რობოტების შეჯიბრება

სეგედის უნივერსტეტში ხელოვნური ინტელექტის მკვლევარები ატარებენ რობოტების დაპროგრამების შეჯიბრებას. თქვენმა მეგობარმა, ჰანგამ, მონაწილეობის მიღება გადაწყვიტა. მიზანი არის შეიქმნას სრულყოფილი *პულიბოტი*, ძაღლის ცნობილი უნგრული ჯიშის, პულის გამორჩეული ინტელექტის აღსანიშნავად.

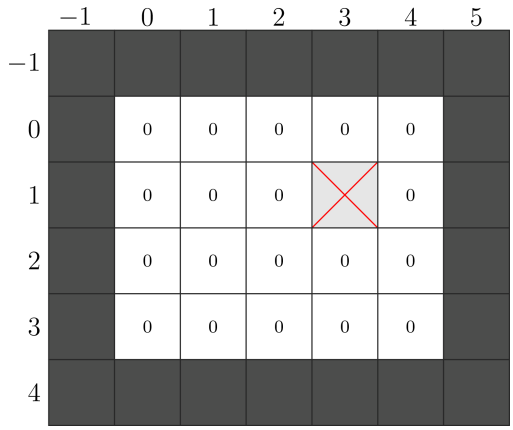
პულიბოტის გამოცდა მოხდება $(H+2) \times (W+2)$ ფორმის ლაბირინთში. ლაბირინთის სტრიქონები გადაინომრება -1-დან H-მდე ჩრდილოეთდან სამხრეთისაკენ და -1-დან W-მდე დასავლეთიდან აღმოსავლეთისაკენ. r სტრიქონსა და c სვეტში მდებარე უჯრა ($-1 \leq r \leq H$, $-1 \leq c \leq W$) აღვნიშნოთ (r, c)-თი.

განვიხილოთ უჯრა (r, c) ისეთი, რომ $0 \leq r < H$ და $0 \leq c < W$. დავარქვათ მისი **მეზობლები** შემდეგ 4 უჯრას:

- $(r, c-1)$ არის უჯრა (r, c)-ს **დასავლეთით**;
- $(r+1, c)$ არის უჯრა (r, c)-ს **სამხრეთით**;
- $(r, c+1)$ არის უჯრა (r, c)-ს **აღმოსავლეთით**;
- $(r-1, c)$ არის უჯრა (r, c)-ს **ჩრდილოეთით**.

უჯრა (r, c)-ს ეწოდება **საზღვრის** უჯრა, თუ $r = -1$, ან $r = H$, ან $c = -1$, ან $c = W$. თითოეული უჯრა, რომელიც არ არის საზღვრის უჯრა, არის **დაბლოკილი** ან **ცარიელი**. დამატებით, თითოეულ ცარიელ უჯრას აქვს **ფერი**, რომელიც წარმოდგენილი იქნება არაუარყოფითი მთელი რიცხვით 0-დან Z_{MAX}-ის ჩათვლით. თავიდან ყველა უჯრის ფერი არის 0.

მაგალითისთვის განვიხილოთ ლაბირინთი ზომებით $H = 4$ და $W = 5$, რომელშიც ერთადერთი დაბლოკილი უჯრა არის $(1, 3)$:



დაბლოკილი უჯრა გამოსახულია ჯვრით, ხოლოს საზღვრის უჯრები გამუქებულია. დანარჩენ უჯრებში ჩაწერილი რიცხვები წარმოადგენენ მათ ფერებს.

ℓ ($\ell > 0$) სიგრძის **გზა** (r_0, c_0)-დან (r_ℓ, c_ℓ)-მდე ეწოდება წყვილ-წყვილად განსხვავებული *ცარიელი* უჯრების მიმდევრობას $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$, რომელშიც თითოეული i-სათვის ($0 \le i < \ell$) უჯრები (r_i, c_i) და (r_{i+1}, c_{i+1}) არიან მეზობლები.

შევნიშნოთ, რომ ℓ სიგრძის გზა შედგება $\ell + 1$ ცალი უჯრისაგან.

