Robot Contest

AI onderzoekers aan de University of Szeged organiseren een wedstrijd in robot programmeren. Je vriend Hanga heeft besloten mee te doen aan deze wedstrijd. Het doel van de wedstrijd is om de ultieme *Pulibot* te programmeren, waarmee ze de grote intelligentie kunnen bewonderen van de beroemde Hongaarse herdershond, de Puli.

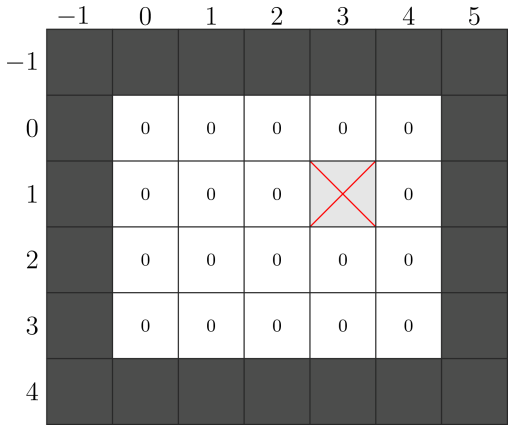
Pulibot zal worden getest in een doolhof van een grid met $(H+2) \times (W+2)$ cellen. De rijen van de grid zijn genummerd van -1 tot en met H van noord naar zuid en de kolommen zijn genummerd van -1 tot en met W van west naar oost. We verwijzen naar de cel in rij r en kolom c van de grid ($-1 \leq r \leq H$, $-1 \leq c \leq W$) als cel (r, c).

Beschouw een cel (r, c) waarbij $0 \leq r < H$ en $0 \leq c < W$. Er zijn dan 4 **buurcellen** van cel (r, c):

- cel $(r, c-1)$ wordt aangeduid als de cel **west** van cel (r, c);
- cel $(r+1, c)$ wordt aangeduid als de cel **zuid** van cel (r, c);
- cel $(r, c+1)$ wordt aangeduid als de cel **oost** van cel (r, c);
- cel $(r-1, c)$ wordt aangeduid als de cel **noord** van cel (r, c).

Cel (r, c) heet een **grenscel** van het doolhof als $r = -1$ of $r = H$ of $c = -1$ of $c = W$. Elke cel die geen grenscel is van het doolhof is of een **obstakelcel** of een **lege** cel. Bovendien heeft iedere lege cel een **kleur**, weergegeven door een niet-negatieve integer van 0 tot en met Z_{MAX}. In de beginsituatie is de kleur van iedere lege cel 0.

Bekijk bijvoorbeeld een doolhof met $H = 4$ en $W = 5$, met daarin één obstakelcel, $(1, 3)$:



De enige obstakelcel wordt aangeduid met een kruis. De grenscellen van het doolhof zijn donker aangegeven. De getallen in de lege cellen geven hun respectievelijke kleuren aan.

Een **pad** van lengte ℓ ($\ell > 0$) van cel (r_0, c_0) tot en met cel (r_ℓ, c_ℓ) is een rij van paarsgewijs verschillende *lege* cellen $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ waarbij voor elke i ($0 \leq i < \ell$) de cellen (r_i, c_i) en (r_{i+1}, c_{i+1}) buren zijn.

Merk op dat een pad van lengte ℓ uit precies $\ell + 1$ cellen bestaat.

Bij de wedstrijd zetten de onderzoekers een doolhof klaar waar er tenminste één pad bestaat vanaf cel $(0, 0)$ naar cel $(H - 1, W - 1)$. Uiteraard betekent dit dat de cellen $(0, 0)$ en $(H - 1, W - 1)$ in elk geval leeg moeten zijn.

Hanga weet niet welke cellen in het doolhof leeg zijn en welke obstakels.

Jouw taak is het om Hanga te helpen Pulibot zo te programmeren dat hij in staat is een *kortste pad* (dat wil zeggen een pad van minimale lengte) te vinden van cel $(0, 0)$ naar cel $(H - 1, W - 1)$ in het onbekende doolhof dat door de onderzoekers is klaargezet. De specificaties van Pulibot en de wedstrijdregels staan hieronder uitgelegd.

Let op dat er in het laatste deel van de opdrachtbeschrijving een visualisatietool wordt beschreven waarmee je Pulibot en zijn programma in actie kunt zien.

