# Robot Contest

Սեգեդի համալսարանի արհեստական բանականության հետազոտողները անց են կացնում ռոբոտների ծրագրավորման մրցույթ։ Ձեր ընկեր Հանգան որոշել է մասնակցել այդ մրցույթին։ Պահանջվում է ստեղծել *Pulibot*, ի պատիվ հունագարական խելացի Puli ցեղատեսակի շների։

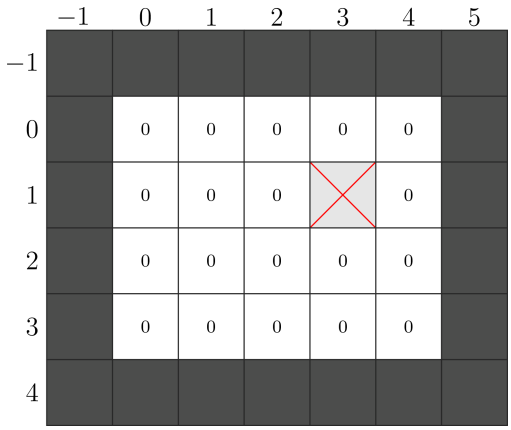
Pulibot-ը պետք է թեստավորվի լաբիրինթոսի վրա՝ բաղկացած $(H+2) \times (W+2)$ չափի վանդակավոր ցանցից։ Ցանցի տողերը համարակալված են $-1$-ից $H$ թվերով հյուսիսից հարավ, իսկ սյուները՝ $-1$-ից $W$ թվերով արևմուտքից արևելք։ Մենք ցանցի $r$ տողի $c$ սյան, ($-1 \leq r \leq H$, $-1 \leq c \leq W$), վանդակին կդիմենք այսպես՝ $(r, c)$։

Դիտարկենք $(r, c)$ վանդակը, որտեղ $0 \leq r < H$ and $0 \leq c < W$։ $(r, c)$ վանդակն ունի $4$ **կից** վանդակներ.

- $(r, c-1)$ վանդակը, որը $(r, c)$ վանդակից **արևմուտք** է,
- $(r+1, c)$ վանդակը, որը $(r, c)$ վանդակից **հարավ** է;
- $(r, c+1)$ վանդակը, որը $(r, c)$ վանդակից **արևելք** է;
- $(r-1, c)$ վանդակը, որը $(r, c)$ վանդակից **հյուսիս** է։

$(r, c)$ վանդակը կոչվում է լաբիրինթոսի **եզրային** վանդակ, եթե $r = -1$, կամ $r = H$, կամ $c = -1$, կամ $c = W$։ Ոչ եզրային վանդակները կամ **խոչընդոտ** են, կամ **դատարկ**։ Բացի դա, բոլոր դատարկ վանդակները **գույն** ունեն, որը ներկայացված է $0$-ից $Z_{MAX}$ ներառյալ տիրույթին պատկանող ամբողջ թվով։ Սկզբում բոլոր վանդակները $0$ գույնն ունեն։

Օրինակ, դիտարկենք $H = 4$ և $W = 5$ չափերով լաբիրինթոս, որի $(1, 3)$ վանդակը խոչընդոտ է։



Նկարում խոչընդոտ վանդակը նշված է խաչով: Լաբիրինթոսի եզրային վանդակները սև են: Դատարկ վանդակներում գրված թվերը ցույց են տալիս նրանց գույները:

$(r_0, c_0)$ վանդակից $(r_\ell, c_\ell)$ վանդակ տանող $\ell$ ($\ell > 0$) երկարության **ճանապարհը** զույգ առ զույգ տարբեր *դատարկ* վանդակների $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ հաջորդականություն է, որտեղ յուրաքանչյուր $i$-ի ($0 \leq i < \ell$) համար $(r_i, c_i)$ և $(r_{i+1}, c_{i+1})$ վանդակները կից են:

Նկատեք, որ $\ell$ երկարության ճանապարհը պարունակում է ճիշտ $\ell + 1$ վանդակ:

Մրցույթի ժամանակ հետազոտողները կառուցում են լաբիրինթոս, որն ունի $(0, 0)$ վանդակից $(H - 1, W - 1)$ վանդակ տանող առնվազն մեկ ճանապարհ: Սա նշանակում է, որ $(0, 0)$ և $(H - 1, W - 1)$ վանդակների դատարկ լինելը երաշխավորված է:

