Catch Me If You Can

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

This is an interactive problem.

Carl Hanratty is a dour FBI agent usually pursuing criminals in the financial sector, but he also likes to play his Pokemon GO game in a park during his time off.

He still plays an old school version of the game with a working pokemon radar. This radar shows one, two or three paws telling him how far the pokemon is from the current player's location. Three paws means that the pokemon is not farther than $3 \cdot r$ meters from the player, two paws mean that it is not farther than $2 \cdot r$ meters, and one paw means not farther than r meters. It is known that the value of r is a real number between 1 and 100 meters, inclusive, but the exact value of r is unknown to Carl.

Since Carl is a very good FBI agent and always strives for excellence, he wants to improve his pokemon catching practice by utilizing a secret FBI analytical task force. You are a part of that analytical team. You will receive field information from Carl, and then direct his movements in order to find and catch the pokemon.

You may assume all the movements are happening on a two-dimensional plane. Carl is located at the point with coordinates $(0, 0)$, and he just saw the pokemon on the radar for the first time (with three paws). This means Carl is initially located $3 \cdot r$ meters away from the pokemon. To communicate with Carl, you are allowed to perform no more than 5 iterations of the following process:

1. You transmit to Carl an angle in degrees: the direction in which he should move. The angle might be any real value from 0 to 360 inclusive.
2. Carl goes straight in this direction until the radar indication changes. This happens if he catches the pokemon (then the game stops), reaches the radar zone that is closer to the pokemon (the number of paws on the radar decreases by 1) or reaches the radar zone that is farther from the pokemon (or pokemon disappears from the radar at all). In the last case, Carl immediately makes a step back to the previous three-paw zone, as he doesn't want to move in a direction which is definitely wrong.
3. Carl tells you the exact distance he traveled and the current number of paws p ($1 \le p \le 3$) of the radar zone. It means the current distance to the pokemon is $p \cdot r$. Note that the communication counts even if Carl returned a distance of 0 which means that walking any positive distance in the given direction will immediately result in radar indication change.

During your 5-th communication with Carl, his behavior is a bit different. As he knows he won't be able to get any instructions from you anymore, he ignores all the radar zones and moves straight in the direction you gave him until either he finds the pokemon or it totally disappears from the radar.

It is guaranteed that the pokemon location is fixed and does not change during each session of the game. It is also guaranteed that the unknown real value r lies between 1 and 100, inclusive.

Interaction Protocol

You have at most 5 communication attempts. To start the next attempt, your program must provide a direction to Carl by printing one real value a ($0 \le a \le 360$) on a separate line. After that, your program gets the result on the next line of the standard input.

If at any moment of time Carl is within $r/10$ from the actual pokemon location, he instantly catches it and returns you one final word "`Gotcha!`" (without quotes).

Otherwise, if you run out of communication sessions with Carl and he was unable to find the pokemon during the last move, he sends you one final line "`The pokemon ran away!`" (without quotes).

In all other cases, Carl transmits you the traveled distance d with at least ten digits after the decimal point and the integer number of paws p ($1 \leq p \leq 3$) of the new radar zone. These two numbers are separated by one space character.

Your communication with Carl must stop when the pokemon is caught or ran away.

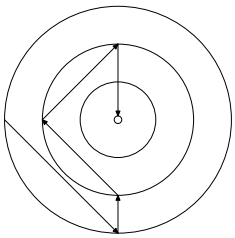
Examples

standard input	standard output
0.0000000000 3 3.0000000000 2 0.0000000000 2 0.0000000000 2 Gotcha!	90 0 270 90 0
14.1421356237 3 3.3333333333 2 9.4280904158 2 9.4280904158 2 Gotcha!	315 90 135 45 270

Note

In the first sample input, pokemon is sitting at point $(9, 0)$.

In the second sample input, pokemon is sitting at point $(10, 0)$.



The pipe from your program to the interactor program and the pipe back have limited size. Your program must read from the standard input to avoid deadlock. Deadlock condition is reported as "`Time Limit Exceeded`".

To flush the standard output stream, use the following statements:

In C, use `fflush(stdout);`

In C++, use `cout.flush();`

In Java, use `System.out.flush();`

In Python, use `sys.stdout.flush()`.

If your program receives an `EOF` (end-of-file) condition on the standard input, it MUST exit immediately with exit code 0. Failure to comply with this requirement may result in "`Time Limit Exceeded`" error.