



Robot Contest (Διαγωνισμός Ρομποτικής)

Οι ερευνητές Τεχνητής Νοημοσύνης στο Πανεπιστήμιο του Szeged διοργανώνουν διαγωνισμό προγραμματισμού ρομπότ. Η φίλη σας, η Hanga, αποφάσισε να λάβει μέρος στο διαγωνισμό. Στόχος είναι να προγραμματίσει το απόλυτο *Pulibot*, θαυμάζοντας τη μεγάλη ευφυΐα της διάσημης ουγγρικής φυλής ποιμενικών σκύλων Puli.

Το Pulibot θα δοκιμαστεί σε έναν λαβύρινθο που αποτελείται από ένα grid (πλέγμα) αποτελούμενο από $(H + 2) \times (W + 2)$ κελιά.

Οι γραμμές του grid αριθμούνται από -1 έως H από βορρά προς νότο και οι στήλες του πλέγματος αριθμούνται από -1 έως W από δύση προς ανατολή.

Αναφερόμαστε στο κελί που βρίσκεται στη γραμμή r και στη στήλη c του grid ($-1 \leq r \leq H$, $-1 \leq c \leq W$) ως κελί (r, c) .


Θεωρήστε ένα κελί (r, c) τέτοιο ώστε $0 \leq r < H$ και $0 \leq c < W$. Υπάρχουν 4 κελιά **γειτονικά** στο κελί (r, c) :

- το κελί $(r, c - 1)$ αναφέρεται ως το κελί **δυτικά** του κελιού (r, c) ,
- το κελί $(r + 1, c)$ αναφέρεται ως το κελί **νότια** του κελιού (r, c) ,
- το κελί $(r, c + 1)$ αναφέρεται ως το κελί **ανατολικά** του κελιού (r, c) ,
- το κελί $(r - 1, c)$ αναφέρεται ως το κελί **βόρεια** του κελιού (r, c) .

Το κελί (r, c) ονομάζεται **οριακό** κελί του λαβύρινθου αν ισχύει $r = -1$ ή $r = H$ ή $c = -1$ ή $c = W$. Κάθε κελί που δεν είναι οριακό κελί του λαβύρινθου είναι είτε **κελί-εμπόδιο** είτε **κενό κελί**. Επιπλέον, κάθε κενό κελί έχει ένα **χρώμα**, που αντιπροσωπεύεται από έναν μη αρνητικό ακέραιο μεταξύ 0 και Z_{MAX} , συμπεριλαμβανομένου.

Αρχικά, το χρώμα κάθε κενού κελιού είναι 0.

Για παράδειγμα, θεωρήστε έναν λαβύρινθο με $H = 4$ και $W = 5$, που περιέχει ένα μόνο κελί-εμπόδιο $(1, 3)$:

	-1	0	1	2	3	4	5
-1							
0		0	0	0	0	0	
1		0	0	0		0	
2		0	0	0	0	0	
3		0	0	0	0	0	
4							

Το μόνο κελί-εμπόδιο συμβολίζεται με σταυρό. Τα οριακά κελιά του λαβύρινθου είναι σκιασμένα. Οι αριθμοί που αναγράφονται σε κάθε κενό κελί αντιπροσωπεύουν τα αντίστοιχα χρώματά τους.

Ένα **μονοπάτι** μήκους ℓ ($\ell > 0$) από το κελί (r_0, c_0) στο κελί (r_ℓ, c_ℓ) είναι μια ακολουθία από, ανά ζεύγη διαφορετικά, κενά κελιά $(r_0, c_0), (r_1, c_1), \dots, (r_\ell, c_\ell)$ στην οποία για κάθε i ($0 \leq i < \ell$) τα κελιά (r_i, c_i) και (r_{i+1}, c_{i+1}) είναι γειτονικά.

Σημειώστε ότι ένα μονοπάτι μήκους ℓ περιέχει ακριβώς $\ell + 1$ κελιά.

Στον διαγωνισμό, οι ερευνητές έστησαν έναν λαβύρινθο στον οποίο υπάρχει τουλάχιστον ένα μονοπάτι από το κελί $(0, 0)$ στο κελί $(H - 1, W - 1)$. Σημειώστε ότι αυτό σημαίνει ότι τα κελιά $(0, 0)$ και $(H - 1, W - 1)$ είναι εγγυημένα κενά.