Հանգան չգիտի, թե լաբիրինթոսի որ վանդակներն են դատարկ, որոնք են խոչընդոտ:

Ձեր խնդիրն է օգնել Հանգային ծրագրավորելու Pulibot-ն այնպես, որ նա կարողանա հետազոտողների կողմից կառուցված անհայտ լաբիրինթոսում գտնել $(0, 0)$ վանդակից $(H - 1, W - 1)$ վանդակ տանող *կարճագույն ճանապարհը* (այսինքն, մինիմալ երկարության ճանապահը): Pulibot-ի սպեցիֆիկիացիան և մրցույթի կանոնները նկարագրված են ստորև:

Նկատեք, որ այս խնդրի շարադարնքի վերջին բաժնում նկարագրված է պատկերող գործիք, որը դուք կարող եք օգտագործել Pullbot-ը վիզուալիզացնելու համար:

## Pulibot-ի սպեցիֆիկացիան

Սահմանենք $(r, c)$ վանդակի **վիճակը**, որտեղ $-1 \leq r \leq H$ և $-1 \leq c \leq W$ որպես ամբողջ թիվ, այնպես, որ

- եթե $(r, c)$ վանդակը եզրային է, ապա նրա վիճակը $-2$ է;
- եթե $(r, c)$ վանդակը խոչընդոտ է, ապա նրա վիճակը $-1$ է;
- եթե $(r, c)$ վանդակը դատարկ է, ապա նրա վիճակը այդ վանդակի գույնն է:

Pulibot-ի ծրագիրը կատարվում է որպես քայլերի հաջորդականություն: Յուրաքանչյուր քայլին Pulibot-ը ճանաչում է հարևան վանդակների վիճակները, ապա կատարում է հրահանգը: Իսկ հրահանգը որոշվում է ճանաչած վիճակների հիման վրա: Ավելի մանրամսն նկարագրված է ստորև:

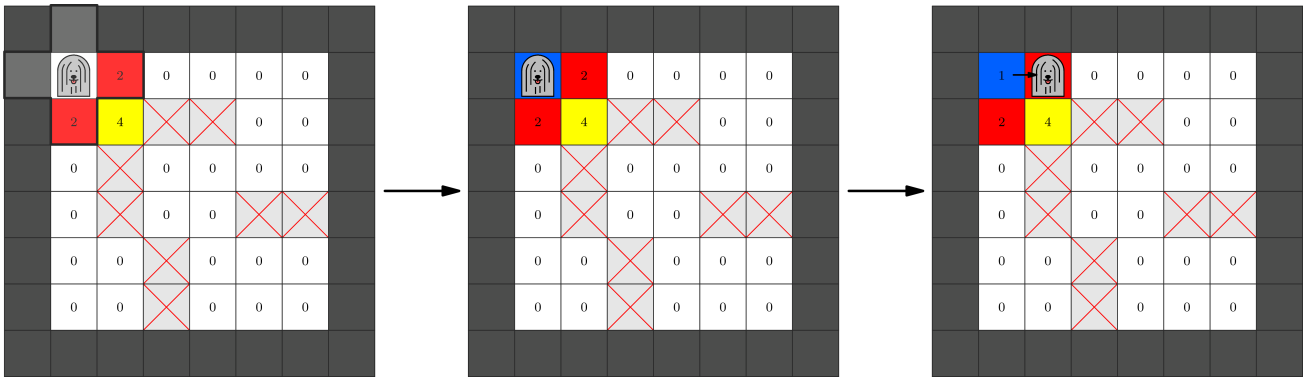
Ենթադրենք, ընթացիկ քայլի սկզբում Pulibot-ը գտնվում է $(r, c)$ վանդակում, որը դատարկ վանդակ է: Քայլը կատարվում է հետևյալ կերպ.

