Find the Box

Problem Name	Find the Box
Time Limit	1 seconds
Memory Limit	1 gigabyte

Maj是一名在LTH工作的机器人研究员。她得知大学地下室里有一笔宝贵的宝藏。这个宝藏被放置在一个位于地下空旷房间里的盒子中。然而，Maj不能直接去寻找这个盒子。地下室非常黑暗，带着灯光前去会引起怀疑。她唯一的方法是远程控制地下室里的一台机器人吸尘器来寻找宝藏。

地下室可以看作是一个H×W的网格，其中行从0到H-1编号（从顶部到底部），列从0到W-1编号（从左到右），这意味着左上角的单元格是(0, 0)，右下角的单元格是(H-1, W-1)。盒子与宝藏位于某个未知的单元格中。

每天晚上，机器人吸尘器从左上角开始在地下室中移动。

每天晚上，Maj可以给机器人一系列移动指令，形式为由字符“<”、“>”、“^”和“v”组成的字符串。具体来说，如果机器人站在位置(r, c)上，四周都没有障碍物，“<”表示机器人向左移动到位置(r, c-1)，“>”表示机器人向右移动到位置(r, c+1)，“^”表示机器人向上移动到位置(r-1, c)，“v”表示机器人向下移动到位置(r+1, c)。

地窖的墙壁是坚固的，所以如果机器人试图移动到网格之外，什么也不会发生。这 盒子也很坚固，推不动。每晚结束时，机器人都会报告其位置，并 返回到左上角。 时间至关重要，因此Maj 决定在尽可能短的时间内找到盒子。

交互

这是一个交互问题。 你的程序应该首先读取一行包含两个整数 H 和 W：网格高度和宽度。

然后，您的程序应该与评分器交互。在每一轮互动中，你应该 打印问号“?”，后跟由字符“<”、“>”、“v”，“^”组成的非空字符串 s， 。这个字符串的长度最多可以是20000。那么，你的程序应该读取两个 整数r, c（$0 \le r \le H-1$，$0 \le c \le W-1$），表示执行指令后机器人的位置。请注意，机器人在每次查询后始终返回到(0, 0)。

当你知道盒子的位置时，请打印“！”后面跟着两个整数r_b, c_b，表示盒子所在的行和列（$0 \le r_b \le H-1$，$0 \le c_b \le W-1$）。在此之后，你的程序必须退出，不再进行任何其他查询。这最后的输出在确定你的得分时不计为查询次数。

确保在发出查询后刷新标准输出，否则你的程序可能会被判定为超时。在Python中，print()会自动刷新输出。在C++中，cout << endl; 除了打印换行符外，也会刷新输出；如果使用printf，请使用fflush(stdout)。

这个评分器是非自适应的，这意味着在交互开始之前就确定了盒子的位置。

约束和评分

1 ≤ H,W ≤ 50.

该盒子永远不会位于 (0, 0)。这意味着 H + W ≥ 3。

每个查询最多可以包含 20 000 条指令。

您最多可以发出 2 500 个查询。(打印最后的答案不算是查询）

您的解决方案将在许多测试用例上进行测试。如果您的解决方案在这些测试上的任何失败 情况（例如报告错误的箱子位置（WA）、崩溃（RTE）、超出时间限制（TLE）、 等），您将获得 0 分和相应的判决。如果您的程序成功找到 所有测试用例中框的位置，您将得到 判定AC，得分计算如下：

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

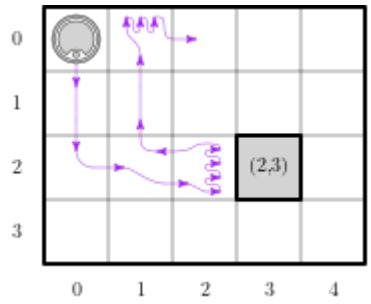
其中 Q 是任何测试用例中使用的最大查询数。打印最终答案 不算是查询。分数将四舍五入到最接近的整数。 特别是，要获得 100 分，您的程序必须最多使用 Q = 2 来解决每个测试用例 查询。下表显示了 Q 的一些值和相关分数。

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

测试工具 为了方便测试你的解决方案，我们提供了一个简单的工具供你下载。请在Kattis问题页面的“附件”中查找。该工具是可选的，你可以对其进行修改。请注意，Kattis上的官方评分程序与测试工具不同。

示例用法（假设H = 4，W = 5，隐藏的盒子位于位置r = 2，c = 3）： 对于Python程序，假设解决方案为solution.py（通常以pypy3 solution.py运行）： python3 testing_tool.py pypy3 solution.py <<< "4 5 2 3" 对于C++程序，首先编译它（例如使用g++ -std=gnu++17 solution.cpp -o solution.out），然后运行：python3 testing_tool.py ./solution.out <<< "4 5 2 3"

示例 考虑样例测试用例。网格的高度H = 4，宽度W = 5，盒子位于位置(r, c) = (2, 3)。下图显示了机器人在执行第一个查询“? vv>>>>>><^^^^>”的指令时的路径，最终机器人停在位置(r, c) = (0, 2)。 在第二个查询之前，机器人会再次回到左上角（0, 0）。然后，解决方案发出另一个查询“? >>>>>>>vvvvvvvvvv”，机器人最终停在右下角(r, c) = (3, 4)。现在，解决方案决定猜测答案，写下“! 2 3”，这是盒子的正确位置。



grader output	your output
4 5	
	? vv>>>>>><^^^^^>
0 2	
	? >>>>>>>>vvvvvvvvv
3 4	
	! 2 3