Robot Contest

Cercetători de IA de la Universitatea din Szeged găzduiesc un concurs de programare pentru roboți. Prietenul tău Hanga a decis să participe la concurs. Scopul concursul este să programeze cel mai tare *Pulibot*, în admirarea profundei inteligențe a faimoasei rase de caini ciobănești Maghiari, Puli.

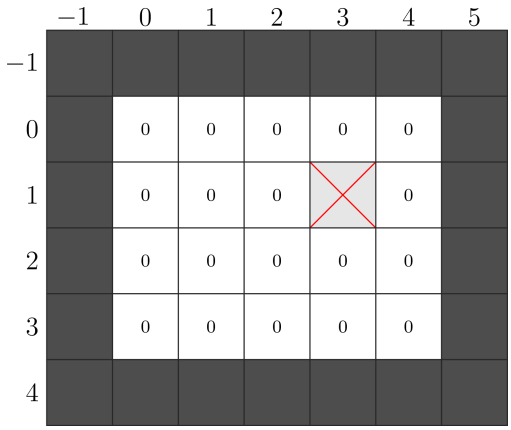
Pulibotul va fi testat pe un labirint format dintr-un grid de $(H+2) \times (W+2)$ celule. Rândurile gridului sunt numerotate de la -1 la H din nord spre sud, iar coloanele gridului sunt numerotate de la -1 la W din vest spre est. Ne referim la celula din rândul r și coloana c al gridului ($-1 \leq r \leq H$, $-1 \leq c \leq W$) ca fiind celula (r, c).

Considerăm o celulă (r, c) pentru care $0 \leq r < H$ și $0 \leq c < W$. Există 4 celule **adiacente** celulei (r, c):

- celula $(r, c-1)$ este celula la **vest** de celula (r, c);
- celula $(r+1, c)$ este celula la **sud** de celula (r, c);
- celula $(r, c+1)$ este celula la **est** de celula (r, c);
- celula $(r-1, c)$ este celula la **nord** de celula (r, c).

Celula (r, c) se numește celulă de **bordură** dacă $r = -1$ sau $r = H$ sau $c = -1$ sau $c = W$. Fiecare celulă care nu este o celulă de bordură este fie un **obstacol** fie o celulă **goală**. Mai mult, fiecare celulă goală are o **culoare**, reprezentată de un întreg nenegativ între 0 și Z_{MAX} inclusiv. Inițial, culoarea fiecărei celule este 0.

De exemplu, considerați un labirint cu $H = 4$ și $W = 5$, conținând o singură celulă obstacol $(1, 3)$.



Singurul obstacol este marcat cu o cruce. Celulele de bordură sunt umplute. Numerele scrise în fiecare celulă îi reprezintă culoarea.

Un **drum** de lungime ℓ ($\ell > 0$) din celula (r_0, c_0) la celula (r_ℓ, c_ℓ) este o secvență de celule *goale* distincte două câte două $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ in care pentru fiecare i ($0 \leq i < \ell$) celulele (r_i, c_i) și (r_{i+1}, c_{i+1}) sunt adiacente.

Observați că un drum de lungime ℓ conține exact $\ell + 1$ celule.

La concurs, cercetătorii au pregătit un labirint în care există cel puțin un drum de la celula $(0, 0)$ la celula $(H - 1, W - 1)$. Observați că acest fapt presupune că celulele $(0, 0)$ și $(H - 1, W - 1)$ sunt garantat goale.

Hanga nu știe care dintre celulele labirintului sunt goale și care dintre celule sunt obstacole.

Misiunea voastră este să îl ajutați pe Hanga să îl programeze pe Pulibot astfel încât să fie capabil să găsească un *drum cel mai scurt* (adică, un drum de lungime minimă) de la celula $(0, 0)$ la celula $(H - 1, W - 1)$ în labirintul necunoscut pregătit de cercetători. Specificațiile lui Pulibot și regulile concursului sunt descrise mai jos.

Vă rugăm să luați la cunoștință că ultima secțiune a acestei probleme descrie o unealtă de vizualizare ce o puteți folosi să îl vizualizați pe Pulibot.

Specificațiile lui Pulibot

Definim **starea** unei celule (r, c) pentru fiecare $-1 \leq r \leq H$ și $-1 \leq c \leq W$ ca un întreg astfel încât:

- dacă celula (r, c) este o bordură atunci starea ei este -2;
- dacă celula (r, c) este un obstacol atunci starea ei este -1;
- dacă celula (r, c) este goală atunci starea ei este culoarea celulei.

