Kick Start 2019 - Practice Round

Number Guessing

Welcome to the Practice Session!

If you experience any technical issues interfering with your ability to participate in the Practice Session, please email us immediately at codejam@google.com. We will have limited support during the session, but will get back to you as soon as possible. For all other feedback, we invite you to submit your thoughts and suggestions via this feedback form after the Practice Session.

Problem

This problem is a well-known classic; we present it primarily as an opportunity for you to try out the interactive judging system.

We are thinking of an integer P within the range (**A**,**B**] — that is, **A** < P ≤ **B**. You have **N** tries to guess our number. After each guess that is not correct, we will tell you whether P is higher or lower than your guess.

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. To help you debug, a local testing tool script (in Python) is provided at the very end of the problem statement.

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.

For each test case, your program will read a single line with two integers **A** and **B**, representing the exclusive lower bound and inclusive upper bound, as described above. In the next line, you will read a single integer **N**, representing the maximum number of guesses you can make. Your program will process up to **N** exchanges with our judge.

For each exchange, your program needs to use standard output to send a single line with one integer Q: your guess. In response to your guess, the judge will print a single line with one word to your input stream, which your program must read through standard input. The word will be `CORRECT` if your guess is correct, `TOO_SMALL` if your guess is less than the correct answer, and `TOO_BIG` if your guess is greater than the correct answer. Then, you can start another exchange.

If your program gets something wrong (e.g., wrong output format, or out-of-bounds values), the judge will send `WRONG_ANSWER` 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 `WRONG_ANSWER`, 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 the appropriate verdict (Wrong Answer, Runtime Error, etc.) 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 verdict.

If your test case is solved within **N** tries, you will receive the `CORRECT` message from the judge, as mentioned above, and then continue to get input (a new line with two integers **A** and **B**, etc.) for the next test case. After **N** tries, if the test case is not solved, the judge will print `WRONG_ANSWER` and then stop sending output to your input stream.

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

Limits

1 ≤ **T** ≤ 20.
A = 0. **N** = 30.
Time limit: 10 seconds per test set.
Memory limit: 1GB.

Test set 1 (Visible)

B = 30.

Test set 2 (Hidden)

B = 10^9.

Sample interaction

Here is a piece of pseudocode that demonstrates an interaction for one test set. Suppose there are three test cases in this test set. The pseudocode first reads an integer t, representing the number of test cases. Then the first test case begins. Suppose the correct answer P is 9 for the first test case. The pseudocode first reads three integers a, b, and n, representing the guessing range and maximum number of tries, respectively, and then outputs a guess 30. Since 30 is greater than 9, the string `TOO_BIG` is received through stdin from the judge. Then the pseudocode guesses 5 and receives `TOO_SMALL` in response. The guess 10 is subsequently printed to stdout which is again too big. Finally the pseudocode guesses 9, and receives `CORRECT` because 9 is the correct answer.

```
t = readline_int()          // reads 3 into t
a, b = readline_two_int()   // reads 0 into a and 30 into b; note that 0 30 is one line
n = readline_int()          // reads 30 into n
printline 30 to stdout      // guesses 30
flush stdout
string s = readline()       // because 30 > 9, reads TOO_BIG into s
printline 5 to stdout       // guesses 5
flush stdout
s = readline()              // reads TOO_SMALL into s since 5 < 9
printline 10 to stdout      // guesses 10
flush stdout
s = readline()              // reads TOO_BIG into s since 10 > 9
printline 9 to stdout       // guesses 9
flush stdout
s = readline()              // reads CORRECT into s
```

The second test case shows what happens if the code continues to read from stdin after the judge stops sending info. In this example, the contestant guesses 31, which is outside the range (0, 30]. As a result, the judging system sends `WRONG_ANSWER` to the input stream of the pseudocode and stops sending anything after that. However, after reading `WRONG_ANSWER` into string s, the code continues to read for the next test case. Since there is nothing in the input stream (judge has stopped sending info), the code hangs and will eventually receive a Time Limit Exceeded Error.

```
a, b = readline_two_int()   // reads 0 into a and 30 into b; note that 0 30 is one line
n = readline_int()          // reads 30 into n
printline 31 to stdout      // guesses 31
flush stdout
string s = readline()       // reads WRONG_ANSWER
a, b = readline_two_int()   // tries to read for the third test case but hangs since
                            // judge has stopped sending info to stdin
```

If the code in the example above exits immediately after reading `WRONG_ANSWER`, it will receive a Wrong Answer judgment instead.

```
a, b = readline_two_int()   // reads 0 into a and 30 into b; note that 0 30 is one line
n = readline_int()          // reads 30 into n
printline 31 to stdout      // guesses 31
flush stdout
string s = readline()       // reads WRONG_ANSWER
exit                        // receives a Wrong Answer judgment
```

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