Skynet

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

This is an interactive problem.

In a layered defense system, there is an $n \times m$ grid. Rows are numbered 1 to n from top to bottom, and columns are numbered 1 to m from left to right. Some cells initially contain obstacles.

An invading drone starts from an empty cell in row n and attempts to break through upward. You need to place additional obstacles to intercept it before it reaches row 1.

Suppose the drone is currently at cell (r, c). If $r > 1$, it moves upward one row in this round, and its target can only be one of the following three positions: $(r-1, c-1)$, $(r-1, c)$, $(r-1, c+1)$. The cell it moves to must satisfy:

- It is within the board boundaries;
- There is no obstacle in that cell.

If the drone reaches row 1, it is considered to have successfully broken through, and your program fails.

After each move of the drone, if it is currently at (r, c) with $r > 1$, you may place **at most one** new obstacle. If you choose to place one, the obstacle must be placed in a currently empty cell (x, y) satisfying $x \in \{r-1, r-2\}$, $1 \le x \le n$, $1 \le y \le m$. That is, you can only add an obstacle in the first or second row above the drone's current position. Newly placed obstacles remain permanently; you may also choose not to place any obstacle in that round.

Your goal is to make the drone unable to perform a move at some moment, thereby intercepting it successfully.

It is guaranteed that for every test case, there exists at least one strategy that ensures you can intercept the drone before it reaches row 1.

Input

The first line contains two integers n and m ($2 \le n, m \le 100$), the number of rows and columns of the board.

The next n lines each contain a string s_i of length m, describing the initial board state:

- If $s_{i,j} =$ `#`, then cell (i, j) initially contains an obstacle;
- If $s_{i,j} =$ `.`, then cell (i, j) is initially empty.

It is guaranteed that row n contains at least one empty cell.

The last line contains two integers r_0 and c_0, the initial position of the drone (r_0, c_0). It is guaranteed that $r_0 = n$ and that this cell is initially empty.

After reading the input, you will interact according to the following rules.

Interaction Protocol

After reading the input, you repeatedly perform the following interaction until the process ends:

1. Output two integers x y.
    - If $(x, y) = (0, 0)$, it means you choose not to place an obstacle in this round;

- Otherwise, it means you place a new obstacle at cell (x, y). In this case the following must hold:
  - The cell is within the board boundaries;
  - The cell currently contains no obstacle;
  - If the drone is currently in row t, then you must have $x \in \{t-1, t-2\}$.

2. Then the interactor moves the drone one step and outputs two integers r and c:
   - If $(r, c) = (0, 0)$, the drone can no longer make a move, meaning you have successfully intercepted it. Your program should terminate immediately;
   - If $(r, c) = (-1, -1)$, you have failed to intercept it. This happens when the drone has already reached row 1 or your output was invalid. Your program should terminate immediately, and you will receive `Wrong answer`;
   - Otherwise, the drone has moved to cell (r, c), and the next round begins.

If your program continues interacting after reading $(0, 0)$ or $(-1, -1)$, or fails to follow the interaction protocol, you will also receive `Wrong answer`.

After outputting, flush the output buffer; otherwise you may get `Idleness limit exceeded`. For example:

- C/C++: `fflush(stdout)` or `cout.flush()`
- Java: `System.out.flush()`
- Python: `sys.stdout.flush()`

Example

standard input	standard output
5 5 .... #.#.. ..#.. ..... 5 3 4 2 0 0	 4 4 3 2

Note

The interactor can be **adaptive**. That is, the drone's moves are not necessarily fixed before the interaction starts.

Initial obstacles and obstacles you place later remain permanently and never disappear.

The drone never enters a cell containing an obstacle.

You must ensure that no matter how the interactor makes moves, your strategy eventually intercepts the drone.

The sample interaction above proceeds as follows:

- Initially the drone is at $(5, 3)$.

- In round 1, you output `4 4`, placing an obstacle at $(4, 4)$. Now row 4 has obstacles at $(4, 3)$ and $(4, 4)$, so the drone can only move to $(4, 2)$. The interactor then outputs `4 2`.
- In round 2, you output `3 2`, placing an obstacle at $(3, 2)$. Now cells $(3, 1)$, $(3, 2)$, $(3, 3)$ in row 3 are all unreachable. The interactor then outputs `0 0`, indicating you have successfully intercepted the drone.