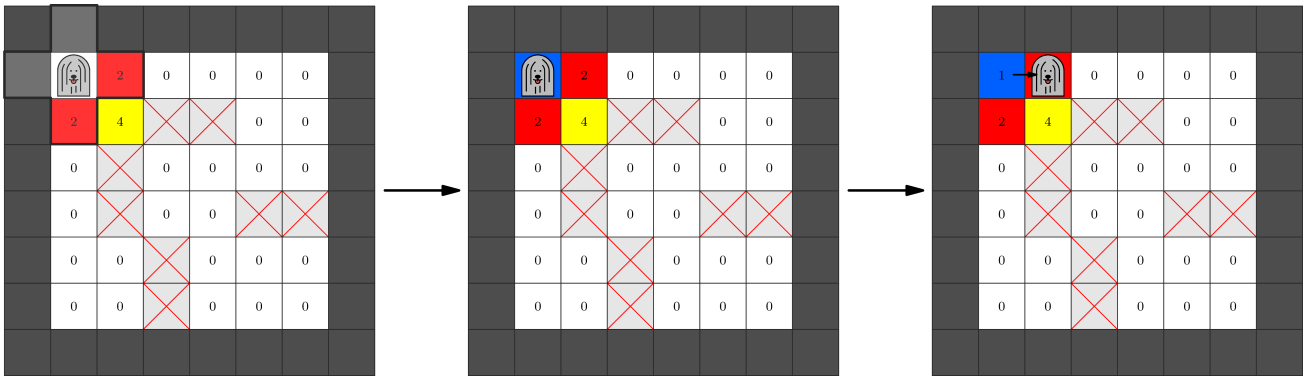
Programul lui Pulibot este executat ca o secvență de pași. În fiecare pas, Pulibot recunoaște starea celulor de lângă el și apoi execută o instrucțiune. Instrucțiunea executată este determinată de stările recunoscute. O descriere mai exactă urmează.

Să presupunem că la începutul pasului curent, Pulibot este la celula (r, c), care este o celulă goală. Pasul se va petrece în felul următor:

1. Inițial, Pulibot recunoaște **șirul de stare** curent, adică, șirul $S = [S[0], S[1], S[2], S[3], S[4]]$ ce conține starea celulei (r, c) și a tuturor celulelor adiacente:
    - $S[0]$ este starea celulei (r, c).
    - $S[1]$ este starea celulei la vest.
    - $S[2]$ este starea celulei la sud.
    - $S[3]$ este starea celulei la est.
    - $S[4]$ este starea celulei la nord.
2. Apoi, Pulibot determină **instrucțiunea** (Z, A) care corespunde șirului de stare recunoscut.

3. În cele din urmă, Pulibot execută acea instrucțiune: setează culoarea celulei (r, c) la culoarea Z, și apoi execută acțiunea A, care este una dintre următoarele:
    - *stai* pe celula (r, c);
    - *treci pe* una dintre cele 4 celule adiacente;
    - *termină programul*.

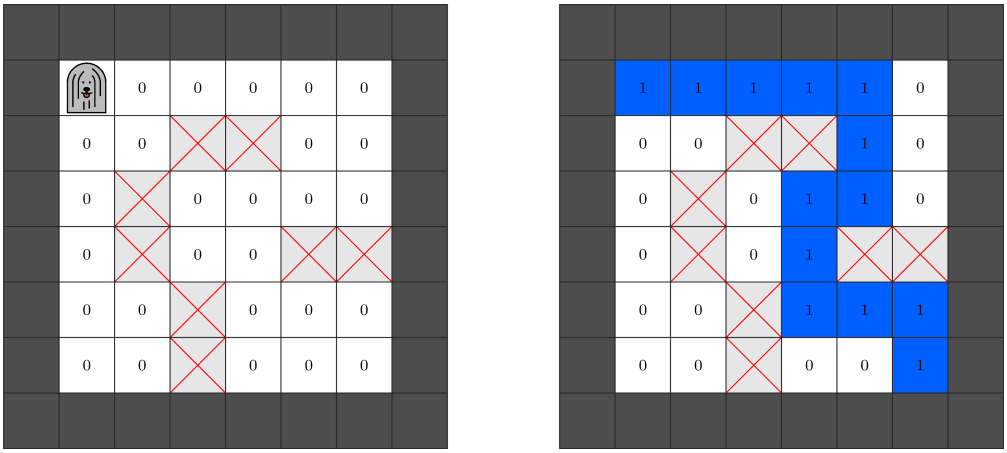
De exemplu, considerați scenariul din partea stângă a următoarei figuri. Pulibot este acum la celula $(0, 0)$ cu culoarea 0. Pulibot recunoaște șirul de stare $S = [0, -2, 2, 2, -2]$. Pulibot poate avea un program care, după recunoașterea acestui șir, setează culoarea celulei curente la $Z = 1$, apoi trece pe celula dinspre est, cum este afișat în mijlocul și în partea dreaptă a următoarei figurii.