Pulibot's Specificaties

De **toestand** van een cel (r, c) voor iedere $-1 \leq r \leq H$ en $-1 \leq c \leq W$ geven we weer als een integer zodat:

- voor een grenscel (r, c) is de toestand -2;
- voor een obstakelcel (r, c) is de toestand -1;
- voor een lege cel (r, c) is de toestand de kleur van de cel.

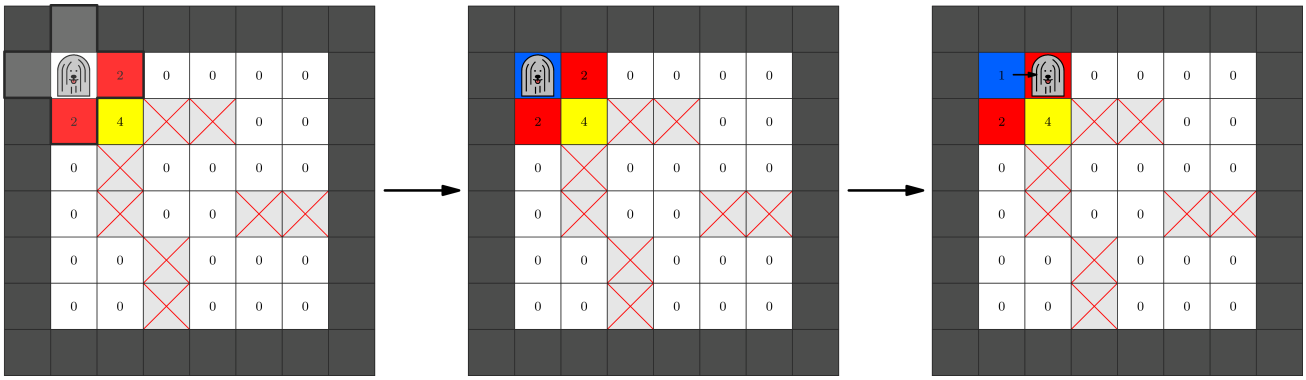
Pulibot's programma wordt uitgevoerd als een rij stappen. Bij elke stap herkent Pulibot de toestand van de buurcellen en daarna voert hij een instructie uit. Deze instructie wordt bepaald door de herkende toestanden. Hieronder een preciezere beschrijving.

Als Pulibot aan het begin van een stap in lege cel (r, c) is, gaat het als volgt.

1. Om te beginnen herkent Pulibot de huidige **toestand array**, dat is de array $S = [S[0], S[1], S[2], S[3], S[4]]$, die bestaat uit de toestand van cel (r, c) en al zijn buurcellen:
   - $S[0]$ is de toestand van cel (r, c).
   - $S[1]$ is de toestand van de cel naar het westen.
   - $S[2]$ is de toestand van de cel naar het zuiden.
   - $S[3]$ is de toestand van de cel naar het oosten.
   - $S[4]$ is de toestand van de cel naar het noorden.
2. Vervolgens bepaalt Pulibot de **instructie** (Z, A) die hoort bij de toestandsarray.
3. Tenslotte voert Pulibot de instructie uit: hij stelt de kleur van cel (r, c) in op kleur Z en voert dan actie A uit; en dat is één van de volgende acties:

- *blijf* in cel (r, c);
- *stap* naar één van de 4 buurcellen;
- *beëindig het programma*.

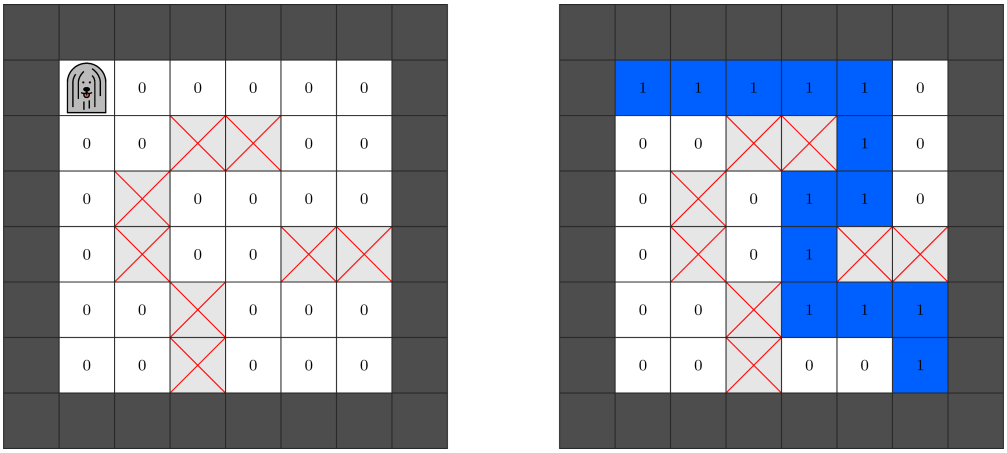
Bekijk bijvoorbeeld het scenario dat aan de linkerkant van de volgende figuur is te zien. Pulibot is nu in cel $(0, 0)$ met de kleur 0. Pulibot herkent de toestandsarray $S = [0, -2, 2, 2, -2]$. Pulibot heeft wellicht een programma waarbij hij als deze array wordt herkend hij de kleur van de huidige cel verandert in $Z = 1$ en dan naar het oosten beweegt zoals aangegeven in het midden en aan de rechterkant van de figuur.