1. Նախ, Pulibot-ը ճանաչում է ընթացիկ **վիճակների զանգվածը**, այսինքն $S = [S[0], S[1], S[2], S[3], S[4]]$ զանգվածը, որը բաղկացած է $(r, c)$ վանդակի և նրան կից վանդակների վիճակներից.
    - $S[0]$-ն $(r, c)$ վանդակի վիճակն է:

- $S[1]$-ը նրանից դեպի արևմուտք կից վանդակի վիճակն է։
- $S[2]$-ը նրանից դեպի հարավ կից վանդակի վիճակն է։
- $S[3]$-ը նրանից դեպի արևելք կից վանդակի վիճակն է։
- $S[4]$-ը նրանից դեպի հյուսիս կից վանդակի վիճակն է։

2. Ապա Pulibot-ը պարզում է $(Z, A)$ **հրահանգը**, որը համապատասխանում է ճանաչած վիճակների զանգվածին։
3. Վերջում Pulibot-ը կատարում է հարահանգը՝ նա $(r, c)$ վանդակի գույնը դարձնում է $Z$, իսկ հետո կատարում է $A$ գործողությունը, որը հետևյալ գործողություններից մեկն է.
   - *մնալ* $(r, c)$ վանդակում;
   - *տեղափոխվել* $4$ կից վանդակներից մեկը;
   - *ավարտել ծրագիրը*։

Օրինակ, դիտարկենք հետևյալ նկարի ձախ կողմում պատկերված դեպքը։ Pulibot-ն այժմ գտնվում է $(0, 0)$ վանդակում, և նրա գույնը $0$ է։ Pulibot-ը ճանաչում է վիճակների $S = [0, -2, 2, 2, -2]$ զանվածը։ Pulibot-ը կարող է ունենալ ծրագիր, որը ճանաչած զանգվածի հիման վրա ընթացիկ վանդակի գույնը դարձնում է $Z = 1$, և շարժվում է դեպի արևելք, ինչպես պատկերված է նկարի մեջտեղի մասում.

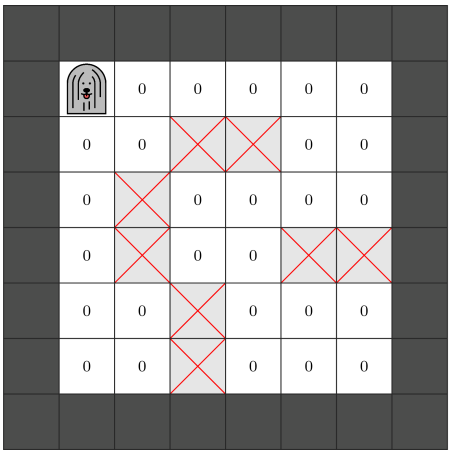


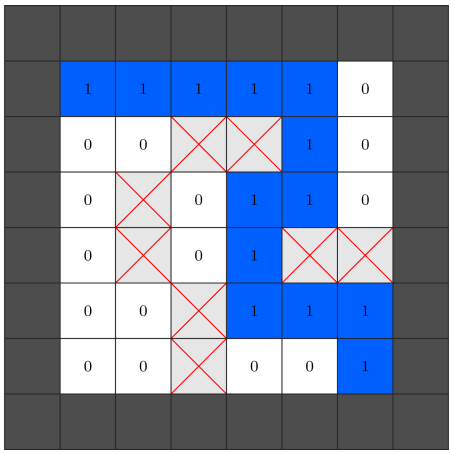
## Ռոբոտների մրցույթի կանոնները

- Սկզբում Pulibot-ը տեղադրվում է $(0, 0)$ վանդակում, և նա սկսում է կատարել իր ծրագիրը։
- Pulibot-ին չի թույլատրվում տեղափոխվել կից վանդակ, եթե այն դատարկ վանդակ չէ։
- Pulibot-ի ծրագիրը պետք է ավարտվի առավելագույնը $500\,000$ քայից հետո։
- Pulibot-ի ծրագրի ավարտից հետո լաբիրինթոսի դատարկ վանդակները պետք է ներկված լինեն այնպես, որ
  - Գոյություն ունի $(0, 0)$-ից $(H - 1, W - 1)$ տանող կարճագույն ճանապարհ, որի մեջ մտնող բոլոր վանդակները ներկված են $1$ գույնով։
  - Մնացած բոլոր դատարկ վանդակները ներկված լինեն $0$ գույնով։
- Pulibot-ը կարող է ավարտել ծրագիրը ցանկացած դատարկ վանդակում։