Η Hanga δεν γνωρίζει ποια κελιά του λαβύρινθου είναι κενά και ποια κελιά είναι κελιά-εμπόδια.

Ο στόχος σας είναι να βοηθήσετε την Hanga να προγραμματίσει το Pulibot έτσι ώστε να είναι ικανό να βρει ένα *κοντινότερο μονοπάτι* (δηλαδή ένα μονοπάτι ελάχιστου μήκους) από το κελί $(0, 0)$ στο κελί $(H - 1, W - 1)$ στον άγνωστο λαβύρινθο που έχουν στήσει οι ερευνητές. Οι προδιαγραφές του Pulibot και οι κανόνες του διαγωνισμού περιγράφονται παρακάτω.

Σημειώστε ότι η τελευταία ενότητα αυτού του προβλήματος περιγράφει ένα εργαλείο εμφάνισης που μπορείτε να χρησιμοποιήσετε για να οπτικοποιήσετε το Pulibot.

Προδιαγραφές του Pulibot

Ορίζουμε την **κατάσταση** ενός κελιού (r, c) για κάθε $-1 \leq r \leq H$ και $-1 \leq c \leq W$ ως έναν ακέραιο αριθμό έτσι ώστε:

- αν το κελί (r, c) είναι οριακό κελί τότε η κατάστασή του είναι -2 ,
- αν το κελί (r, c) είναι κελί-εμπόδιο τότε η κατάστασή του είναι -1 ,
- αν το κελί (r, c) είναι ένα κενό κελί τότε η κατάστασή του είναι το χρώμα του κελιού.

Το πρόγραμμα του Pulibot εκτελείται ως μια ακολουθία βημάτων. Σε κάθε βήμα, το Pulibot αναγνωρίζει τις καταστάσεις των κοντινών κελιών και στη συνέχεια εκτελεί μια εντολή. Η εντολή

που εκτελεί καθορίζεται από τις αναγνωρισμένες καταστάσεις. Ακολουθεί μια πιο ακριβής περιγραφή.

Ας υποθέσουμε ότι στην αρχή του τρέχοντος βήματος, το Pulibot βρίσκεται στο κελί (r, c) , το οποίο είναι ένα κενό κελί. Το βήμα εκτελείται ως εξής:

1. Κατ' αρχάς, το Pulibot αναγνωρίζει τον τρέχοντα πίνακα **καταστάσεων**, δηλαδή τον πίνακα $S = [S[0], S[1], S[2], S[3], S[4]]$, που αποτελείται από την κατάσταση του κελιού (r, c) και όλων των γειτονικών κελιών:
 - $S[0]$ είναι η κατάσταση του κελιού (r, c) .
 - $S[1]$ είναι η κατάσταση του κελιού που βρίσκεται δυτικά.
 - $S[2]$ είναι η κατάσταση του κελιού προς τα νότια.
 - $S[3]$ είναι η κατάσταση του κελιού προς τα ανατολικά.
 - $S[4]$ είναι η κατάσταση του κελιού προς βορρά.
2. Στη συνέχεια, το Pulibot προσδιορίζει την **εντολή** (Z, A) που αντιστοιχεί στον αναγνωρισμένο πίνακα καταστάσεων.
3. Τέλος, το Pulibot εκτελεί αυτή την εντολή: θέτει το χρώμα του κελιού (r, c) στο χρώμα Z . και στη συνέχεια εκτελεί την ενέργεια A , η οποία είναι μία από τις ακόλουθες ενέργειες:
 - παραμένει στο κελί (r, c) ,
 - μετακινείται σε ένα από τα 4 γειτονικά κελιά,
 - τερματίζει το πρόγραμμα.