შეჯიბრებაზე მკვლევარებმა მოამზადეს ლაბირინთი, რომელშიც არსებობს ერთი მაინც გზა $(0, 0)$ უჯრიდან $(H - 1, W - 1)$ უჯრამდე. შევნიშნოთ, რომ უჯრები $(0, 0)$ და $(H - 1, W - 1)$ გარანტირებულად ცარიელი იქნება.

ჰანგამ არ იცის რომელი უჯრებია ცარიელი და რომელი დაბლოკილი.

თქვენი ამოცანაა დაეხმაროთ ჰანგას პულიბოტის დაპროგრამებაში, რომელსაც ექნება შესაძლებლობა *უმოკლესი გზის* (გზა, რომელსაც აქვს უმცირესი შესაძლო სიგრძე) პოვნისა $(0, 0)$-დან $(H - 1, W - 1)$-მდე უცნობ ლაბირინთში. პულიბოტის შესაძლებლობები და შეჯიბრების წესები აღწერილია ქვემოთ.

პულიბოტის სპეციფიკაცია

აღვნიშნოთ (r, c) უჯრის **მდგომარეობა** თითოეული $-1 \le r \le H$ და $-1 \le c \le W$-სთვის ერთი მთელი რიცხვით შემდეგნაირად:

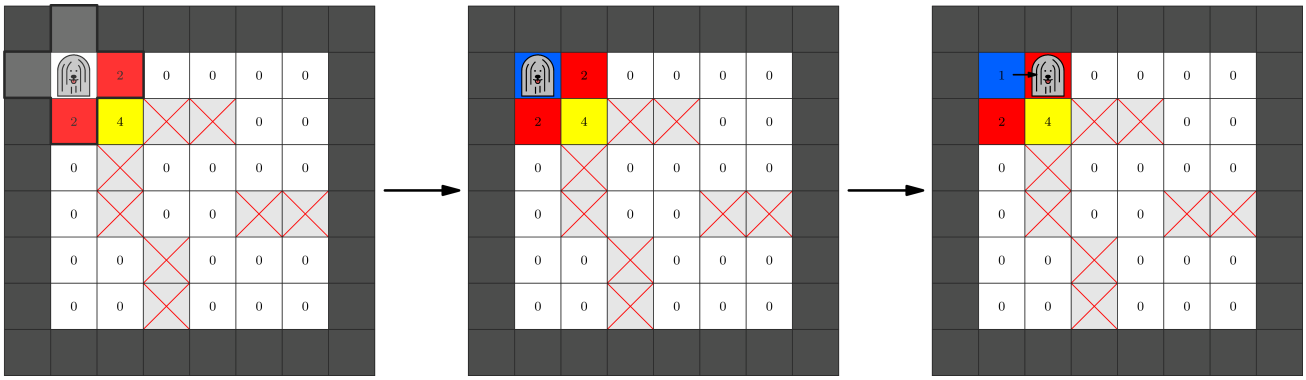
- თუ (r, c) არის საზღვრის უჯრა, მაშინ მდგომარეობა იქნება -2;
- თუ (r, c) არის დაბლოკილი უჯრა, მაშინ მდგომარეობა იქნება -1;
- თუ (r, c) არის ცარიელი უჯრა, მაშინ მდგომარეობა იქნება მისი ფერის მნიშვნელობა.

პულიბოტის პროგრამა სრულდება, როგორც ბიჯების თანმიმდევრობა. თითოეულ ბიჯზე პულიბოტი დააფიქსირებს გარშემო მყოფი უჯრების მდგომარეობებს, დაადგენს მომდევნო შესასრულებელ ინსტრუქციას და შემდეგ შეასრულებს მას. განვიხილოთ ერთი ბიჯის მსვლელობა დეტალურად. დავუშვათ, მიმდინარე ბიჯის დასაწყისში პულიბოტი განთავსებულია (r, c) უჯრაში, რომელიც არის ცარიელი. ბიჯი შესრულდება შემდეგნაირად:

1. პირველ რიგში პულიბოტი აფიქსირებს **მდგომარეობების მასივს**. ეს არის მასივი $S = [S[0], S[1], S[2], S[3], S[4]]$, რომელიც შედგება (r, c)-სა და მისი მეზობელი უჯრების მდგომარეობებისაგან:
    - $S[0]$ არის (r, c) უჯრის მდგომარეობა.
    - $S[1]$ არის მდგომარეობა უჯრისა დასავლეთით.
    - $S[2]$ არის მდგომარეობა უჯრისა სამხრეთით.
    - $S[3]$ არის მდგომარეობა უჯრისა აღმოსავლეთით.
    - $S[4]$ არის მდგომარეობა უჯრისა ჩრდილოეთით.
2. შემდეგ, მდგომარეობების მასივზე დაყრდნობით პულიბოტი დაადგენს **ინსტრუქციას** (Z, A).