Robot Wedstrijd Regels

- Aan het begin wordt Pulibot op cel $(0, 0)$ gezet en daar begint hij met zijn programma..
- Pulibot mag niet bewegen naar een niet-lege cel.
- Pulibot's programma moet eindigen na maximaal $500\,000$ stappen.
- Als Pulibot's programma wordt afgesloten, moeten de lege cellen in het doolhof zo zijn gekleurd dat geldt:
  - Er bestaat een kortste pad van $(0, 0)$ naar $(H - 1, W - 1)$ waarvoor de kleur van iedere cel in het pad 1 is.
  - De kleur van elke andere lege cel is 0.
- Pulibot mag op elke lege cel zijn programma beëindigen.

Het volgende figuur geeft een voorbeelddoolhof met $H = W = 6$. De configuratie aan het begin staat links aangegeven en de verwachte kleuring van de lege cellen nadat het programma is beëindigd staat rechts.



Implementatiedetails

Jij moet de volgende functie implementeren.

```
void program_pulibot()
```

- Deze functie moet het programma voor Pulibot schrijven. Dit programma moet correct werken voor alle waarden van H en W en voor ieder geldig doolhof.
- Deze functie wordt voor iedere testcase eenmaal aangeroepen.

Deze functie kan onderstaande functie meermaals aanroepen om Pulibot's programma te maken:

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

- S: array van lengte 5 dat een toestandsarray weergeeft.
- Z: een niet-negatieve integer die een kleur weergeeft.
- A: een enkel karakter dat een actie van Pulibot weergeeft:
  - `H`: blijf;
  - `W`: beweeg naar west;
  - `S`: beweeg naar zuid (south);
  - `E`: beweeg naar oost (east);
  - `N`: beweeg naar noord;
  - `T`: beëindig het programma.
- Door deze functie aan te roepen laat je Pulibot weten: als de toestandsarray S is bereikt, voer dan de instructie (Z, A) uit.

Als je deze functie vaker dan een keer aanroept met dezelfde toestandsarray S krijg je de beoordeling `Output isn't correct`.

Je hoeft `set_instruction` niet voor elk mogelijk toestandsarray te definiëren. Maar als de functie wordt aangeroepen met een toestandsarray waarvoor de functie niet was gedefiniëerd, krijg je de beoordeling `Output isn't correct`.

Als `program_pulibot` klaar is, zal de grader het programma voor Pulibot op één of meer doolhoven toepassen. Deze aanroepen tellen *niet* mee voor de tijdlimiet voor jouw oplossing. De grader is *niet* adaptief, dat wil zeggen dat de verzameling doolhoven al vastligt voor iedere testcase.

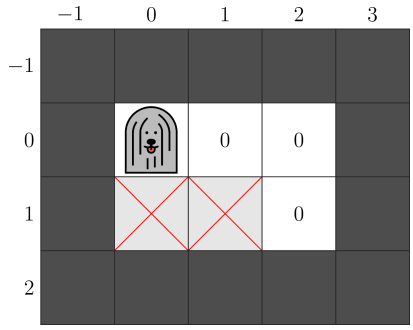
Als Pulibot één (of meer) Robot Wedstrijd Regels overtreedt voordat het programma wordt beëindigd, krijg je een beoordeling `Output isn't correct`.