Regulile concursului cu roboți

- La început, Pulibot este amplasat pe celula $(0, 0)$ și începe să își execute programul.
- Pulibot nu are voie să se mute pe o celulă care nu este goală.
- Programul lui Pulibot trebuie să se termine după cel mult $500\,000$ de pași.
- După ce programul lui Pulibot se termină, celulele goale din labirint ar trebui să fie colorate astfel încât:
    - Există un drum cel mai scurt de la $(0, 0)$ la $(H - 1, W - 1)$ pentru care culoarea fiecărei celule incluse în drum este 1.
    - Toate celelalte celule au culoarea 0.
- Pulibot are voie să își termine programul pe orice celulă.

De exemplu, următoarea figură arată un labirint posibil cu $H = W = 6$. Configurația inițială este arătată pe stânga și o colorare acceptabilă după terminarea programului este arătată pe dreapta.



Detalii de implementare

Ar trebui să implementați următoarea procedură.

```
void program_pulibot()
```

- Această procedură ar trebui să producă programul lui Pulibot. Acest program ar trebui să funcționeze corect pentru toate valorile lui H și W și oricare labirint care satisface constrângerile problemei.
- Procedura aceasta este apelată exact odată pentru fiecare caz de test.

Această procedură poate face apeluri la următoarea procedură pentru a produce programul lui Pulibot.

```
void set_instruction(int[] S, int Z, char A)
```

- S: un șir de lungime 5 care descrie șirul de stare.
- Z: un întreg nenegativ care reprezintă o culoare.
- A: un singur caracter care reprezintă o acțiune a lui Pulibot în felul următor:
  - `H`: stai pe loc;
  - `W`: mergi spre vest;
  - `S`: mergi spre sud;
  - `E`: mergi spre est;
  - `N`: mergi spre nord;
  - `T`: termină programul.
- Apelarea acestei proceduri îl învață pe Pulibot ca odată ce recunoaște șirul de stare S el ar trebui să execute instrucțiunea (Z, A).

Apelarea acestei proceduri de mai multe ori cu același șir de stare S va rezulta într-un verdict `Output isn't correct`.

Nu este necesar să apelezi `set_instruction` cu fiecare șir de stări posibile S. Totuși, dacă Pulibot ulterior recunoaște un șir de stare pentru care o instrucțiune nu a fost setată, vei primi verdictul `Output isn't correct`.

Dupa ce `program_pulibot` se termină, evaluatorul invocă programul lui Pulibot pentru unul sau mai multe labirinturi. Aceste invocări *nu* contează către limita de timp pentru soluția voastră. Evaluatorul *nu* este adaptiv, adică, mulțimea de labirinturi este predefinită în fiecare caz de test.

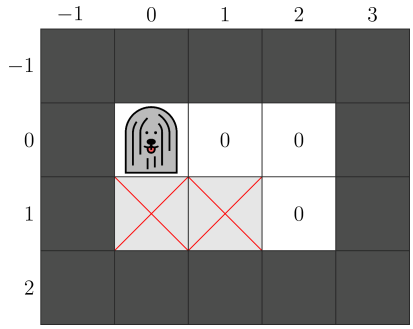
Dacă Pulibot încalcă oricare dintre regulile concursului de roboți înainte ca el să termine programul, vei primi verdictul `Output isn't correct`.