3. საბოლოოდ, პულიბოტი ასრულებს ამ ინსტრუქციას: ის (r, c) უჯრის ფერის მნიშვნელობას გახდის Z-ის ტოლს და შემდეგ შეასრულებს ქმედება A-ს, რომელიც იქნება ჩამოთვლილთაგან ერთ-ერთი:
    - *დარჩენა* (r, c)-ში;
    - *გადასვლა* 4 მეზობელი უჯრიდან ერთ-ერთში;
    - *პროგრამის დასრულება*.

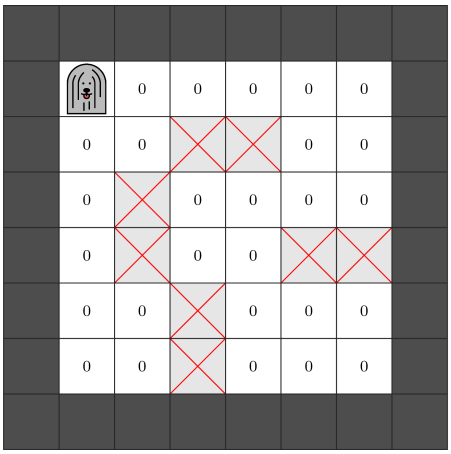
მაგალითისთვის, განვიხილოთ სიტუაცია, რომელიც მოცემულია მარცხენა ნახაზზე. პულიბოტი ამჟამად იმყოფება $(0, 0)$ უჯრაში ფერით 0. პულიბოტი აფიქსირებს მდგომარეობების მასივს $S = [0, -2, 2, 2, -2]$. პულიბოტს შეიძლება ჰქონდეს პროგრამა, რომელიც მოცემული მდგომარეობების მასივის დაფიქსირებისას გახდის მიმდინარე უჯრის ფერს $Z = 1$ და შემდეგ გადაადგილდება აღმოსავლეთით, როგორც გამოსახულია შუა და მარჯვენა ნახატებზე:

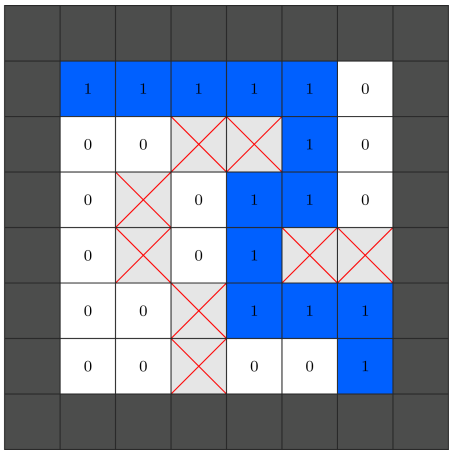


რობოტების შეჯიბრების წესები

- დასაწყისში პულიბოტი განთავსებულია $(0, 0)$ უჯრაში და იწყებს პროგრამის შესრულებას.
- პულიბოტს არ აქვს უფლება გადავიდეს უჯრაზე, რომელიც არ არის ცარიელი.
- პულიბოტის პროგრამა უნდა დასრულდეს მაქსიმუმ $500\,000$ ნაბიჯის შემდეგ.
- პროგრამის დასრულების შემდეგ ცარიელი უჯრები უნდა იყოს გაფერადებული შემდეგნაირად:
    - არსებობს უმოკლესი გზა $(0, 0)$-დან $(H - 1, W - 1)$-მდე, რომელშიც შემავალი თითოეული უჯრის ფერი არის 1.
    - ყველა დანარჩენი ცარიელი უჯრის ფერი არის 0.
- პულიბოტს შეუძლია დაასრულოს მუშაობა ნებისმიერ უჯრაში.