Voorbeeld

De functie `program_pulibot` kan als volgt `set_instruction` aanroepen:

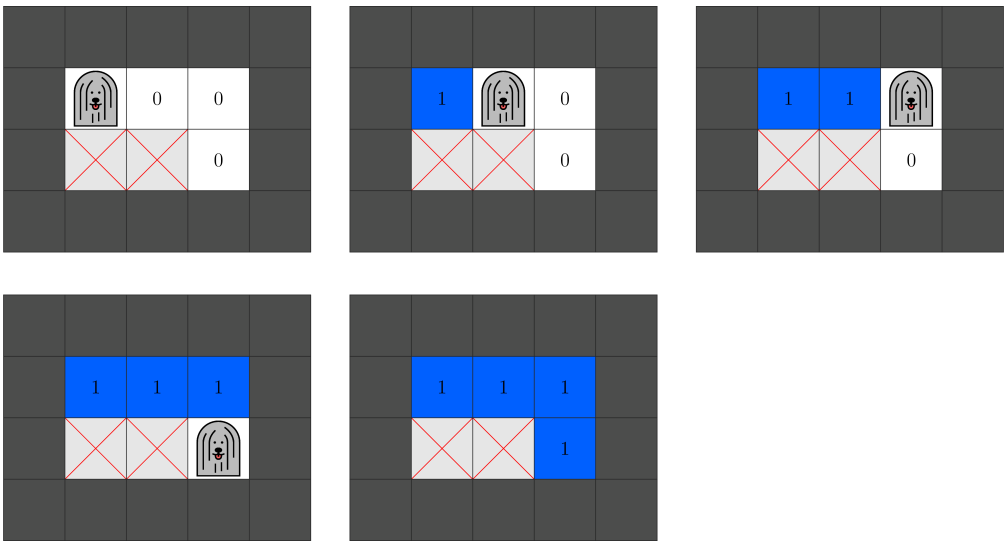
Aanroep	Instructie voor toestandsarray S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Stel de kleur in op 1 en beweeg naar het oosten
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Stel de kleur in op 1 en beweeg naar het oosten
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Stel de kleur in op 1 en beweeg naar het zuiden
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Stel de kleur in op 1 en beëindig het programma

Denk aan een scenario met $H = 2$ en $W = 3$, en het doolhof weergegeven in onderstaande figuur.



Met dit doolhof doet Pulibot's programma vier stappen. De toestandsarrays die Pulibot herkent en de instructies die hij uitvoert komen precies overeen met de vier aanroepen van `set_instruction` in de volgorde hierboven. De laatste opdracht beëindigt het programma.

In de volgende figuur zie je de toestand van het doolhof voor elk van de vier stappen en de eindtoestand nadat het programma is beëindigd.



Maar merk op dat dit programma van 4 instructies misschien niet het kortste pad zal vinden in andere geldige doolhoven. Daarom krijgt het programma als het wordt gesubmit een beoordeling `Output isn't correct`.

Randvoorwaarden

$Z_{MAX} = 19$. Pulibot kan de kleuren 0 tot en met 19 gebruiken.

Voor ieder doolhof waarmee Pulibot wordt getest geldt:

- $2 \leq H, W \leq 15$
- Er is minstens één pad van cel $(0, 0)$ naar cel $(H - 1, W - 1)$.

Subtasks

1. (6 punten) Er is geen obstakelcel.
2. (10 punten) $H = 2$
3. (18 punten) Tussen elke twee lege cellen is er precies één pad.
4. (20 punten) Elk kortste pad van cel $(0, 0)$ naar cel $(H - 1, W - 1)$ heeft lengte $H + W - 2$.
5. (46 punten) Geen aanvullende voorwaarden.

Als in één van de testcases de aanroep van de functie `set_instruction`, of Pulibot's programma tijdens de uitvoering niet overeenkomt met de beperkingen onder de implementatiedetails, wordt de score voor die subtask 0.

In elke subtask kun je een gedeeltelijke score krijgen voor een kleuring die bijna correct is.

Formeel:

- De oplossing van een testcase is **compleet** als de kleuring van de lege cellen voldoet aan de Robot Wedstrijd Regels.
- De oplossing van een testcase is **gedeeltelijk** als de kleuring er als volgt uitziet:
  - Er bestaat een kortste pad van $(0, 0)$ naar $(H - 1, W - 1)$ waarvoor de kleur van elke cel in het pad gelijk is aan 1.
  - Geen enkele andere cel heeft als kleur 1.
  - Er zijn lege cellen met een kleur, anders dan 0 en 1.