Exemplu

Procedura `program_pulibot` poate face apeluri la `set_instruction` în felul următor:

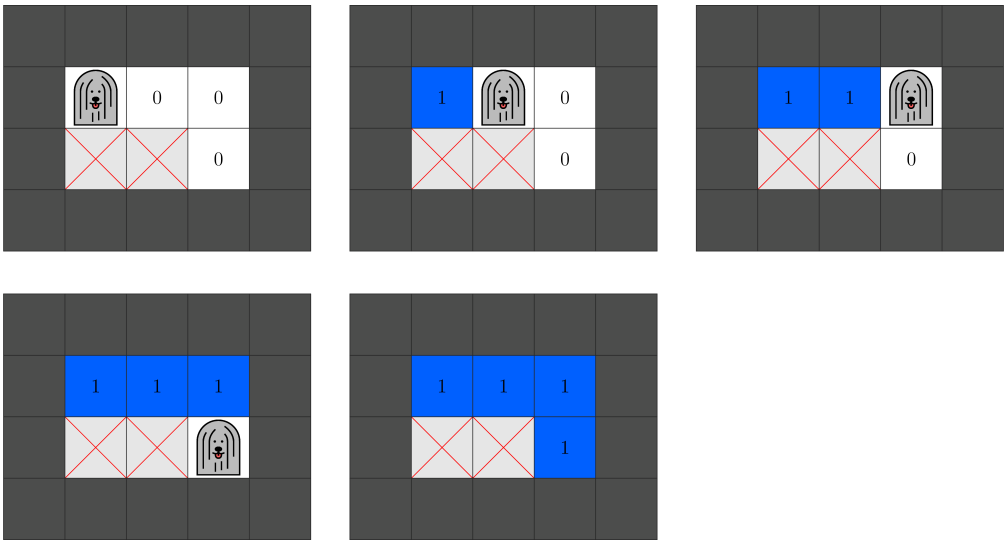
Apel	Instrucțiunea pentru șirul de stare S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Setăm culoarea la 1 și mergem spre est
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Setăm culoarea la 1 și mergem spre est
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Setăm culoarea la 1 și mergem spre sud
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Setăm culoarea la 1 și terminăm programul

Consideră scenariul unde $H = 2$ și $W = 3$, și labirintul este cel din figura următoare.



Pentru acest labirint în particular programul lui Pulibot rulează în patru pași. Șirurile de stări pe care Pulibot le recunoaște și instrucțiunile pe care le execută corespund exact cu cele patru apeluri la `set_instruction` făcute anterior, în aceeași ordine. Ultima dintre aceste instrucțiuni termină programul.

Următoarea figură arată labirintul înainte de fiecare dintre cei patru pași, și culorile finale după terminare.



Totodată, luați la cunoștință că acest program de 4 instrucțiuni ar putea să nu găsească cel mai scurt drum în alte labirinturi valide. Așadar, dacă el va fi submitat, va primi verdictul `Output isn't correct`.

Restricții

$Z_{MAX} = 19$. Așadar Pulibot poate folosi culorile de la 0 la 19 inclusiv.

Pentru fiecare labirint utilizat pentru a îl testa pe Pulibot:

- $2 \leq H, W \leq 15$
- Există măcar un drum de la celula $(0, 0)$ la celula $(H - 1, W - 1)$.

Subprobleme

1. (6 puncte) Nu există celule cu obstacole în labirint.
2. (10 puncte) $H = 2$
3. (18 puncte) Există exact un drum între oricare două celule goale.
4. (20 de puncte) Fiecare dintre drumurile cele mai scurte de la $(0, 0)$ la $(H - 1, W - 1)$ are lungimea $H + W - 2$.
5. (46 de puncte) Fără restricții suplimentare.

Dacă, în oricare caz de test, apelurile la procedura `set_instruction` sau la programul lui Pulibot, pe parcursul execuției acestuia nu sunt conforme cu restricțiile descrise în detaliile de implementare, scorul soluției voastre pentru acea subproblemă va fi 0.

În fiecare subproblemă puteți obține un scor parțial prin producerea unei colorări care este aproape corectă.

Formal:

- Soluția unui caz de test este **completă** dacă colorarea finală a celulelor goale satisface regulile concursului de roboți.
- Soluția unui caz de test este **parțiala** dacă colorarea finală este în felul următor:
  - Există un drum cel mai scurt de la $(0, 0)$ la $(H-1, W-1)$ pentru care culoarea fiecărei celule incluse în drum este 1.
  - Nu exista alte celule goale în grid cu culoarea 1.
  - Unele celule goale din grid au alte culori decât 0 sau 1.

Dacă soluția voastră la un caz de test nu este nici completă nici parțială, scorul vostru pentru cazul de test corespunzător va fi 0.

În subproblemele 1-4, scorul pentru o soluție completă este 100% și scorul pentru o soluție parțială pentru un caz de test este 50% din punctele pentru acea subproblemă.