მაგალითისათვის, ქვემოთ მოცემული ნახაზი გვიჩვენებს ლაბირინთს, სადაც $H = W = 6$. საწყისი კონფიგურაცია გამოსახულია მარცხენა ნახაზზე და ერთ-ერთი დამაკმაყოფილებელი გაფერადება მუშაობის დასრულების შემდეგ გამოსახულია მარჯვენა ნახაზზე:





იმპლემენტაციის დეტალები

თქვენ იმპლემენტაცია უნდა გაუკეთოთ შემდეგ პროცედურას.

```
void program_pulibot()
```

- ეს პროცედურა წარმოადგენს პულიბოტის პროგრამას. პროგრამამ უნდა იმუშაოს სწორად H და W-ს ყველა მნიშვნელობისათვის, რომლებიც აკმაყოფილებენ ამოცანის შეზღუდვებს.
- ეს პროცედურა გამოძახებული იქნება ზუსტად ერთხელ თითოეული ტესტისათვის.

ამ პროცედურას შეუძლია გამოიძახოს შემდეგი პროცედურები პულიბოტის პროგრამის შესაქმნელად:

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

- S: 5 ზომის მასივი, რომელიც აღწერს მდგომარეობების მასივს.
- Z: არაუარყოფითი რიცხვი, რომელიც გამოსახავს ფერს.
- A: ერთი სიმბოლო, რომელიც გამოსახავს პულიბოტის ქმედებას შემდეგნაირად:
  - `H`: ადგილზე დარჩენა;
  - `W`: მოძრაობა დასავლეთით;
  - `S`: მოძრაობა სამხრეთით;
  - `E`: მოძრაობა აღმოსავლეთით;
  - `N`: მოძრაობა ჩრდილოეთით;
  - `T`: პროგრამის დასრულება.
- ამ პროცედურის გამოძახება პულიბოტს აძლევს ინსტრუქციას, რომ S მდგომარეობის დაფიქსირების შემთხვევაში უნდა შეასრულოს ქმედება (Z, A).

ამ პროცედურის ერთი და იგივე S-ისთვის რამდენჯერმე გამოძახების შემთხვევაში თქვენს მიერ მიღებული ვერდიქტი იქნება `Output isn't correct`.

არ არის აუცილებელი გამოიძახოთ პროცედურა `set_instruction` მდგომარეობების ყველა შესაძლო S მასივისათვის. თუმცა, თუ პულიბოტი დააფიქსირებს მდგომარეობების მასივს,

რომლისთვისაც ინსტრუქცია არ მიუღია, თქვენ მიიღებთ ვერდიქტს `Output isn't correct`.

მას შემდეგ, რაც `program_pulibot` დაასრულებს მუშაობას, გრადერი გამოიძახებს პულიბოტის პროგრამას ერთი ან მეტი ლაბირინთისათვის. ეს გამოძახებები *არ ჩაითვლება* თქვენი ამოხსნის მუშაობის დროში. გრადერი *არ არის* ადაპტიური, ანუ ლაბირინთები წინასწარ არის განსაზღვრული თითოეული ტესტისათვის.

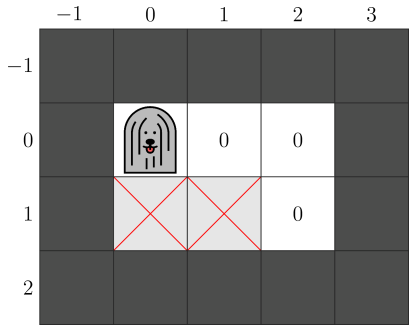
თუ პულიბოტი დაარღვევს რობოტების შეჯიბრის რომელიმე წესს მუშაობის დასრულებამდე, თქვენ მიიღებთ ვერდიქტს `Output isn't correct`.