Als je oplossing van een testcase niet compleet is en ook niet gedeeltelijk, krijg je voor de betreffende testcase 0 punten.

In de subtasks 1-4, krijg je 100% van de punten voor de subtasks voor een complete oplossing en 50% van de punten als je oplossing gedeeltelijk is.

In subtask 5, hangt je score af van het aantal kleuren dat Pulibot's program gebruikt. Preciezer gezegd, noem $Z^\star$ de maximale waarde van Z in alle aanroepen van `set_instruction`. De score van de testcase wordt bepaald volgens onderstaande tabel:

Voorwaarde	Score (compleet)	Score (gedeeltelijk)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

De score voor elke subtask is het minimum van de punten voor de testcases in de subtask.

Sample Grader

De sample grader leest de invoer in het volgende format:

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

Hier is m een array van H arrays van W integers, die de niet grenscellen van het doolhof beschrijven. $m[r][c] = 0$ als cel (r, c) leeg is en $m[r][c] = 1$ als cel (r, c) een obstakelcel is.

De sample grader roept eerst `program_pulibot()` aan. Als de sample grader een overtreding van het protocol constateert, print de sample grader `Protocol Violation: <MSG>` en dan stopt hij, waarbij `<MSG>` één van de volgende foutmeldingen is:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ klopt niet voor een zekere i of de lengte van S is ongelijk aan 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ geldt niet.
- `Invalid action`: het karakter A is niet één van de volgende: `H`, `W`, `S`, `E`, `N` of `T`.
- `Same state array`: `set_instruction` werd vaker dan eens met dezelfde array S aangeroepen.

In overige gevallen, als `program_pulibot` gewoon eindigt, wordt het programma van Pulibot door de sample grader uitgevoerd op het doolhof uit de invoer.

De sample grader produceeert twee soorten uitvoer.

Allereerst schrijft de sample grader een logbestand van de acties van Pulibot weg in een bestand `robot.bin` in de actieve directory. Dit bestand dient als invoer voor de visualisatietool die in de volgende paragraaf wordt beschreven.

Daarnaast print de sample grader als Pulibot's program niet succesvol wordt beëindigd één van de volgende foutmeldingen:

- `Unexpected state`: Pulibot gebruikt een toestandsarray waar `set_instruction` niet mee werd aangeroepen.
- `Invalid move`: een actie resulteerde erin dat Pulibot naar een niet-lege cel wilde bewegen.
- `Too many steps`: Pulibot deed $500\,000$ stappen zonder zijn programma te beëindigen.

Laat in andere gevallen $e[r][c]$ de toestand van cel (r, c) zijn als Pulibot's programma eindigt. De sample grader print H regels in het volgende format:

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

Display Tool

In het attachment package voor deze taak zit een bestand `display.py`. Als dat wordt aangeroepen laat dit Python script de acties van Pulibot zien in het doolhof van de invoer die de sample grader had ontvangen. Daarom moet het bestand `robot.bin` aanwezig zijn in de actieve directory.

Om het script aan te roepen, moet je het volgende commando ingeven.

```
python3 display.py
```

Een eenvoudige grafische interface verschijnt. Belangrijkste dingen zijn de volgende:

- Je kunt de toestand zien van het hele doolhof. De huidige locatie van Pulibit wordt aagegeven door een rechthoek.
- Je kunt door de stappen van Pulibot grasduinen door de pijltjestoetsen of de sneltoetsen te gebruiken. Je kunt ook naar een sepecifieke stap springen.
- De volgende stap in Pulibot's programma staat onder in beeld. Dat toont de huidige toestandsarray en de instructie die zal worden uitgevoerd. Na de laatste stap toont het of één van de foutmeldingen van de grader, of `Terminated` als het programma succesvol is geëindigd.
- Voor elk getal dat een kleur aanduidt kun je een achtergrondkleur instellen en/of een weer te geven tekst. Die tekst wordt dan in iedere cel met die kleur weergegeven. Kleuren en teksten aangeven kan op de volgende manieren:
  - Druk op de knop `Colors` en stel ze in in een dialoog venster.
  - Bewerk de inhoud van `colors.txt`.
- Om `robot.bin` opnieuw in te laden kun je de `Reload` button gebruiken. Dat is handig als de inhoud van het bestand `robot.bin` is aangepast.