Hitta Lådan

Problemnamn	Find the Box
Tidsgräns	1 sekund
Minnesgräns	1 gigabyte

På LTH jobbar en robotforskare, Maj, som har blivit besatt. I universitets källare har hon lärt sig att det finns en värdefull skatt. Skatten finns i en låda i ett tomt rum långt långt ner. Men Maj kan inte själv gå dit, eftersom det är för mörkt för att se. Att ta med ett ljus skulle verka suspekt. Hon kontrollerar en robotdammsugare (som bor i källaren) för att uppnå samma effekt.

Källaren är representerad som en $H \times W$ rutnät där raderna är numrerade från 0 to $H-1$ (uppifrån och ner) och kolumnerna är numrerade från 0 to $W-1$ (vänster till höger). Detta innebär att $(0,0)$ är cellen högst upp till vänster och att $(H-1, W-1)$ är cellen längst ner till höger. Skatten finns inte i $(0,0)$ men förutom det vet ingen vet vilken cell som innehåller denna vackra skatt. Men robotdammsugaren börjar i det högsta vänstra hörnet och åker runt i källaren för sig själv varje natt.

Maj ger roboten en sekvens av instruktioner för hur den ska flytta på sig vardera dag. Detta sker i formen av en sträng som innehåller karaktärerna "<", ">", "^" och "v" vilket ger robotdammsugaren sitt uppdrag.

Formellt, om roboten står på cell (r, c) som är inte är blockerad på någon sida flyttar "<" roboten vänster till cell $(r, c-1)$, ">" flyttar roboten höger till cell $(r, c+1)$, "^" flyttar roboten upp till cell $(r-1, c)$, och "v" flyttar roboten ner till cell $(r+1, c)$.

Källarväggarna är väldigt solida, så om roboten försöker att åka ut från rutnätet så händer ingenting. Skattens låda är också solid, och kan inte flyttas kring. Vid varje natts slut, rapporterar roboten sin position, och åker sedan tillbaka till hörnet som är högst upp och längst vänster ut. Tiden är av ytterst vikt, så Maj vill hitta lådan på så få nätter som möjligt och få sitt uppdragsavlsut.

Interaktion

Det här är ett interaktivt problem.

- Ditt program, ska börja med att läsa en rad med två heltal H och W: höjden och bredden av rutnätet. Lådan kommer aldrig att vara på robotens startposition $(0,0)$.

- Sen ska ditt program interagera med domaren. I varje runda av interaktionen ska du skriva ut ett frågetecken "`?`", följt av en icke-tom sträng s som innehåller karaktärerna "`<`", "`>`", "`^`", "`v`". Längden av strängen kan vara som mest $20\,000$. Sen så ska ditt program läsa in två heltal r, c ($0 \leq r \leq H-1$, $0 \leq c \leq W-1$), positionen av roboten, efter att den följt instruktionen. Notera att roboten åker tillbaka till $(0, 0)$ efter varje instruktion.
- När du vet lådans position, skriv ut "`!`" följt av två heltal r_b, c_b, raden och kolumnen där lådan finns ($0 \leq r_b \leq H-1$, $0 \leq c_b \leq W-1$). Efter det här ska ditt program avslutas utan att skriva ut fler instruktioner. Den sista instruktionen räknas inte som en fråga när dina poäng avgörs.

Kom ihåg att använda flush standard output efter du skrivit ut en instruktion, annars kan ditt program bedömmas som Time Limit Exceeded (tidsgräns överskriden). I Python, så kommer `print()` använda flush automatiskt. I C++ så kommer `cout << endl;` also att använda flush och dessutom göra en radbrytning; om du använder printf, så använd `fflush(stdout)`.

Domaren är icke-adaptiv, vilket innebär att positionen av lådan är bestämd före interaktionen börjar.

Begränsningar och betygsättning

- $1 \leq H, W \leq 50$.
- Lådan kommer aldrig att vara på cell $(0, 0)$. Det här betyder att $H + W \geq 3$.
- Varje fråga kan bestå av som mest $20\,000$ instruktioner.
- Du kan ställa som mest $2\,500$ frågor.

Din lösning kommer att testas på en mängd testfall. Om din lösning misslyckas på *något* av dessa testfall (t ex genom att rapportera fel position för lådan (WA), krascha (RTE), överskrida tidsgränsen (TLE) osv.) kommer du att få 0 poäng och den lämpliga domen.

Om ditt program lyckas hitta rätt position på lådan i *alla* testfall så kommer du få domen (AC) och ditt betyg kommer att beräknas på följande sätt:

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

där Q är det *maximala* antalet frågor som används på något testfall. Att skriva ut det slutgiltiga svaret räknas inte som en fråga. Dina poäng kommer att avrundas till närmaste associativa heltal.

I synnerhet, för att få 100 poäng så måste ditt program lösa alla testfall genom att använda som mest $Q = 2$ frågor. Tabellen nedan visar några värden av Q och dess associerade poängsättning.

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

Testverktyg

För att förenkla testandet av din lösning så har vi skapat ett enkelt verktyg som du kan ladda ner. Se "bilagor" på botten av Kattisproblemsidan. Det är frivilligt att använda verktyget. Notera at den officiella domarprogramet på Kattis är annorlunda från testprogramet.

Exempel användning (med $H = 4$, $W = 5$, och den gömda lådan på position $r = 2$, $c = 3$):

För python program, säg `solution.py` (normallt körd som `pypy3 solution.py`):

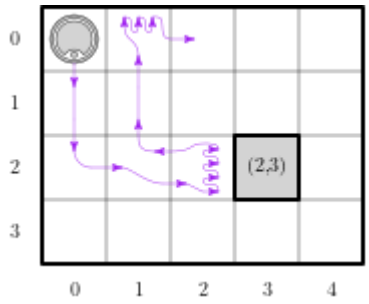
```
python3 testing_tool.py pypy3 solution.py <<<"4 5 2 3"
```

For C++ program, kompilera det först (t ex med `g++ -std=gnu++17 solution.cpp -o solution.out`) och kör sen:

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

Exempel

Fundera på exempel testfallet. Rutnätet har höjd $H = 4$ och bredd $W = 5$, och lådan är på position $(r, c) = (2, 3)$. Figuren nedan visar robotens väg när den följer instruktionerna på första frågan "`? vv>>>>>><^^^^^>`", vilket resulterar i att roboten slutar vid positionen $(r, c) = (0, 2)$. Innan den andra frågan kommer roboten att gå tillbaka till det högsta vänstra hörnet $(0, 0)$ igen. Sen så kommer din lösning ställa en till fråga. "`? >>>>>>>>vvvvvvvvvv`" där din robot slutar i det lägsta högra hörnet $(r, c) = (3, 4)$. Nu så kommer lösningen gissa på svaret genom att skriva "`! 2 3`", vilket är den korrekta positionen av lådan.



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