Για παράδειγμα, θεωρήστε το σενάριο που εμφανίζεται στα αριστερά του ακόλουθου σχήματος. Το Pulibot βρίσκεται αυτή τη στιγμή στο κελί $(0, 0)$ με χρώμα 0. Το Pulibot αναγνωρίζει τον πίνακα καταστάσεων $S = [0, -2, 2, 2, -2]$. Το Pulibot μπορεί να έχει ένα πρόγραμμα το οποίο, μόλις αναγνωρίσει αυτόν τον πίνακα, θέτει το χρώμα του τρέχοντος κελιού σε $Z = 1$ και στη συνέχεια κινείται προς τα ανατολικά, όπως φαίνεται στη μέση και στα δεξιά του σχήματος:

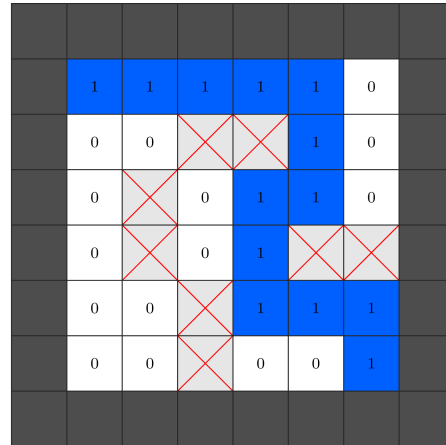
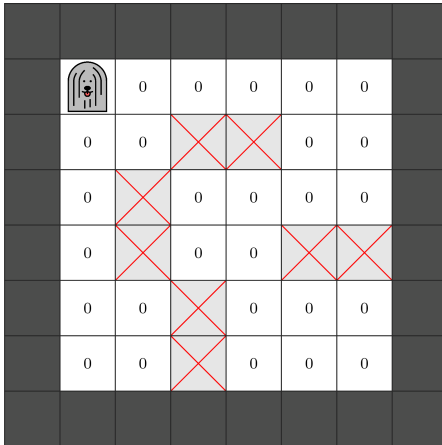
![Actions of a step of Pulibot](robot-actions.png "700")

Κανόνες Διαγωνισμού Ρομποτικής

- Στην αρχή, το Pulibot τοποθετείται στο κελί $(0, 0)$ και αρχίζει να εκτελεί το πρόγραμμά του.
- Το Pulibot δεν επιτρέπεται να μετακινηθεί σε κελί που δεν είναι κενό.
- Το πρόγραμμα του Pulibot πρέπει να τερματιστεί μετά από το πολύ 500 000 βήματα.
- Μετά τον τερματισμό του προγράμματος του Pulibot, τα κενά κελιά στο λαβύρινθο πρέπει να χρωματιστούν έτσι ώστε:
 - Να υπάρχει ένα συντομότερο μονοπάτι από $(0, 0)$ σε $(H - 1, W - 1)$ για το οποίο το χρώμα κάθε κελιού που περιλαμβάνεται στο μονοπάτι είναι 1.
 - Το χρώμα κάθε άλλου κενού κελιού να είναι 0.
- Το Pulibot μπορεί να τερματίσει το πρόγραμμά του σε οποιοδήποτε κενό κελί.

Για παράδειγμα, το ακόλουθο σχήμα δείχνει έναν πιθανό λαβύρινθο με $H = W = 6$. Η αρχική διαμόρφωση εμφανίζεται στα αριστερά και ο αναμενόμενος χρωματισμός των κενών κελιών μετά

τον τερματισμό εμφανίζεται στα δεξιά:



Λεπτομέρειες Υλοποίησης

Θα πρέπει να εφαρμόσετε την ακόλουθη συνάρτηση.

```
void program_pulibot()
```

- Αυτή η συνάρτηση θα πρέπει να δημιουργεί το πρόγραμμα του Pulibot. Αυτό το πρόγραμμα θα πρέπει να λειτουργεί σωστά για όλες τις τιμές των H και W και για κάθε λαβύρινθο που πληροί τους περιορισμούς του προβλήματος.
- Αυτή η συνάρτηση καλείται ακριβώς μία φορά για κάθε test case.

