Ache a Caixa

Nome do Problema	Ache a Caixa
Limite de Tempo	1 segundo
Limite de Memória	1 gigabyte

Maj é uma pesquisadora de robótica que trabalha na Universidade de Lund. Ela ficou sabendo sobre um tesouro valioso no porão da universidade. O tesouro está em uma caixa localizada em uma sala vazia no subsolo. Infelizmente, Maj não pode simplesmente ir procurar a caixa. Está muito escuro no porão e ir até lá com uma lanterna levantaria suspeitas. Sua única maneira de encontrar o tesouro é controlar remotamente um robô aspirador de pó que habita o porão.

O porão é representado como um *grid* $H \times W$, no qual as linhas são numeradas de 0 a $H-1$ (de cima para baixo) e as colunas são numeradas de 0 a $W-1$ (da esquerda para a direita), o que significa que a célula do canto superior esquerdo é $(0,0)$ e a célula do canto inferior direito é $(H-1, W-1)$. A caixa com o tesouro está em uma célula desconhecida que não seja a célula $(0,0)$. Todas as noites, o robô aspirador de pó começa no canto superior esquerdo e se move pelo porão.

A cada noite, Maj pode dar ao robô uma sequência de instruções sobre como ele deve se mover na forma de uma *string* com os caracteres "<", ">", "^" and "v". Formalmente, se o robô estiver parado na célula (r, c) que está desbloqueada em todos os lados, "<" move o robô para a esquerda, para a célula $(r, c-1)$, ">" move o robô para a direita, para a célula $(r, c+1)$, "^" move o robô para cima, para a célula $(r-1, c)$, e "v" move o robô para baixo, para a célula $(r+1, c)$.

As paredes do porão são sólidas, então se o robô tentar se mover para fora do *grid*, nada acontecerá. A caixa também é sólida e não pode ser empurrada. No final de cada noite, o robô informará sua localização e voltará para o canto superior esquerdo.

Como o tempo é essencial, Maj decide encontrar a caixa no menor número possível de noites.

Interação

Esse é um problema interativo.

- Seu programa deve começar lendo uma linha com dois números inteiros H e W: a altura e a largura do *grid*.

- Em seguida, seu programa deve interagir com o corretor. Em cada rodada de interação, você deve imprimir um ponto de interrogação "`?`", seguido de uma *string* não-vazia s que consiste nos caracteres "`<`", "`>`", "`^`", "`v`". O tamanho dessa *string* pode ser de no máximo $20\,000$. Em seguida, seu programa deve ler dois números inteiros r, c ($0 \le r \le H - 1$, $0 \le c \le W - 1$), a localização do robô após a execução das instruções. Note que o robô sempre volta para $(0, 0)$ após cada consulta.
- Quando você descobrir a localização da caixa, imprima "`!`" seguido dos dois números inteiros r_b, c_b, a linha e a coluna da caixa ($0 \le r_b \le H - 1$, $0 \le c_b \le W - 1$). Depois disso, seu programa deve encerrar sem fazer nenhuma outra consulta. Essa saída final não conta como uma consulta ao determinar sua pontuação.

Certifique-se de dar *flush* na saída padrão após imprimir uma consulta, caso contrário, seu programa poderá ser julgado como *Time Limit Exceeded*. Em Python, `print()` faz o *flush* automaticamente. Em C++, `cout << endl;` também faz o *flush* além de imprimir uma nova linha; se estiver usando printf, use `fflush(stdout)`.

O corretor não é adaptativo, o que significa que a posição da caixa é determinada antes do início da interação.

Restrições e Pontuação

- $1 \le H, W \le 50$.
- A caixa nunca estará localizada em $(0, 0)$. Isso significa que $H + W \ge 3$.
- Cada consulta pode consistir em no máximo $20\,000$ instruções.
- Você pode imprimir no máximo $2\,500$ consultas (imprimir a resposta final não conta como uma consulta).

Sua solução será testada em vários casos de teste. Se a sua solução falhar em *qualquer* desses casos de teste (por exemplo: informando a posição errada da caixa (*Wrong Answer*), encerrando abruptamente (*Runtime Error*), excedendo o limite de tempo (*Time Limit Exceeded*), etc.), você receberá 0 pontos e o veredito apropriado.

Se seu programa encontrar com sucesso a posição da caixa em *todos* os casos de teste, você obterá o veredito *Accepted* e a pontuação calculada da seguinte forma:

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

onde Q é o número máximo de consultas usadas em qualquer caso de teste. Imprimir a resposta final não conta como uma consulta. A pontuação será arredondada para o número inteiro mais próximo.