Օրինակ, հետևյալ նկարում պատկերված է $H = W = 6$ չափի լաբիրինթոս։ Սկզբնական կոնֆիգուրացիան պատկերված է ձախ մասում, իսկ ծրագրի ավարտից հետո դատարկ

վանդակների սպասվող գունավորումը պատկերված է աջ մասում։





## Իրականացման մանրամասներ

Դուք պետք է ծրագրավորեք հետևյալ ֆունկցիան.

```
void program_pulibot()
```

- Այս ֆունկցիան պետք է ստեղծի Pulibo-ի ծրագիրը։ Այդ ծրագիրը պետք է ճիշտ աշխատի $H$-ի և $W$-ի բոլոր արժեքների և խնդրի պայմաններին բավարարող բոլոր լաբիրինթոսների համար։
- Ֆունկցիան յուրաքանչյուր թեստի համար կանչվում է ճիշտ մեկ անգամ։

Այս ֆունկցիան Pulibot-ի ծրագիրը ստեղծելու համար կարող է կատարել հետևյալ ֆունկցիայի կանչեր.

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

- $S$: 5 երկարության վիճակների զանգված։
- $Z$: գույնը ներկայացնող ոչ բացասական ամբողջ թիվ։
- $A$: սիմվոլ, որը նեկայացնում է Pulibot-ի գործողությունը հետևյալ կերպ.
  - `H`: մնալ տեղում;
  - `W`: գնալ դեպի արևմուտք;
  - `S`: գնալ դեպի հարավ;
  - `E`: գնալ դեպի արևելք;
  - `N`: գնալ դեպի հյուսիս;
  - `T`: ավարտել ծրագիրը։
- Այս ֆունկցիայի կանչը հրահանգ է տալիս Pulibot-ին որ նա $S$ վիճակների զանգվածը ճանաչելուց հետո, պետք է կատարի $(Z, A)$ հրահանգը։

Այս ֆունկցիան նույն $S$ վիճակների զանգվածի համար մի քանի անգամ կանչելու դեպքում կտրվի `Output isn't correct` վճիռը։

Յուրաքանչյուր հնարավոր $S$-ի համար չի պահանջվում կանչել `set_instruction` ֆունկցիան։ Սակայն, եթե Pulibot-ը հետագայում հանդիպի այնպիսի վիճակների զանգվածի, որի համար հրահանգ չկա, Դուք կստանաք `Output isn't correct` վճիռ։

`program_pulibot`ֆունկցիայի ավարտից հետո գրեյդերը կաշխատեցնի Pulibot-ի ծրագիրը մեկ կամ մի քանի լաբիրինթոսների վրա։ Դա ձեր լուծման time limit-ի վրա *չի* ազդի։ Գրեյդերը հարմարվողական *չէ*, այսինքն յուրաքանչյուր թեստի մեջ լաբիրինթոսները նախապես ֆիքսված են։

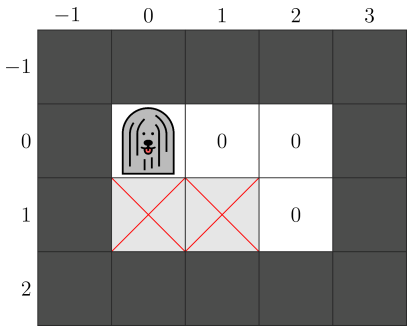
Եթե Pulibot-ը նախքան իր ծրագրի ավարտը խախտի ռոբոտների մրցույթի կանոններից որևէ մեկը, Դուք կստանաք `Output isn't correct` վճիռը։

## Օրինակ

`program_pulibot` ֆունկցիան կարող է `set_instruction`-ի կանչեր անել հետևյալ կերպ.

| Կանչ | Հրահանգ $S$ վիճակների զանգվածի համար |
|---|---|
| `set_instruction([0, -2, -1, 0, -2], 1, E)` | Գույնը դարձնել 1 և շարժվել դեպի արևելք |
| `set_instruction([0, 1, -1, 0, -2], 1, E)` | Գույնը դարձնել 1 և շարժվել դեպի արևելք |
| `set_instruction([0, 1, 0, -2, -2], 1, S)` | Գույնը դարձնել 1 և շարժվել դեպի հարավ |
| `set_instruction([0, -1, -2, -2, 1], 1, T)` | Գույնը դարձնել 1 և ավարտել ծրագիրը։ |

