Finn boksen

Oppgavenavn	Find the Box
Tidsbegrensning	1 sekund
Minnebegrensning	1 gigabyte

Maj arbeider som robotforsker ved LTH. Hun har hørt om en verdifull skatt i kjelleren til universitetet. Skatten er i en boks i et tomt rom dypt under bakken. Dessverre kan ikke Maj bare gå og lete etter boksen. Det er veldig mørkt i kjelleren, og å gå med en lykt vil vekke mistanke. Hennes eneste mulighet til å finne skatten er å fjernstyre en robotstøvsuger som befinner seg i kjelleren.

Kjelleren er representert av et $H \times W$ rutenett, hvor radene er nummerert fra 0 til $H-1$ (topp til bunn) og kolonnene fra 0 til $W-1$ (venstre til høyre). Den øverste cellen til venstre er da $(0,0)$ og den nederst til høyre $(H-1, W-1)$. Boksen med skatten er i en ukjent celle. Hver natt starter robotstøvsugeren i det øverste venstre hjørnet og beveger seg rundt i kjelleren.

Hver natt kan Maj gi roboten en sekvens instrukser om hvordan den skal bevege seg i form av en streng bestående av tegnene "<", ">", "^" og "v". Formelt sett betyr dette at hvis roboten står på cellen (r, c) som er ublokkert på alle sider, vil "<" bevege roboten til venstre til celle $(r, c-1)$, ">" bevege roboten til høyre til celle $(r, c+1)$, "^" bevege roboten opp til celle $(r-1, c)$, og "v" vil bevege roboten ned til celle $(r+1, c)$.

Kjellerveggene er statiske, så om roboten prøver å bevege seg ut av rutenettet, vil ingenting skje. Boksen er også statisk og kan ikke dyttes. På slutten av natten vil roboten oppgi sin posisjon, og returnere til det øverste venstre hjørnet.

Tiden er dyrebar, så Maj ønsker å finne boksen på så få netter som mulig.

Interaksjon

Dette er en interaktiv oppgave.

- Programmet ditt skal starte med å lese heltallene H og W: høyden og bredden til rutenettet.

- Etter dette skal programmet ditt interagere med *graderen*. I hver runde med interaksjon skal du skrive et spørsmålstegn "`?`" etterfulgt av en ikke-tom streng s bestående av tegnene "`<`", "`>`", "`^`", "`v`". Lengden til strengen kan høyst være $20\,000$. Etter dette skal programmet ditt lese to heltall r, c ($0 \leq r \leq H-1$, $0 \leq c \leq W-1$), posisjonen til roboten etter gjennomføringen av instruksene. Merk at roboten går tilbake til $(0, 0)$ etter hver forespørsel.
- Når du vet posisjonen til boksen, print "`!`", etterfulgt av to heltall r_b, c_b, raden og kolonnen til boksen ($0 \leq r_b \leq H-1$, $0 \leq c_b \leq W-1$). Etter dette må prgrammet slutte uten å gjøre flere forespørsler. Det endelige output-et teller ikke som en handling ved bestemmelsen av poengsummen din.

Forsikre deg om å flushe standard output etter hver spørring, ellers kan løsningen din klassifiseres som Time Limit Exceeded. I Python flusher `print()` automatisk. I C++ flusher `cout << endl;` også, sammen med å printe en ny linje. Anvender du printf, bruk `fflush(stdout)`.

Retteprogrammet (engelsk: *graderen*) er ikke-adaptiv, noe som betyr at boksens posisjon er bestemt før interaksjonen begynner.

Begrensninger og poenggiving

- $1 \leq H, W \leq 50$.
- Boksen vil aldri ha posisjonen $(0, 0)$. Det betyr at $H + W \geq 3$.
- Hver forespørsel kan bestå av høyst $20\,000$ instruksjoner.
- Du kan ikke sende flere enn $2\,500$ forespørsler.

Løsningen din vil bli testet på flere tester. Hvis løsningen din feiler på *noen* av testene (for eksempel ved å rapportere feil posisjon til boksen (WA), å crashe (RTE), å gå utover tidsbegrensningen (TLE), osv.) mottar du 0 poeng.

Om programmet ditt korrekt identifiserer boksens posisjon i *alle* testene, får du en godkjent løsning (AC) og en score beregnet på følgende måte:

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

hvor Q er det *største* antallet forespørsler brukt på noen av testene. Å printe det endelige svaret teller ikke som en forespørsel. Scoren rundes av til nærmeste heltall.

