Robot

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

This is an interactive problem.

A mischievous robot is navigating an infinitely large 2D grid. However, its movement is confined to the first quadrant, meaning its coordinates (x, y) must always satisfy $x > 0$ and $y > 0$. Initially, the robot is located at grid coordinates (s_x, s_y). All grid cells are initially white.

The game proceeds in discrete time steps, up to a maximum of $T = 1000$ steps. In each time step, the following sequence occurs:

1. **Your Move:** You choose integer coordinates (x_m, y_m) such that $1 \leq x_m \leq T$ and $1 \leq y_m \leq T$. You then mark this cell black. A cell, once marked black, remains black for the rest of the game. You can mark any cell within the range, including the cell where the robot is currently located.

2. **Robot's Move:** The robot, currently at position (r_x, r_y), observes the grid (including the cell you just marked black). It knows the locations of all currently black cells. It then moves to a new position (n_x, n_y). The new position (n_x, n_y) must be one of the 8 cells immediately adjacent (horizontally, vertically, or diagonally) to (r_x, r_y). That is, $|n_x - r_x| \leq 1$ and $|n_y - r_y| \leq 1$, and $(n_x, n_y) \neq (r_x, r_y)$. Additionally, the robot must stay within the first quadrant, so $n_x > 0$ and $n_y > 0$. The interactor (controlling the robot) will choose one valid move for the robot. The robot's strategy is unknown to you.

3. **Outcome Check:** After the robot moves to (n_x, n_y), the system checks if the cell (n_x, n_y) is black.

- If (n_x, n_y) is black, the robot explodes.
- If (n_x, n_y) is white, the game continues to the next time step, with the robot now at (n_x, n_y).

Your goal is to make the robot explode within T time steps.

Input

The first line of input contains two integers s_x and s_y ($1 \leq s_x, s_y \leq 20$), the initial coordinates of the robot.

Interaction Protocol

The interaction proceeds in turns for at most T turns. In each turn t (from $t = 1$ to T):

1. Your program must print one line containing two space-separated integers x_m and y_m, representing the coordinates of the cell you choose to mark black in this turn. Remember to flush the output stream.

2. The interactor reads your chosen coordinates (x_m, y_m).

3. The interactor determines the robot's next move to (n_x, n_y) based on its current position and the set of all black cells (including the one just marked at (x_m, y_m)).

4. The interactor checks if the cell (n_x, n_y) is black.

- If it is black (robot explodes), the interactor will print a single line containing '0 0' and terminate. Your program should then read these values and terminate successfully.
- If it is white, the interactor will print a single line containing two space-separated integers n_x and n_y, the new coordinates of the robot. Your program should read these coordinates to know the robot's current position for the next turn.

If the robot has not exploded after you have made T moves, your program should terminate. Your solution will be judged as incorrect in this case (or if you exceed the turn limit or make an invalid move).

To flush your output, you can use:

- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java.
- `stdout.flush()` in Python.

Note: If your program does not terminate properly, it may be judged as a Presentation Error (PE).

Example

standard input	standard output
5 5 5 4 5 3 5 2 5 1 0 0	 6 1 4 1 6 2 4 2 5 2