Αυτή η συνάρτηση μπορεί να καλεί την ακόλουθη συνάρτηση για την δημιουργία του προγράμματος του Pulibot:

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

- S : πίνακας μήκους 5 που περιγράφει έναν πίνακα καταστάσεων.
- Z : ένας μη αρνητικός ακέραιος που αντιπροσωπεύει ένα χρώμα.
- A : ένας απλός χαρακτήρας που αντιπροσωπεύει μια ενέργεια του Pulibot ως εξής:
 - H: παραμονή,
 - W: μετακίνηση προς τα δυτικά,
 - S: μετακίνηση προς τα νότια,
 - E: μετακίνηση προς τα ανατολικά,
 - N: μετακίνηση προς το βορρά,
 - T: τερματισμός του προγράμματος.
- Η κλήση αυτής της συνάρτησης δίνει εντολή στο Pulibot ότι μόλις αναγνωρίσει την κατάσταση S θα πρέπει να εκτελέσει την εντολή (Z, A) .

Η κλήση αυτής της συνάρτησης πολλές φορές με την ίδια κατάσταση S θα έχει ως απάντηση το `Output isn't correct`.

Δεν απαιτείται η κλήση της εντολής `set_instruction` με κάθε πιθανό πίνακα καταστάσεων S . Ωστόσο, αν το Pulibot αναγνωρίσει αργότερα μια κατάσταση για την οποία δεν έχει οριστεί μια εντολή, θα έχει ως απάντηση το `Output isn't correct`.

Αφού ολοκληρωθεί το `program_pulibot`, ο grader καλεί το πρόγραμμα του Pulibot σε έναν ή περισσότερους λαβύρινθους. Αυτές οι κλήσεις δεν υπολογίζονται στο χρονικό όριο για τη λύση σας. Ο grader δεν είναι adaptive (προσαρμοστικός), δηλαδή το σύνολο των λαβυρίνθων είναι προκαθορισμένο σε κάθε test case.

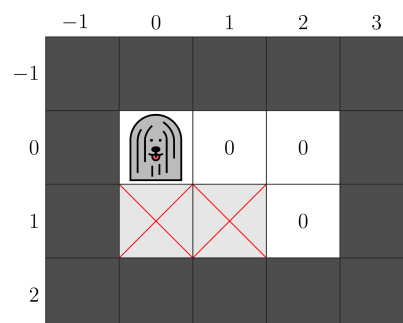
Εάν το Pulibot παραβιάσει οποιονδήποτε από τους κανόνες του διαγωνισμού ρομπότ πριν τερματίσει το πρόγραμμά του, θα έχει ως απάντηση το `Output isn't correct`.

Παράδειγμα

Η συνάρτηση `program_pulibot` μπορεί να κάνει κλήσεις στην εντολή `set_instruction` ως εξής:

Κλήση	Εντολή για τον πίνακα καταστάσεων S
<code>set_instruction([0, -2, -1, 0, -2], 1, E)</code>	Θέσε το χρώμα σε 1 και μετακινήσου ανατολικά
<code>set_instruction([0, 1, -1, 0, -2], 1, E)</code>	Θέσε το χρώμα σε 1 και μετακινήσου ανατολικά
<code>set_instruction([0, 1, 0, -2, -2], 1, S)</code>	Θέσε το χρώμα σε 1 και μετακινήσου νότια
<code>set_instruction([0, -1, -2, -2, 1], 1, T)</code>	Θέσε το χρώμα σε 1 και τερμάτισε το πρόγραμμα

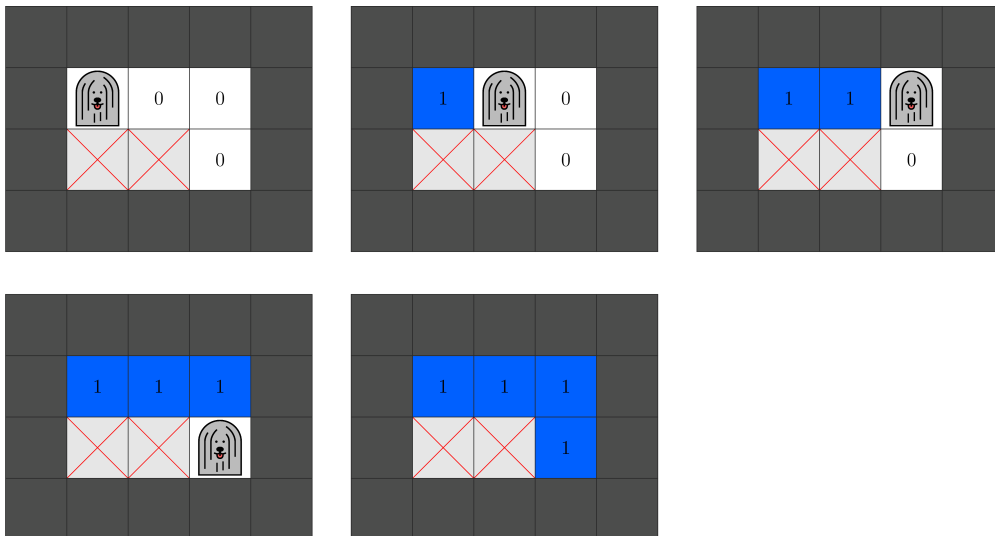
Θεωρήστε ένα σενάριο όπου $H = 2$ και $W = 3$, και ο λαβύρινθος απεικονίζεται στο ακόλουθο σχήμα.