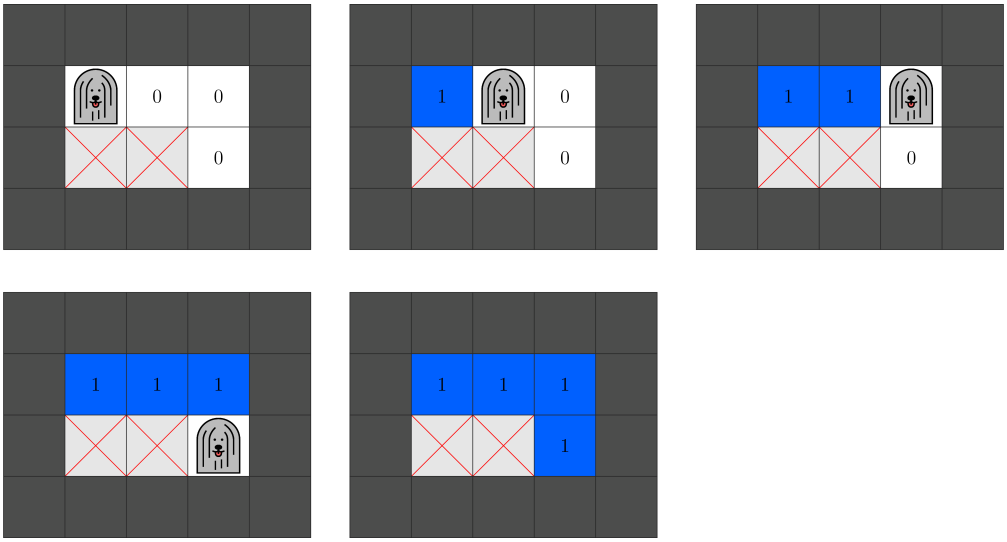
Դիտարկենք ստորև նկարկում պատկերված լաբիրինթոսը, որտեղ $H = 2$ և $W = 3$.




Այս կոնկրետ լաբիրինթոսի համար Pulibot-ի ծրագիրն ընթանում է չորս քայլով։ Pulibot-ը ճանաչում է վիճակների զանգվածները և կատարում հրահանգները, որոնք ճշտորեն համապատասխանում են `set_instruction` ֆունկցիայի վերը նկարագրված չորս

կանչերին, նույն հերթականությամբ։ Այդ հրահանգներից վերջինը ավարտում է ծրագիրը։

Հետևյալ նկարը ցույց է տալիս լաբիրինթոսի վիճակը չորս քայլերից յուրաքանչյուրից առաջ և նրա վերջնական վիճակը ավարտից հետո։




Նկատենք, սակայն, որ կարող են լինել ուրիշ այնպիսի լաբիրինթոսները, որ 4 հրահանգներից բաղկացած այս ծրագիրը չգտնի կարճագույն ճանապարհը։ Հետևաբար, այն submit անելու դեպքում, կստացվի `Output isn't correct` վճիռը։

## Սահմանափակումներ

$Z_{MAX} = 19$. Այսինքն, Pulibot-ը կարող է օգտագործել 0-ից 19, ներառյալ, գույները։

Pulibot-ի թեստավորման համար տրվող բոլոր լաբիրինթոսներում

- $2 \leq H, W \leq 15$
- Գոյություն ունի $(0, 0)$ վանդակից $(H - 1, W - 1)$ վանդակ տանող առնվազն մեկ ճանապարհ։

## Ենթախնդիրներ

1. (6 միավոր) Լաբիրինթոսում խոչընդոտ չկա։
2. (10 միավոր) $H = 2$
3. (18 միավոր) Դատարկ վանդակներից յուրաքանչյուր զույգի միջև գոյություն ունի ճիշտ մեկ ճանապարհ։
4. (20 միավոր) $(0, 0)$ վանդակից $(H - 1, W - 1)$ վանդակ տանող կարճագույն ճանապարհի երկարությունը $H + W - 2$ է։
5. (46 միավոր) Լրացուցիչ սահմանափակումներ չկան։

Եթե ենթախնդրի որևէ թեստում `set_instruction` ֆունկցիայի կանչերը կամ Pulibot-ի ծրագրի կատարումը չեն համապատասխանում սահմանափակումներին, ձեր

ծրագրին այդ ենթախնդրի համար կտրվի $0$ միավոր։

Յուրաքանչյուր ենթախնդրում դուք կարող եք ունենալ մասնակի միավոր համարյա ճիշտ գունավորում տալու դեպքում։

Ֆորմալ.

