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, $A < P \le B$. 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 \mathbf{T} indicating the number of test cases. Then, you need to process \mathbf{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, T00_SMALL if your guess is less than the correct answer, and T00_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 $\bf N$ 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 $\bf A$ and $\bf B$, etc.) for the next test case. After $\bf N$ 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

 $1 \le T \le 20$. **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 T00_BIG is received through stdin from the judge. Then the pseudocode guesses 5 and receives T00_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
                           // quesses 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 T00 SMALL into s since 5 < 9
                           // guesses 10
printline 10 to stdout
flush stdout
                           // reads TOO_BIG into s since 10 > 9
s = readline()
printline 9 to stdout
                           // quesses 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.

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