Em particular, para receber 100 pontos, seu programa deve resolver todos os casos de teste usando no máximo $Q = 2$ consultas. A tabela abaixo mostra alguns valores de Q e a pontuação correspondente.

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

Corretor Exemplo

Para facilitar o teste de sua solução, fornecemos um corretor exemplo que você pode baixar. Consulte “attachments” (anexos) na parte inferior da página do problema no Kattis. O uso do corretor exemplo é opcional e você tem permissão para alterá-lo. Observe que o corretor oficial no Kattis é diferente do corretor exemplo.

Exemplo de uso (com $H = 4$, $W = 5$ e a caixa escondida na posição $r = 2$, $c = 3$):

Para programas python, por exemplo, `solution.py` (normalmente executado como `pypy3 solution.py`):

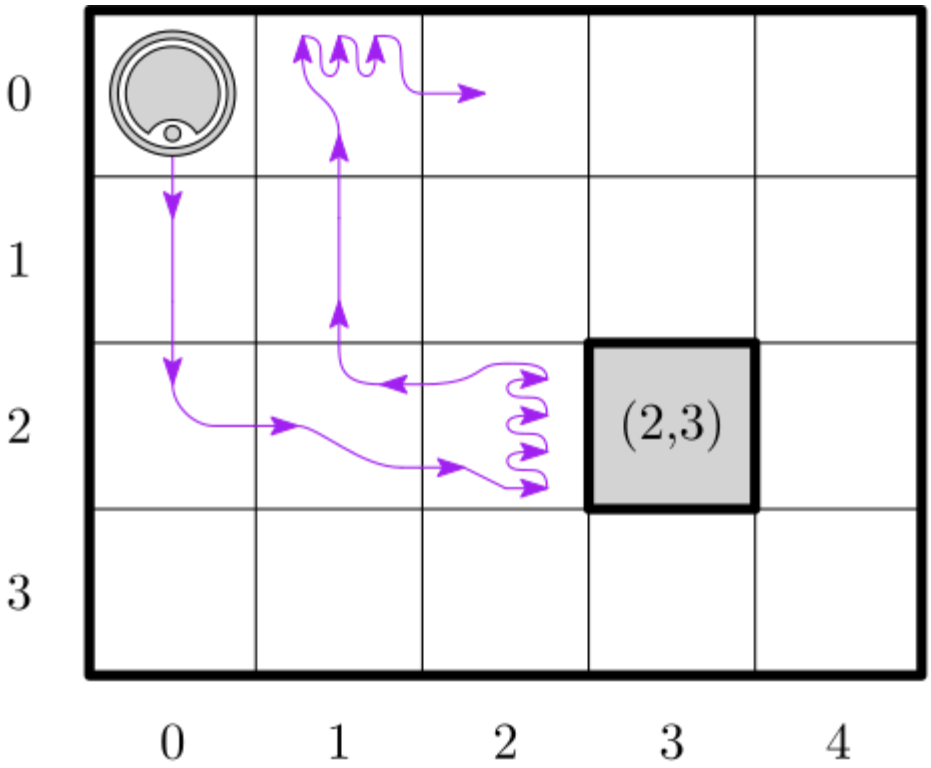
```
python3 testing_tool.py pypy3 solution.py <<<"4 5 2 3"
```

Para programas C++, primeiro compile-o (por exemplo, com `g++ -std=gnu++17 solution.cpp -o solution.out`) e, em seguida, execute:

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

Exemplo

Considere o caso de teste de exemplo. O *grid* tem altura $H = 4$ e largura $W = 5$, e a caixa está na posição $(r, c) = (2, 3)$. A figura abaixo mostra o caminho do robô ao seguir as instruções da primeira consulta “`? vv>>>>>><^^^^>`”, o que faz com que o robô termine na posição $(r, c) = (0, 2)$. Antes da segunda consulta, o robô voltará para o canto superior esquerdo $(0, 0)$ novamente. Em seguida, a solução faz outra consulta “`? >>>>>>>>vvvvvvvvvv`” para a qual o robô termina no canto inferior direito $(r, c) = (3, 4)$. Agora a solução decide adivinhar a resposta, imprimindo “`! 2 3`”, que é a posição correta da caixa.



saída do corretor	sua saída
4 5	
	? vv>>>>>><^^^^^>
0 2	
	? >>>>>>>>vvvvvvvvv
3 4	
	! 2 3