- Թեստի լուծումը **լրիվ** է, եթե դատարկ վանդակների վերջնական գունավորումը բավարարում է ռոբոտների մրցույթի պայմաններին։
- Թեստի լուծումը **մասնակի** է, եթե վերջնական գունավորումն այսպիսին է.
  - Գոյություն ունի $(0, 0)$-ից $(H - 1, W - 1)$ տանող կարճագույն ճանապարհ, որի բոլոր վանդակների գույնը $1$ է։
  - Ցանցում ուրիշ դատարկ վանդակ չկա, որի գույնը $1$ է։
  - Ցանցում կան դատարկ վանդակներ, որոնց գույնը ոչ $0$ է, ոչ էլ $1$։

Եթե Ձեր լուծումը տվյալ թեստի համար ոչ լրիվ է, ոչ էլ մասնակի, ապա այդ թեստ համար կտրվի $0$ միավոր։

1-4, ենթախնդիրներում մասնակի լուծման համար կտրվի ենթախնդրի համար նախատեսված միավորի 50%-ը։

5-րդ ենթախնդրում, Ձեր միավորը կախված է Pulibot-ի ծրագրում օգտագործված գույների քանակից։ Ավելի ճշգրիտ, $Z^\star$-ով նշանակենք `set_instruction`-ի բոլոր կանչերում $Z$-ի մեծագույն արժեքը։ Թեստի համար միավորը հաշվվում է հետևյալ աղյուսակին համապատասխան.

| Պայման | Միավոր (լրիվ) | Միավոր (մասնակի) |
|---|---|---|
| $11 \leq Z^\star \leq 19$ | $20 + (19 - Z^\star)$ | $12 + (19 - Z^\star)$ |
| $Z^\star = 10$ | $31$ | $23$ |
| $Z^\star = 9$ | $34$ | $26$ |
| $Z^\star = 8$ | $38$ | $29$ |
| $Z^\star = 7$ | $42$ | $32$ |
| $Z^\star \leq 6$ | $46$ | $36$ |

Յուրաքանչյուր ենթախնդրի համար տրվում է այդ ենթախնդրի թեստերից ստացված միավորներից մինիմումը։

## Գրեյդերի նմուշ

Գրեյդերի նմուշը մուտքային տվյալները կարդում է հետևյալ ձևաչափով.

- Տող $1$. $H$ $W$

- Տող $2 + r$ ($0 \le r < H$). $m[r][0]\ m[r][1]\ \ldots\ m[r][W-1]$

Այստեղ, $m$-ը $H$ երկարության զանգված է, որի տարրերը $W$ երկարության ամբողջ թվերի զանգվածներ են, սրանով նկարագրվում են լաբիրինթոսի ոչ եզրային վանդակները:. $m[r][c] = 0$, եթե $(r, c)$ վանդակը դատարկ է, և $m[r][c] = 1$, եթե $(r, c)$ վանդակը խոչընդոտ է:

Գրեյդերի նմուշը սկզբում անում է `program_pulibot()`-ի կանչ: Եթե գրեյդերի նմուշը հայտնաբերում է պրոտոկոլի խախտում, տպում է `Protocol Violation: <MSG>` հաղորդագրությունը և ավարտում է աշխատանքը, որտեղ `<MSG>`-ը սխալի վերաբերյալ հետևյալ հաղորդագրություններից մեկն է.

