Find the Box

Nombre del problema	Find the Box
Límite de tiempo	1 segundo
Límite de memoria	1 gigabyte

Maj es una investigadora en robótica que trabaja en la LTH. Ella se ha enterado que hay un valioso tesoro en el sótano de la universidad. El tesoro está en una caja localizada en una profunda sala vacía bajo tierra. Desafortunadamente, Maj no puede simplemente ir y buscar la caja. El sótano está muy oscuro y llevar una linterna levantaría sospechas. Su única forma de encontrar el tesoro es controlar remotamente un robot aspirador que se encuentra en el sótano.

El sótano se representa con una cuadrícula de dimensiones $H \times W$, donde las filas se numeran de 0 a $H-1$ (de arriba a abajo) y las columnas se numeran de 0 a $W-1$ (de izquierda a derecha), es decir, la celda de arriba a la izquierda será $(0,0)$ y la celda de abajo a la izquierda $(H-1, W-1)$. La caja con el tesoro está en una celda desconocida, distinta a la celda $(0,0)$. Cada noche, el robot aspirador empieza arriba a la izquierda y se mueve por el sótano.

Cada noche, Maj puede darle al robot una secuencia de instrucciones sobre cómo debe moverse en forma de un string usando los caracteres "<", ">", "^" and "v". la celda Formalmente, si el robot está en la celda (r,c) y no está bloqueado en ninguna dirección, "<" mueve el robot a la celda $(r, c-1)$, ">" mueve el robot a la celda $(r, c+1)$, "^" mueve el robot a la celda $(r-1, c)$, y "v" mueve el robot a la celda $(r+1, c)$.

Los muros del sótano son sólidos, por lo que si el robot intenta moverse hacia fuera de la cuadrícula, nada sucederá. La caja también es sólida, y no se puede empujar. Al final de cada noche, el robot reportará su localización, y volverá a la esquina superior izquierda.

El tiempo es esencial, por lo que Maj decide buscar la caja en el mínimo número de noches posible.

Interacción

Este es un problema interactivo.

- Tu programa debe empezar leyendo una línea con dos enteros H y W: el alto y el ancho de la cuadrícula.

- Después, tu programa debe interactuar con el grader. En cada ronda de interacción, debes imprimir un interrogante "`?`", seguido de un string no vacío s que solo contenga los caracteres "`<`", "`>`", "`^`", "`v`". La longitud del string puede ser como máximo $20\,000$. Luego, tu programa debe leer dos enteros r, c ($0 \le r \le H-1$, $0 \le c \le W-1$), la localización del robot tras ejecutar las instrucciones. Recuerda que el robot siempre vuelve a la posición $(0, 0)$ tras cada interacción.
- Cuando conozcas la posición de la caja, imprime "`!`" seguido de dos enteros r_b, c_b, la fila y la columna donde se encuentra la caja ($0 \le r_b \le H-1$, $0 \le c_b \le W-1$). Tras esto, tu programa debe finalizar sin realizar ninguna otra consulta. Este output final no cuenta como ronda a la hora de calcular tu puntuación.

Asegúrate de hacer flush tras cada consulta, o tu programa podría recibir el veredicto *Time Limit Exceeded*. En Python, `print()` hace flush automáticamente. En C++, `cout << endl;` también hace flush además de imprimir un salto de línea; si usas printf, usa `fflush(stdout)`.

El grader no es adaptativo, es decir, la posición de la caja se decide antes de que empiece la interacción.

Restricciones y Puntuación

- $1 \le H, W \le 50$.
- La caja nunca estará en $(0, 0)$. Esto implica que $H + W \ge 3$.
- Cada consulta puede contener como máximo $20\,000$ instrucciones.
- Puedes realizar como máximo $2\,500$ consultas (imprimir la respuesta final no cuenta como consulta).

Tu solución será puesta a prueba en distintos casos de prueba. Si to solución falla en *cualquiera* de estos casos de prueba (por ejemplo dando una posición de la caja errónea (Wrong Answer), crasheando (Runtime Error), excediendo el límite de tiempo (Time Limit Exceeded), etc.), recibirás 0 puntos y el veredicto apropiado.

Si tu programa encuentra la posición de la caja satisfactoriamente en \emph{todos} los casos de prueba, recibirás el veredicto *AC*, y una puntuación calculada de la siguiente manera:

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

donde Q el el *máximo* número de consultas usadas en cualquiera de los casos de prueba. Imprimir la respuesta final no cuenta como consulta. La puntuación se redondea al entero más cercano.