მაგალითი

პროცედურა `program_pulibot`-ს შეუძლია გამოიძახოს `set_instruction` შემდეგნაირად:

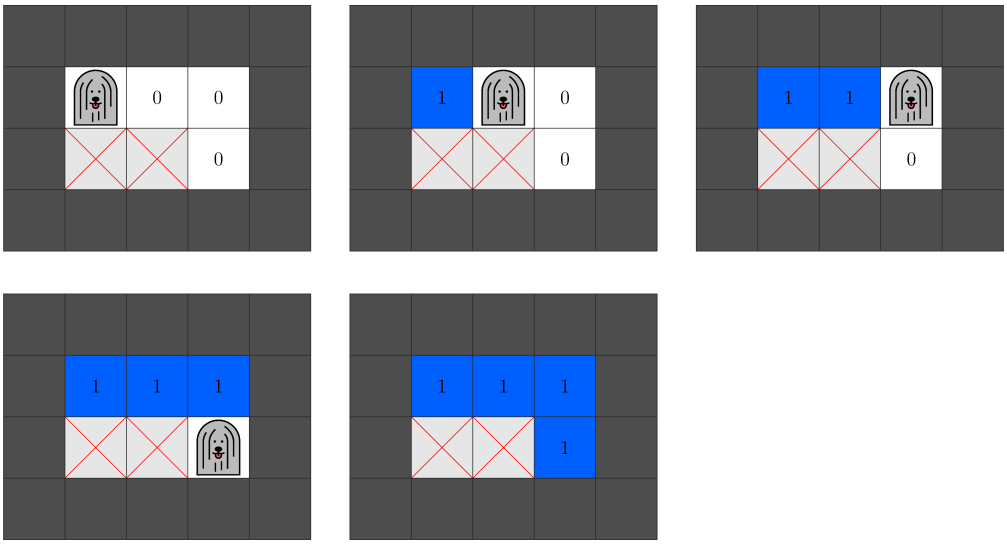
გამოძახება	ინსტრუქცია S მდგომარეობისთვის
`set_instruction([0, -2, -1, 0, -2], 1, E)`	გახადე უჯრის ფერი 1 და გადადი აღმოსავლეთით
`set_instruction([0, 1, -1, 0, -2], 1, E)`	გახადე უჯრის ფერი 1 და გადადი აღმოსავლეთით
`set_instruction([0, 1, 0, -2, -2], 1, S)`	გახადე უჯრის ფერი 1 და გადადი სამხრეთით
`set_instruction([0, -1, -2, -2, 1], 1, T)`	გახადე უჯრის ფერი 1 და დაასრულე მუშაობა

განვიხილოთ სიტუაცია როცა $H = 2$ და $W = 3$ და ლაბირინთი გამოსახულია ნახაზზე.



ამ კონკრეტული ლაბირინთისათვის პულიბოტის პროგრამა შეასრულებს 4 ბიჯს. მდგომარეობები, რომლებსაც პულიბოტი აფიქსირებს და შესაბამისი ქმედებები აღწერილია ზემოთ მოცემულ ცხრილში იმ თანმიმდევრობით, რომლითაც მათი შესრულება მოუწევს. ბოლო ქმედება დაასრულებს პროგრამის მუშაობას.

ქვემოთ მოცემული ნახაზი გვიჩვენებს მთელი ლიბირინთის მდგომარეობას, თითოეულ ბიჯამდე და ბოლოს მდგომარეობას მუშაობის დასრულების შემდეგ.



თუმცა, შევნიშნოთ, რომ პროგრამამ მხოლოდ 4 ინსტრუქციით შეიძლება ვერ იპოვოს უმოკლესი გზა სხვა ვალიდური ლაბირინთებისათვის. შესაბამისად, ასეთი გაშვების შემთხვევაში, თქვენი მიღებული ვერდიქტი იქნება `Output isn't correct`.

შეზღუდვები

$Z_{MAX} = 19$. ანუ, პულიბოტს შეუძლია გამოიყენოს ფერები 0-დან 19-ის ჩათვლით.

თითოეული ლაბირინთისათვის, რომელზეც შეიძლება გაიტესტოს პულიბოტი:

- $2 \leq H, W \leq 15$
- არსებობს ერთი გზა მაინც $(0, 0)$-დან $(H - 1, W - 1)$-მდე.