- `Invalid array`: ինչ-որ $i$ կա, որի համար $-2 \le S[i] \le Z_{MAX}$ պայմանը չի կատարվել, կամ $S$-ի երկարությունը $5$ չէ:
- `Invalid color`: $0 \le Z \le Z_{MAX}$ պայմանը խախտվել է:
- `Invalid action`: $A$ սիմվոլը տարբերվում է `H`, `W`, `S`, `E`, `N` և `T` սիմվոլներից, ոչ մեկին հավասար չէ:
- `Same state array`: `set_instruction` ֆունկցիան միևնույն $S$ զանգվածի համար կանչվել է առնվազն երկու անգամ:

Հակառակ դեպքում, երբ `program_pulibot`-ն ավարտվում է, գրեյդերը կատարում է Pulibot-ի ծրագիրը մուտքային տվյալներում նկարագրված լաբիրինթոսում:

Գրեյդերի նմուշը երկու բան է արատածում:

Առաջին, գրեյդերի նմուշը կգրի Pulibot-ի գործողությունների լոգը `robot.bin` ֆայլում, որը կպավիի աշխատանքային ֆոլդերում: Այդ ֆայլը պարունակությունն օգտագործվում է որպես մուտքային տվյալներ վիզուալիզացիա անող գործիքի համար, որը նկարագրված է հետևյալ բաժնում:

Երկրորդ, եթե Pulibot-ի ծրագիրը հաջող չի ավարտվում, գրեյդերի նմուշը տպում է սխալների վերաբերյալ հետևյալ հաղորդագրություններից մեկը.

- `Unexpected state`: Pulibot-ը ճանաչում է այնպիսի վիճակների զանգված, որով `set_instruction`-ը չի կանչվել:
- `Invalid move`: հրահանգի կատարման արդյունքում Pulibot-ը հայտնվում է ոչ դատարկ վանդակում:
- `Too many steps`: Pulibot-ը կատարում է ավելի քան $500\,000$ քայլ առանց ծրագիրն ավարտելու:

Հակառակ դեպքում, դիցուք $e[r][c]$-ը $(r, c)$ վանդակի վիճակն է Pulibot-ի ծրագրի կատարումից հետո: Գրեյդերի նմուշը տպում է $H$ տող հետևյալ ձևաչափով.

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

# Պատկերող գործիք

Այս խնդրին կցված փաթեթը պարունակում է `display.py` անունով ֆայլ։ Աշխատեցնելու դեպքում Python-ով գրված script-ը պատկերում է գրեյդերի նմուշի մուտքային տվյալներում նկարագրված լաբիրինթոսում Pulibot-ի գործողությունները։ Դրա համար աշխատանքային ֆոլդերում պետք է առկա լինի `robot.bin` անունով երկուական ֆայլը։

Python-ով գրված script-ը աշխատեցնելու համար հավաքեք տպեք հրամանը.

```
python3 display.py
```

Պարզ գրաֆիկական ինտերֆեյս է հայտնվում։ Հիմնական հատկանիշները հետեւյալն են.

- Դուք կարող եք դիտարկել ամբողջական լաբիրինթոսի կարգավիճակը։ Pulibot-ի ներկայիս գտնվելու վայրը ընդգծված է ուղղանկյունով։
- Դուք կարող եք թերթել Pulibot-ի քայլերը՝ կտտացնելով սլաքների կոճակները կամ սեղմելով դրանց համապատասխան ստեղները։ Դուք կարող եք նաև ցատկել մինչև որոշակի քայլի։
- Pulibot-ի ծրագրում առաջիկա քայլը ցույց է տրված ներքևում։ Այն ցույց է տալիս ընթացիկ վիճակների զանգվածը և հրահանգները, որ պետք է կատարվեն։ Վերջնական քայլից հետո այն ցույց է տալիս գրեյդերի սխալների հաղորդագրություններից մեկը, կամ `Terminated` բառը, եթե ծրագիրը հաջողությամբ ավարտվել է։
- Գույն ներկայացնող յուրաքանչյուր թվի դուք կարող եք վերագրել ինչպես տեսանելի ֆոնի գույն, այնպես էլ պատկերվող տեքստ։ Պատկերվող տեքստը կարճ տող է, որը պետք է տպվի միևնույն գույն ունեցող բոլոր վանդակներում։ Ֆոնի գույնը կամ պատկերվող տեքստը դուք կարող եք վերագրել հետևյալ երկու եղանակով։
  - Դրեք դրանք երկխոսության պատուհանի միջոցով, որը բացվում է `Colors` կոճակը կտտեցնելու դեպքում,
  - Գրեք դրանք `colors.txt` ֆայլում։
- `robot.bin`-ը վերաբեռնելում համար օգտագործեք `Reload` կոճակը։ Դա օգտակար է, եթե `robot.bin`-ի պարունակությունը փոխվել է։