En particular, para recibir 100 puntos, tu programa debe resolver todos los casos de prueba usando como máximo $Q = 2$ queries. La tabla de debajo muestra algunos valores de Q y la puntuación que tendrías con este número de consultas.

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

Herramienta de testeo

Para facilitar el testeo de tu solución, tienes una sencilla herramienta que puedes descargar. Ver “attachments” al final de la página de Kattis del problema. El uso de la herramienta es opcional, y puedes modificarla. Ten en cuenta que el grader oficial en Kattis es distinto a la herramienta de testeo.

Ejemplo de uso (con $H = 4$, $W = 5$, y la caja oculta en la posición $r = 2$, $c = 3$):

Para programas en Python, digamos `solution.py` (habitualmente ejecutado con `pypy3 solution.py`):

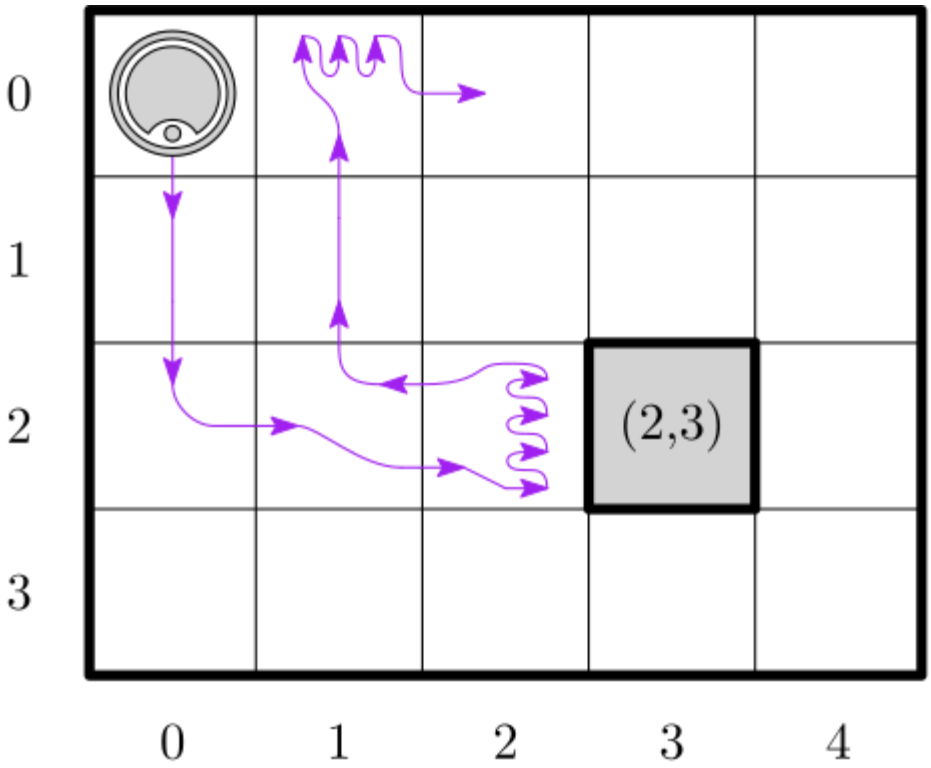
```
python3 testing_tool.py pypy3 solution.py <<<"4 5 2 3"
```

Para programas en C++, primero lo compilamos (por ejemplo con `g++ -std=gnu++17 solution.cpp -o solution.out`) y luego ejecutamos:

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

Test case de ejemplo

Coinsidera el test case de ejemplo. La cuadrícula tiene altura $H = 4$ y anchura $W = 5$, y la caja está oculta en la posición $(r, c) = (2, 3)$. La imagen de debajo muestra el camino del robot cuando sigue las instrucciones de la primera consulta “`? vv>>>>>><^^^^^>`”, que resulta en la posición final del robot $(r, c) = (0, 2)$. Antes de la segunda consulta, el robot vuelve a la posición inicial $(0, 0)$ de nuevo. Entonces la solución lleva a cabo otra consulta “`? >>>>>>>>vvvvvvvvvv`” inferior derecha $(r, c) = (3, 4)$. Finalmente la solución decide dar una respuesta, escribiendo “`! 2 3`”, la cuál es correcta ya que coincide con la posición de la caja.



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