Για τον συγκεκριμένο λαβύρινθο το πρόγραμμα του Pulibot εκτελείται σε τέσσερα βήματα. Οι πίνακες καταστάσεων που αναγνωρίζει το Pulibot και οι εντολές που εκτελεί αντιστοιχούν

ακριβώς στις τέσσερις κλήσεις της εντολής `set_instruction` που γίνονται παραπάνω, με τη σειρά. Η τελευταία από αυτές τις εντολές τερματίζει το πρόγραμμα.

Το ακόλουθο σχήμα δείχνει την κατάσταση του λαβύρινθου πριν από κάθε ένα από τα τέσσερα βήματα και την τελική του κατάσταση μετά τον τερματισμό.



Ωστόσο, σημειώστε ότι αυτό το πρόγραμμα που αποτελείται από 4 εντολές μπορεί να μην βρίσκει το συντομότερο μονοπάτι σε άλλους έγκυρους λαβύρινθους. Επομένως, αν υποβληθεί, θα έχει ως απάντηση το `Output isn't correct`.

Περιορισμοί

$Z_{MAX} = 19$. Επομένως, το Pulibot μπορεί να χρησιμοποιήσει χρώματα από 0 έως 19, συμπεριλαμβανομένων.

Για κάθε λαβύρινθο που χρησιμοποιείται για τη δοκιμή του Pulibot:

- $2 \leq H, W \leq 15$
- Υπάρχει τουλάχιστον ένα μονοπάτι από το κελί $(0, 0)$ στο κελί $(H - 1, W - 1)$.

Subtasks (υποπροβλήματα)

1. (6 πόντοι) Δεν υπάρχει κανένα κελί-εμπόδιο στο λαβύρινθο.
2. (10 πόντοι) $H = 2$
3. (18 πόντοι) Υπάρχει ακριβώς ένα μονοπάτι μεταξύ κάθε ζεύγους κενών κελιών.
4. (20 πόντοι) Το συντομότερο μονοπάτι από το κελί $(0, 0)$ στο κελί $(H - 1, W - 1)$ έχει μήκος $H + W - 2$.
5. (46 βαθμοί) Χωρίς επιπλέον περιορισμούς.

Εάν, σε οποιοδήποτε test case, οι κλήσεις της συνάρτησης `set_instruction` ή το πρόγραμμα του Pulibot κατά την εκτέλεση δεν συμμορφώνονται με τους περιορισμούς που περιγράφονται

στην ενότητα Λεπτομέρειες Υλοποίησης, η βαθμολογία της λύσης σας για το συγκεκριμένο υποπρόβλημα θα είναι 0.

Σε κάθε υποπρόβλημα, μπορείτε να λάβετε μερική βαθμολογία δημιουργώντας έναν χρωματισμό που είναι σχεδόν σωστός.

Τυπικά:

- Η λύση μιας δοκιμασίας είναι **ολοκληρωμένη** αν ο τελικός χρωματισμός των κενών κελιών ικανοποιεί τους Κανόνες Διαγωνισμού Ρομποτικής.
- Η λύση μιας δοκιμασίας είναι **μερική** εάν ο τελικός χρωματισμός έχει την ακόλουθη μορφή:
 - Υπάρχει ένα συντομότερο μονοπάτι από το $(0, 0)$ στο $(H - 1, W - 1)$ για το οποίο το χρώμα κάθε κελιού που περιλαμβάνεται στο μονοπάτι είναι 1.
 - Δεν υπάρχει άλλο κενό κελί στο πλέγμα με χρώμα 1.
 - Κάποιο κενό κελί στο πλέγμα έχει χρώμα διαφορετικό από 0 και 1.

