EGOI '23 P3 - Find the Box

Time limit: 1.0s **Memory limit:** 1G

European Girls' Olympiad in Informatics: 2023 Day 1 Problem 3

Maj is a robotics researcher who works at Lund University. She has learned about a valuable treasure in the cellar of the university. The treasure is in a box located in an empty room deep underground. Unfortunately, Maj cannot just go and look for the box. It is very dark in the cellar and going there with a light would raise suspicion. Her only way to find the treasure is to remotely control a robot vacuum cleaner that inhabits the cellar.

The cellar is represented as an $H \times W$ grid, where the rows are numbered from 0 to $H-1$ (top to bottom) and the columns are numbered from 0 to $W-1$ (left to right), meaning that the top left cell is $(0,0)$ and the bottom right cell is $(H-1, W-1)$. The box with the treasure is in some unknown cell, other than the cell $(0,0)$. Every night, the robot vacuum cleaner starts in the top left corner and moves around the cellar.

Each night, Maj can give the robot a sequence of instructions on how it should move in the form of a string consisting of characters `<`, `>`, `^`, and `v`. Formally, if the robot is standing on cell (r, c) that is unblocked on all sides, `<` moves the robot left to cell $(r, c-1)$, `>` moves the robot right to cell $(r, c+1)$, `^` moves the robot up to cell $(r-1, c)$, and `v` moves the robot down to cell $(r+1, c)$.

The cellar walls are solid, so if the robot attempts to move outside the grid, nothing will happen. The box is also solid, and cannot be pushed. At the end of each night, the robot will report its location and go back to the top left corner.

Time is of the essence, so Maj decides to find the box in as few nights as possible.

Interaction

This is an interactive problem.

- Your program should start by reading a line with two integers H and W: the height and width of the grid.
- Then, your program should interact with the grader. In each round of interaction, you should print a question mark `?` followed by a non-empty string s consisting of characters `<`, `>`, `^`, `v`. The length of this string can be at most $20,000$. Then, your program should read two integers r and c $(0 \le r \le H-1, 0 \le c \le W-1)$, the location of the robot after executing the instructions. Note that the robot always goes back to $(0,0)$ after each query.
- When you know the location of the box, print `!` followed by the two integers r_b, c_b, the row and column of the box $(0 \le r_b \le H-1, 0 \le c_b \le W-1)$. After this, your program must exit without making any further queries. This final output does not count as a query when determining your score.

Make sure to flush standard output after issuing a query, or else your program might get judged as Time Limit Exceeded. In Python, `print()` flushes automatically. In C++, `cout << endl;` also flushes in addition to printing a newline; if using printf, use `fflush(stdout)`.

The grader is non-adaptive, meaning that the position of the box is determined before the interaction begins.

Note that upon receiving invalid input, the interactor will terminate and stop communicating with your submission.

Constraints and Scoring

- $1 \leq H, W \leq 50$.
- The box will never be located at $(0, 0)$. This means that $H + W \geq 3$.
- Each query can consist of at most $20\,000$ instructions.
- You can issue at most $2\,500$ queries (printing the final answer does not count as a query).

Your solution will be tested on a number of test cases. If your solution fails on any of these test cases (e.g. by reporting the wrong box position - Wrong Answer, crashing - Runtime Error, exceeding the time limit - Time Limit Exceeded, etc.), you will receive 0 points and the appropriate verdict.

If your program successfully finds the position of the box in all test cases, you will get the verdict Accepted, and a score calculated as follows:

$$\text{score} = \min\left(\frac{100\sqrt{2}}{\sqrt{Q}}, 100\right) \text{ points},$$

where Q is the maximum number of queries used on any test case. Printing the final answer does not count as a query. The score will be rounded to the nearest integer.

In particular, to receive 100 points, your program must solve every test case using at most $Q = 2$ queries. The table below shows some values of Q and the associated score.

Q	2	3	4	5	...	20	...	50	...	2500
Score	100	82	71	63	...	32	...	20	...	3

Testing Tool

To facilitate the testing of your solution, we provide a simple tool that you can download (testing_tool.py). The tool is optional to use, and you are allowed to change it. Note that the official grader program is different from the testing tool.

Example usage (with $H = 4$, $W = 5$, and the hidden box at position $(r = 2, c = 3)$):

For Python programs, say `solution.py` (normally run as `pypy3 solution.py`): python3 testing_tool.py pypy3 solution.py <<<"4 5 2 3"

For C++ programs, first compile it (e.g., with `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`) and then run: python3 testing_tool.py ./solution.out <<<"4 5 2 3"

Sample Interaction

`>>>` denotes your output. Do not print this out.

```
4 5
>>> ? vv>>>>>><^^^^^>
0 2
>>> ? >>>>>>>>vvvvvvvvvv
3 4
>>> ! 2 3
```

Explanation for Sample Interaction

The grid has height $H = 4$ and width $W = 5$, and the box is at position $(r, c) = (2, 3)$. The figure below shows the robot's path when following the instructions of the first query `? vv>>>>>>^`. It results in the robot ending up at position $(r, c) = (0, 2)$. Before the second query, the robot will go back to the top left corner $(0, 0)$ again. Then the solution issues another query `? >>>>>>>vvvvvvvvvv` for which the robot ends up in the bottom right corner $(r, c) = (3, 4)$. Now the solution decides to guess the answer, by writing `! 2 3`, which is the correct position of the box.