ქვეამოცანები

1. (6 ქულა) ლაბირინთი არ შეიცავს დაბლოკილ უჯრებს.
2. (10 ქულა) $H = 2$.
3. (18 ქულა) არსებობს ზუსტად ერთი გზა ცარიელი უჯრების ნებისმიერ წყვილს შორის.
4. (20 ქულა) უმოკლეს გზას $(0, 0)$-დან $(H - 1, W - 1)$-მდე აქვს სიგრძე $H + W - 2$.
5. (46 ქულა) დამატებითი შეზღუდვების გარეშე.

თუ რომელიმე ტესტისათვის `set_instruction`-ის გამოძახებები ან პულიბოტის პროგრამა არ დააკმაყოფილებს იმპლემენტაციის დეტალებში მოცემულ შეზღდვებს, ქვეამოცანაში თქვენი ამოხსნის მიერ მიღებული ქულა იქნება 0.

თითოეულ ქვეამოცანაში შეგიძლიათ მიიღოთ ნაწილობრივი ქულა სწორთან ახლოს მყოფი გაფერადების მიღების შემთხვევაში.

ფორმალურად:

- ტესტის ამოხსნა არის **სრულყოფილი**, თუ ცარიელი უჯრების საბოლოო გაფერადება აკმაყოფილებს რობოტების შეჯიბრების წესებს.

- ტესტის ამოხსნა არის **ნაწილობრივი**, თუ საბოლოო გაფერადება გამოიყურება შემდეგნაირად:
  - არსებობს უმოკლესი გზა $(0, 0)$-დან $(H-1, W-1)$-მდე, რომლის თითოეული უჯრის ფერი არის 1.
  - სხვა არცერთი ცარიელი უჯრის ფერი არ არის 1.
  - რომელიმე ცარიელ უჯრას აქვს 0 და 1-სგან განსხვავებული ფერი.

თუ თქვენი ამოხსნა ტესტზე არ არის არც სრული და არც ნაწილობრივი, თქვენ მიიღებთ 0 ქულას.

1-4 ქვეამოცანებში თქვენ მიიღებთ ქულების 100%-ს, თუ თქვენი ამოხსნა არის სრული და ქულების 50%-ს, თუ თქვენი ამოხსნა არის ნაწილობრივი.

მე-5 ქვეამოცანაში თქვენი ქულა დამოკიდებულია თქვენს მიერ გამოყენებული ფერების რაოდენობაზე. უფრო ზუსტად, $Z^\star$-ით აღვნიშნოთ Z-ის მაქსიმალური მნიშვნელობა `set_instruction`-ის ყველა გამოძახებას შორის. თქვენი ქულა გამოითვლება მოცემული ცხრილის მიხედვით:

პირობა	ქულა (სრული)	ქულა (ნაწილობრივი)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

თითოეული ქვეამოცანის ქულა არის მინიმალური ამ ქვეამოცანის ტესტებში მიღებულ ქულებს შორის.

სანიმუშო გრადერი

სანიმუშო გრადერი კითხულობს ინფორმაციას შემდეგნაირად:

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

აქ m არის H ზომის მასივი შემდგარი W ზომის მთელი რიცხვების მასივებისგან, რომლებიც აღწერენ ლაბირინთის უჯრებს საზღვრების გამოკლებით. $m[r][c] = 0$, თუ (r, c) არის ცარიელი უჯრა და $m[r][c] = 1$, თუ (r, c) არის დაბლოკილი.

სანიმუშო გრადერი ჯერ გამოიძახებს `program_pulibot()`-ს. თუ სანიმუშო გრადერი დააფიქსირებს პროტოკოლის დარღვევას, ის გამოიტანს `Protocol Violation: <MSG>` და დაასრულებს მუშაობას, სადაც `<MSG>` იქნება ერთ-ერთი შემდეგთაგანი:

- `Invalid array`: არ სრულდება $-2 \leq S[i] \leq Z_{MAX}$ რომელიმე i-სთვის ან S-ის ზომა არ არის 5.
- `Invalid color`: არ სრულდება $0 \leq Z \leq Z_{MAX}$.
- `Invalid action`: სიმბოლო A განსხვავებულია `H`, `W`, `S`, `E`, `N` და `T` სიმბოლოებისგან.
- `Same state array`: `set_instruction` იქნა გამოძახებული ერთი და იგივე S მასივისთვის ორჯერ ან მეტჯერ.

