Find the Box

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

Мей — исследовательница робототехники, работающая в LTH. Она узнала о том, что в подвале университета находятся сокровища. Сокровища находятся в ящике, расположенном в пустой комнате глубоко под землей. К сожалению, Мей не может просто пойти и поискать ящик. В подвале очень темно, а если она пойдёт туда с фонарём, то это вызовет подозрения. Единственный способ найти сокровища — дистанционно управлять роботом-пылесосом, находящимся в подвале.

Рассмотрим подвал как прямоугольную сетку размера $H \times W$, где строки пронумерованы от 0 до $H-1$ (сверху вниз) , а столбцы пронумерованы от 0 до $W-1$ (слева направо), то есть левая верхняя клетка имеет координаты $(0,0)$, а правая нижняя — $(H-1,W-1)$. Координаты клетки, где находится ящик с сокровищем, неизвестны, однако известно что ящик не находится в клетка с координатами $(0,0)$. Каждую ночь робот-пылесос стартует в левом верхнем углу и перемещается по подвалу, выполняя заранее заданную последовательность инструкций.

Каждый вечер Мей даёт роботу последовательность инструкций по перемещению, представленную в виде строки, состоящей из символов "<", ">", "^" и "v". Если робот стоит на клетке (r,c), которая окружена другими пустыми клетками, то "<" перемещает его влево на клетку $(r,c-1)$, ">" перемещает робота вправо на клетку $(r,c+1)$, "^" перемещает робота вверх на клетку $(r-1,c)$, а "v" перемещает робота вниз на клетку $(r+1,c)$.

Если робот пытается выйти за пределы сетки или зайти на клетку с ящиком, то ничего не происходит. В конце каждой ночи робот сообщает свои координаты и возвращается в левый верхний угол.

Время не ждет, поэтому Мей решает найти ящик за как можно меньшее количество ночей.

Interaction

Это интерактивная задача.

- В начале ваша программа должна считать первую строку, которая содержит два целых числа H и W — высоту и ширину сетки соответственно.
- Чтобы задать последовательностей инструкций роботу выведите знак вопроса "`?`", а затем непустую строку s состоящую из символов "`<`", "`>`", "`^`", "`v`". Длина строки не должна превышать $20\,000$. После каждого такого запроса считайте два целых числа r, c ($0 \le r \le H-1$, $0 \le c \le W-1$) — координаты робота после выполнения инструкций. Заметим, что после каждого запроса робот всегда возвращается в точку $(0, 0)$.
- Как только вы узнаете местоположение ящика, выведите "`!`", а затем два целых числа r_b, c_b — координаты ($0 \le r_b \le H-1$, $0 \le c_b \le W-1$). После этого программа должна завершить работу, не делая никаких дополнительных запросов. Это не считается запросом про определении ваших баллов.

Ваше решение получит вердикт «Time Limit Exceeded», если вы не будете ничего выводить или забудете сделать операцию flush после выполнения запроса.

В Python при использовании `print()` flush выполняется автоматически. В C++, используйте `cout << endl;`; если вы используете printf, то используйте `fflush(stdout)`.

Тестирование не является адаптивным, то есть для каждого теста координаты ящика определяются до начала взаимодействия с вашим решением.

Constraints and Scoring

- $1 \le H, W \le 50$.
- Гарантируется, что ящик никогда не находится в начальной позиции робота, $(0, 0)$, что означает что $H + W \ge 3$.
- Длина последовательности инструкций не должна превышать $20\,000$.
- Вы можете задать не более $2\,500$ запросов (вывод ответа не считается запросов).

Ваша программа должна пройти все тесты. Если ваша программа не работает *хотя бы на одном* из тестов (например, неправильные координаты ящика (WA), Runtime Error (RTE), Time Limit Exceeded (TLE) и т.д.), вы получите 0 баллов и соответствующий вердикт.

Если ваша программа находит координаты ящика для *всех* тестов, вы получите вердикт AC, количество ващих баллов будет определено следующим образом:

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

где Q — это *максимальное* количество запросов по всем тестам. Вывод ответа не считается запросом при подсчёте баллов. `score` будет округлён до ближайшего целого числа.