For å motta 100 poeng må programmet ditt løse alle testcaser ved å bruke maks $Q = 2$ forespørsler. Tabellen under viser noen verdier av Q og tilhørende score.

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

Testverktøy

For å tilrettelegge for testing av løsningen din, har vi gjort tilgjengelig et enkelt verktøy for nedlasting. Se "attachments" på bunnen av oppgavesiden på Kattis. Dette verktøyet er frivillig å bruke, og du står fritt til å endre på programmet. Merk at den offisielle *graderen* på Kattis er et annet program.

Eksempelbruk (med $H = 4$, $W = 5$, og den skjulte boksen i posisjon $r = 2$, $c = 3$):

For pythonprogrammer, si `solution.py` (normalt kjørt som `pypy3 solution.py`):

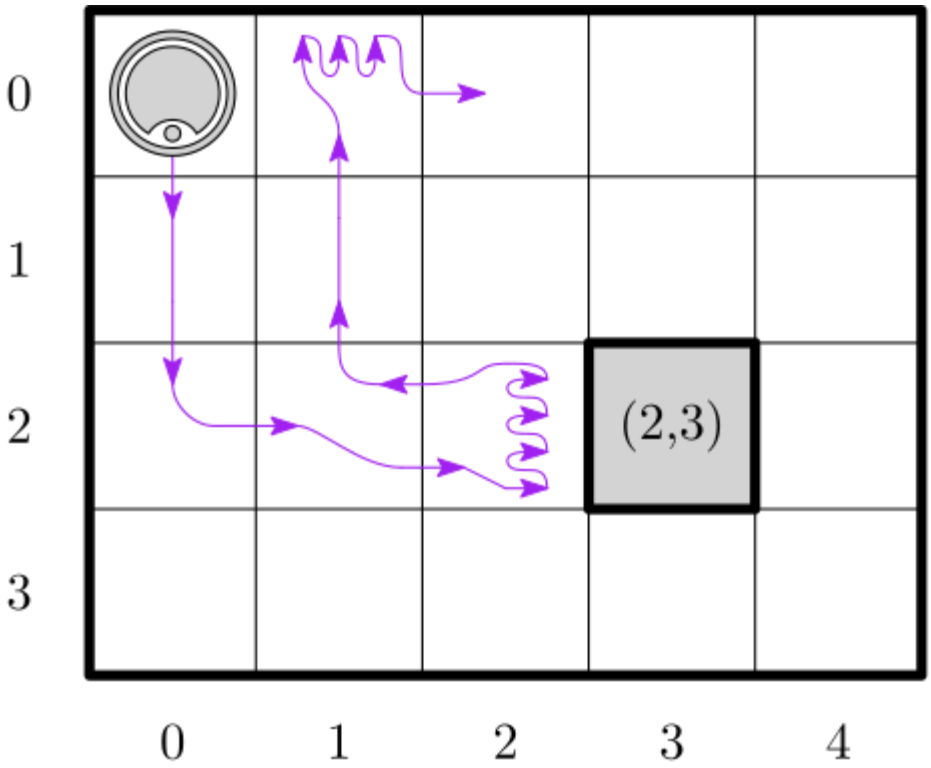
```
python3 testing_tool.py pypy3 solution.py <<<"4 5 2 3"
```

For C++-programmer, kompiler først (for eks. med `g++ -std=gnu++17 solution.cpp -o solution.out`) og kjør så:

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

Eksempel

Betrakt eksempeltesten under. Rutenettet har høyde $H = 4$ og bredde $W = 5$, og boksen er plassert i posisjon $(r, c) = (2, 3)$. Figuren under viser robotens sti når den følger instruksene fra den første forespørselen "`? vv>>>>>><^^^^^>`", som resulterer i at roboten ender opp i posisjon $(r, c) = (0, 2)$. Før den andre forespørselen vil roboten gå tilbake til hjørnet øverst til venstre $(0, 0)$ igjen. Så skriver løsningen en ny forespørsel "`? >>>>>>>>vvvvvvvvvv`" hvor roboten ender opp i det nedre høyre hjørnet $(r, c) = (3, 4)$. Nå bestemmer løsningen seg for å gjette svaret, ved å skrive "`! 2 3`", som er den eksakte posisjonen til boksen.



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