Code Jam 2020 - Round 1B

Blindfolded Bullseye

Problem

Gary has a large square wall that is exactly 2×10^9 nanometers tall and 2×10^9 nanometers wide. Gary has a dartboard placed on the wall. The dartboard is circular and its radius is between **A** and **B** nanometers, inclusive. The dartboard is fully contained within the wall, but it may touch its edges. The center of the dartboard is an integer number of nanometers from each edge of the wall.

Gary invited his friend Mika over to play an interesting game. Gary blindfolds Mika and challenges her to throw a dart at the center of the dartboard. To help her, whenever Mika throws a dart at the wall, Gary will tell her whether the dart hit the dartboard.

Mika does not know where on the wall the dartboard is, but since Mika is very skilled at darts, she can throw darts with nanometer precision. That is, she can aim and hit exactly any point that is an integer number of nanometers away from each edge of the wall. Immediately after throwing each dart, Gary tells her whether she hit the center of the dartboard, some other part of it, or missed it completely and hit the bare wall.

Can you help Mika hit the center of the dartboard, without throwing more than 300 darts?

Input and Output

Initially, your program should read a single line containing three integers **T**, **A** and **B**, indicating the number of test cases and the inclusive minimum and maximum values for the dartboard's radius, in nanometers, respectively. (Notice that **A** and **B** are the same for every test case within a test set.) Then, you need to process **T** test cases.

We represent the points that darts can be aimed at as pairs (x, y), where x and y are integers between -10^9 and 10^9, inclusive. The pair (x, y) is the point that is $x + 10^9$ nanometers away from the left edge of the wall and $y + 10^9$ nanometers away from the bottom edge of the wall. Point (0, 0) is therefore at the exact center of the wall.

For each test case, there is a secretly chosen radius R for the dartboard, and a secretly chosen center of the dartboard (X, Y). R, X, and Y are integers chosen for each test case by the judges in a designed (not random) way, within the limits. For each test case you need to process up to 300 exchanges with the judge. Your program represents Mika and the judge program represents Gary. Each exchange consists of Mika (your program) choosing where to throw a dart and Gary (the judging program) giving information about that position.

The i-th exchange consists of your program first outputting a single line containing two integers X_i and Y_i, both between -10^9 and 10^9, inclusive, and the judge responding with a single line containing either:

- `CENTER` if $X_i = X$ and $Y_i = Y$
- `HIT` if $0 < (X - X_i)^2 + (Y - Y_i)^2 \leq R^2$
- `MISS` in all other cases.

After sending `CENTER`, the judge will start waiting for the first exchange of the next test case, if there is any.

If you output a line that is incorrectly formatted or with an out of bounds value, the judge will respond with a single line containing `WRONG`. If 300 exchanges occur (including 300 responses from the judge) without you receiving `CENTER`, or if you ever receive `WRONG`, the judge will finish all communication,

wait for your own program to also finish, and give a Wrong Answer verdict. After sending the **T**-th `CENTER`, on the other hand, the judge will finish all communication, wait for your own program to finish, and give a Correct verdict. If, while waiting for your program to finish, time or memory limits are exceeded, the corresponding verdict will be assigned instead. (Note that verdicts are not messages sent to your program.)

Limits

Time limit: 30 seconds per test set.
Memory limit: 1GB.
$1 \leq \mathbf{T} \leq 20$.
$\mathbf{A} \leq R \leq \mathbf{B}$.
$-10^9 + R \leq X \leq 10^9 - R$.
$-10^9 + R \leq Y \leq 10^9 - R$.

Test set 1 (Visible Verdict)

$\mathbf{A} = \mathbf{B} = 10^9 - 5$.

Test set 2 (Visible Verdict)

$\mathbf{A} = \mathbf{B} = 10^9 - 50$.

Test set 3 (Hidden Verdict)

$\mathbf{A} = 10^9 / 2$.
$\mathbf{B} = 10^9$.

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

The interactive runner was changed after the 2020 Qualification Round. Be sure to download the latest version.

Sample Interaction

The following sample interaction uses the limits of Test Set 1.

```
// The following reads 20 into t and 999999995 into a and b.
t, a, b = readline_int_list()
// The judge secretly picks R = 999999995 (it had no choice) and X = -1,
// Y = 3 (it did have a choice here). (Note: the actual Test Set 1 will
// not necessarily use the values in this example.)
// We try to throw at the upper left corner of the wall, and the dartboard
// does not overlap with that point.
```

```
printline -1000000000 1000000000 to stdout
flush stdout
r = readline_string()  // reads MISS.
// We try to throw at the center of the wall. That does hit the dartboard,
// but not the center.
printline 0 0 to stdout
flush stdout
r = readline_string()  // reads HIT.
// We make a super lucky choice and throw at the center of the dartboard.
printline -1 3 to stdout
flush stdout
r = readline_string()  // reads CENTER.
// The judge begins the next test case. It secretly picks R = 999999995
// and X = 5, Y = 5.
// We accidentally throw a dart out of the allowed range.
printline -1234567890 1234567890 to stdout
flush stdout
r = readline_string()  // reads WRONG.
exit  // exits to avoid an ambiguous TLE error.
```