Чтобы получить 100 баллов, ваша программа должна пройти все тесты, используя не более $Q = 2$ запросов. В таблице показаны некоторые значения Q и соответствующее им количество баллов.

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

Testing Tool

Чтобы упростить процесс вашего решения, вы можете скачать `testing_tool`. На странице задачи в системе Kattis в самом низу в разделе “attachments”. Это опциональная `testing_tool` и вы можете менять её. Обратите внимание, что программа, которая используется в тестирующей системе Kattis отличается от этой `testing_tool`.

Пример использования (для $H = 4$, $W = 5$, и ящика с координатами $r = 2$, $c = 3$):

Для python. Допустим, что ваша программа называется `solution.py` (обычно вы запускаете её так: `pypy3 solution.py`):

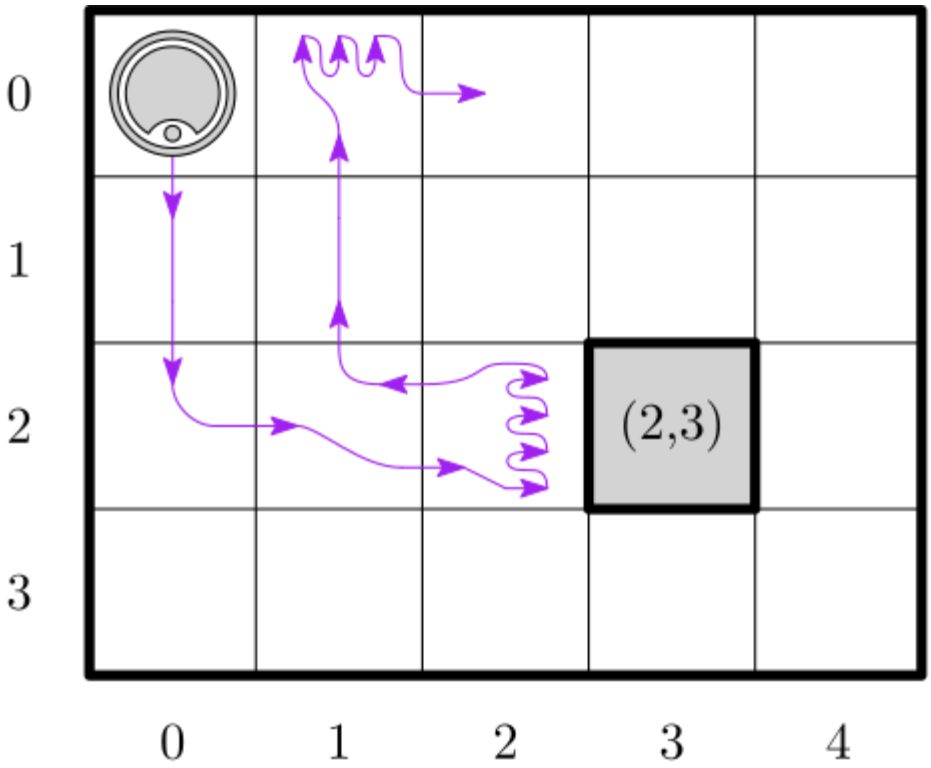
```
python3 testing_tool.py pypy3 solution.py <<<"4 5 2 3"
```

Для C++, сначала, скомпилируйте (например так: `g++ -std=gnu++17 solution.cpp -o solution.out`), после этого исполните:

```
python3 testing_tool.py ./solution.out <<<"4 5 2 3"
```

Example

Рассмотрим пример для $H = 4$, $W = 5$, и ящика с координатами $r = 2$, $c = 3$. На рисунке изображён путь робота, исполняющего последовательность инструкций для первого запроса “`? vv>>>>>><^^^^^>`”, который закончится клетке с координатами $(r, c) = (0, 2)$. Перед вторым запросом, робот вернётся обратно в верхнюю левую клетку с координатами $(0, 0)$. После этого решение выведет следующий запрос: “`? >>>>>>>>vvvvvvvvvv`”, после которого робот окажется в правом нижнем углу с координатами $(r, c) = (3, 4)$. После этого программа выводит ответ: “`! 2 3`”, который является верным.



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