În subproblema 5, scorul vostru depinde de numărul maxim de culori folosite în programul lui Pulibot. Mai exact, fie $Z^\star$ valoarea maxima a lui Z pentru toate apelurile făcute la `set_instruction`. Scorul unui caz de test este calculat conform următorului tabel:

Condiție	Scor (complet)	Scor (parțial)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

Scorul pentru fiecare subproblemă este minimul de puncte pentru oricare caz de test din acea subproblemă.

Exemplu de Grader

Exemplul de Grader citește inputul în următorul format:

- linia 1: H W
- linia $2 + r$ ($0 \leq r < H$): $m[r][0]$ $m[r][1]$ $\ldots$ $m[r][W-1]$

Aici m este un șir de H șiruri de W întregi ce descrie celule non-bordură ale labirintului. $m[r][c] = 0$ dacă celula (r, c) este o celulă goală, și $m[r][c] = 1$ dacă celula (r, c) este un obstacol.

Exemplul de Grader apelează mai întâi `program_pulibot()`. Dacă exemplu de Grader detectează o încălcare a protocolului, el printează `Protocol Violation: <MSG>` și termină execuția, unde `<MSG>` este unul dintre următoarele mesaje de eroare:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ nu este satisfăcut pentru o valoare a lui i, sau lungimea lui S nu este 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ nu este satisfăcut.
- `Invalid action`: caracterul A nu este unul dintre `H`, `W`, `S`, `E`, `N` sau `T`.
- `Same state array`: `set_instruction` a fost apelat cu același șir S de măcar două dăți.

Altfel, când `program_pulibot` se termină, exemplul de Grader execută programul lui Pulibot în labirintul descris de input.

Exemplul de Grader produce două output-uri.

Mai întâi, exemplul de Grader va scrie o jurnalizare a acțiunilor lui Pulibot în fișierul `robot.bin` în directorul curent. Acest fișier va servi drept date de intrare la unealta de vizualizare descrisă în secțiunea următoare.

Apoi, dacă programul lui Pulibot nu se termină cu succes, exemplul de Grader va afișa unul dintre următoarele mesaje de eroare:

- `Unexpected state`: Pulibot a recunoscut un șir de stare pentru care `set_instruction` nu a fost apelat.
- `Invalid move`: executarea unei acțiuni a rezultat în intrarea lui Pulibot într-o celulă negoală.
- `Too many steps`: Pulibot a executat $500\,000$ de pași fără să-și termine programul.

Altfel, fie $e[r][c]$ starea celulei (r, c) după ce programul lui Pulibot se termină. Exemplul de Grader printează H linii în următorul format:

- Linia $1 + r$ ($0 \leq r < H$): $e[r][0]$ $e[r][1]$ $\ldots$ $e[r][W-1]$

Unealtă de vizualizare

Pachetul de atașamente pentru această problemă conține un fișier numit `display.py`. Când este invocat, acest script Python vizualizează acțiunile lui Pulibot în labirintul descris de datele de intrare ale exemplului de Grader. Pentru acest lucru, fișierul binar `robot.bin` trebuie să fie prezent în directorul curent.

Pentru a invoca scriptul, executați următoarea comandă:

```
python3 display.py
```

O interfață grafică simplă va apărea. Opțiunile principale sunt următoarele:

- Puteți observa statutul întregului labirint. Locația curentă a lui Pulibot este indicată de un dreptunghi.
- Puteți vedea pașii lui Pulibot prin apăsarea butoanelor săgeată sau apelând hotkey-urile lor. Puteți totodată să săriți la un pas specific.

- Următorul pas în programul lui Pulibot este afișat în partea de jos. Arată șirul de stare curent și instrucțiunea pe care o va executa. După ultimul pas, va arăta fie un mesaj de eroare al Grader-ului, sau `Terminated` dacă programul se termina cu succes.
- Pentru fiecare număr ce reprezintă o culoare, puteți desemna o culoare vizuală de fundal, cât și un text de afișat. Textul de afișat este un șir de caractere scurt care va apărea în fiecare celulă cu acea culoare. Puteți desemna culori de fundal și texte de afișat în oricare dintre modurile ce urmează:
    - Setându-le într-o fereastră-dialog după apăsarea butonului `Colors`.
    - Editând conținutul fișierului `colors.txt`.
- Pentru a reîncărca `robot.bin`, folosiți butonul `Reload`. Este util dacă conținutul lui `robot.bin` a fost schimbat.