წინააღმდეგ შემთხვევაში, როცა `program_pulibot` დაასრულებს მუშაობას, სანიმუშო გრადერი შეასრულებს პულიბოტის პროგრამას მიწოდებული ლაბირინთისათვის.

სანიმუშო გრადერს გამოაქვს ორი შედეგი.

ჯერ სანიმუშო გრადერი ბეჭდავს პულიბოტის ქმედებების ჩანაწერებს ფაილში `robot.bin` სამუშაო დირექტორიაში. ეს ფაილი მოქმედებს ინპუტად ქვემოთ აღწერილი ვიზუალიზაციის ინსტრუმენტისათვის.

შემდეგ, თუ პულიბოტის პროგრამა არ დაასრულებს მუშაობას წარმატებით, სანიმუშო გრადერი ბეჭდავს ჩამოთვლილთაგან ერთ-ერთ შეტყობინებას:

- `Unexpected state`: პულიბოტმა დააფიქსირა მდგომარეობა, რომლისთვისაც `set_instruction` არ იყო გამოძახებული.
- `Invalid move`: ქმედება დასრულდა არაცარიელ უჯრაში გადასვლით.
- `Too many steps`: პულიბოტმა შეასრულა $500\,000$-ზე მეტი ბიჯი მუშაობის დასრულებამდე.

წინააღმდეგ შემთხვევაში, $e[r][c]$-ით აღვნიშნოთ (r, c) უჯრის მდგომარეობა პულიბოტის პროგრამის დასრულების შემდეგ. სანიმუშო გრადერი დაბეჭდავს H სტრიქონს შემდეგი ფორმატით:

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

ვიზუალიზაციის ინსტრუმენტი

ამოცანაზე მიბმული ფოლდერი შეიცავს ფაილს `display.py`. გამოძახების შემთხვევაში ეს პითონის სკრიპტი გამოსახავს პულიბოტის ქმედებებს სანიმუშო გრადერისთვის მიწოდებული მონაცემების მიხედვით. ამისთვის დირექტორიაში უნდა არსებობდეს ფაილი `robot.bin`.

სკრიპტის გამოსაძახებლად შეასრულეთ შემდეგი ბრძანება.

```
python3 display.py
```

ეკრანზე გამოვა მარტივი გრაფიკული ინტერფეისი. მთავარი ფუნქციები შემდეგია:

- შეგიძლიათ დააკვირდეთ მთლიანი ლაბირინთის მდგომარეობას. პულიბოტის მიმდინარე მდებარეობა იქნება მონიშნული მართკუთხედით.
- შეგიძლიათ დაათვალიეროთ პულიბოტის ნაბიჯები ისრის ღილაკებზე დაჭერით, ან გადახტეთ კონკრეტულ ნაბიჯზე.

- მომავალი ბიჯი პულიბოტის პროგრამაში გამოსახულია ფანჯრის ქვედა ნაწილში. ის აჩვენებს მიმდინარე მდგომარეობების მასივს და ინსტრუქციას, რომელსაც პულიბოტი შეასრულებს. საბოლოო ბიჯის შემდეგ ნაჩვენები იქნება ან გრადერის მიერ გამოტანილი შენიშვნა, ან `Terminated`, თუ პროგრამა წარმატებით დასრულდა.
- თითოეულ რიცხვს, რომელიც შეესაბამება ფერს, შეგიძლიათ მიანიჭოთ ფონის ფერი და გამოსატანი ტექსტი. გამოსატანი ტექსტი მოკლე სტრინგია, რომელიც ჩაიწერება ყველა უჯრაში, რომელსაც ეს ფერი აქვს. ფონის ფერებისა და გამოსატანი ტექსტის მინიჭება შეგიძლიათ შემდეგი გზით:
  - მიანიჭეთ ისინი დიალოგის ფანჯარაში `Colors` ღილაკზე დაჭერის შემდეგ.
  - შეცვალეთ `colors.txt` ფაილის შიგთავსი.
- `robot.bin` ფაილის განსაახლებლად, გამოიყენეთ `Reload` ღილაკი. ეს იქნება გამოსადეგი, თუ `robot.bin` ფაილის შიგთავსი შეიცვალა.