards the limit on the number of rounds, and the judge will not send a response.

Then, your program must print another line with 2×**T** integers: the (2×**i**-1)-th and the (2×**i**)-th integers in this line are the distinct numbers of the pens that you take to the South Pole in the **i**-th test case. The judge will not send a response, and your program must then terminate with no error.

If the judge receives unexpected output from your program at any moment, the judge will print a single number -1 and not print any further output. If your program continues to wait for the judge after receiving a -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 memory limit is exceeded, or your program gets a runtime error, you will receive the appropriate judgment.

You can assume that the pens are given to you in random order. These orders are chosen uniformly at random and independently for each test case and for each submission. *Therefore even if you submit exactly the same code twice the judge will use different random orders.*

Limits

Time limit: 90 seconds per test set.
Memory limit: 1GB.
N = 15.

Test Set 1 (Visible Verdict)

T = 20000.
C = 10900 (**C**=0.545×**T**).

Test Set 2 (Visible Verdict)

T = 20000.
C = 12000 (**C**=0.6×**T**).

Test Set 3 (Visible Verdict)

T = 100000.
C = 63600 (**C**=0.636×**T**).

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

The following interaction does not correspond to any of the three test sets, as its values of **T** and **N** are too small. It merely serves to demonstrate the protocol.

Input to your program Output of your program

```
2 5 1
                4 5
1 0
                4 3
0 1
                0 2
0 1
                0 0
                3 4 3 4
```

Here is the same interaction, explained:

```
  // The following reads 2 into t, 5 into n and 1 into c.
  t, n, c = readline_int_list()
  // The judge secretly picks the number of units for each pen:
  // in test case 1: 2 0 4 1 3
  // in test case 2: 1 3 2 4 0
  // We write with the 4-th pen in test case 1, and with the 5-th pen in test case 2.
  printline 4 5 to stdout
  flush stdout
  // Reads 1 0, as the 4-th pen in test case 1 still had ink left,
  // but the 5-th pen in test case 2 did not.
  a1, a2 = readline_int_list()
  // We write with the 4-th pen in test case 1 again, and with the 3-rd pen in test case 2.
  printline 4 3 to stdout
  flush stdout
  // Reads 0 1.
  a1, a2 = readline_int_list()
  // We only write in test case 2 this time, with the 2-nd pen.
  printline 0 2 to stdout
  flush stdout
  // Reads 0 1.
  a1, a2 = readline_int_list()
  // We decide we are ready to answer.
  printline 0 0 to stdout
  flush stdout
  // We take the 3-rd and the 4-th pens to the South Pole in both test cases.
  printline 3 4 3 4 to stdout
  flush stdout
  // In test case 1, the remaining amounts in the 3-rd and the 4-th pens are 4 and 0, and 4+0<5,
  // so we did not succeed.
  // In test case 2, the remaining amounts in the 3-rd and the 4-th pens are 1 and 4, and 1+4≥5,
  // so we succeeded.
  // We have succeeded in 1 out of 2 test cases, which is good enough since c=1.
  exit
```