Εάν η λύση σας σε ένα test case δεν είναι ούτε πλήρης ούτε μερική, η βαθμολογία σας για το test case θα είναι 0.

Στα subtasks 1-4, η βαθμολογία για μια πλήρη λύση είναι 100% και η βαθμολογία για μια μερική λύση σε ένα test case είναι το 50% των βαθμών του subtask.

Στο subtask 5, η βαθμολογία σας εξαρτάται από τον αριθμό των χρωμάτων που χρησιμοποιούνται στο πρόγραμμα του Pulibot. Πιο συγκεκριμένα, συμβολίστε με Z^* τη μέγιστη τιμή του Z σε όλες τις κλήσεις που έγιναν στην `set_instruction`. Η βαθμολογία του test case υπολογίζεται σύμφωνα με τον ακόλουθο πίνακα:

Συνθήκη	Βαθμολογία (πλήρης)	Βαθμολογία (μερική)
$11 \leq Z^* \leq 19$	$20 + (19 - Z^*)$	$12 + (19 - Z^*)$
$Z^* = 10$	31	23
$Z^* = 9$	34	26
$Z^* = 8$	38	29
$Z^* = 7$	42	32
$Z^* \leq 6$	46	36

Η βαθμολογία για κάθε subtask είναι το ελάχιστο των βαθμών για τα test cases του subtask.

Ενδεικτικός Grader

Ο ενδεικτικός grader διαβάσει την είσοδο με την ακόλουθη μορφή:

- γραμμή 1: $H \ W$

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

Εδώ, m είναι ένας πίνακας H πινάκων W ακεραίων αριθμών, που περιγράφει τα μη οριακά κελιά του λαβύρινθου. $m[r][c] = 0$ αν το κελί (r, c) είναι κενό κελί και $m[r][c] = 1$ αν το κελί (r, c) είναι κελί-εμπόδιο.

Ο ενδεικτικός Grader καλεί πρώτα το `program_pulibot()`. Εάν ο ενδεικτικός Grader εντοπίσει παραβίαση του πρωτοκόλλου εκτυπώνει `Protocol Violation: <MSG>` και τερματίζει, όπου `<MSG>` είναι ένα από τα ακόλουθα μηνύματα σφάλματος:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ δεν πληρείται για κάποιο i ή το μήκος του S δεν είναι 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ δεν πληρείται.
- `Invalid action`: ο χαρακτήρας A δεν είναι ένας από τους H, W, S, E, N or T.
- `Same state array`: `set_instruction` κλήθηκε με τον ίδιο πίνακα S τουλάχιστον δύο φορές.

Διαφορετικά, όταν ολοκληρωθεί το `program_pulibot`, ο ενδεικτικός Grader εκτελεί το πρόγραμμα του Pulibot στο λαβύρινθο που περιγράφεται από την είσοδο.

Ο ενδεικτικός Grader παράγει δύο εξόδους. Πρώτον, ο ενδεικτικός Grader γράφει ένα αρχείο καταγραφής των ενεργειών του Pulibot στο αρχείο `robot.bin` στον κατάλογο εργασίας. Αυτό το αρχείο χρησιμοποιείται ως είσοδος για το εργαλείο οπτικοποίησης που περιγράφεται στην επόμενη ενότητα.

Δεύτερον, εάν το πρόγραμμα του Pulibot δεν τερματιστεί επιτυχώς, ο ενδεικτικός Grader εκτυπώνει ένα από τα ακόλουθα μηνύματα σφάλματος:

- `Unexpected state`: Το Pulibot αναγνώρισε έναν πίνακα καταστάσεων με τον οποίο δεν κλήθηκε η εντολή `set_instruction`.
- `Invalid move`: η εκτέλεση μιας ενέργειας είχε ως αποτέλεσμα το Pulibot να μετακινηθεί σε ένα μη κενό κελί.
- `Too many steps`: Το Pulibot εκτέλεσε 500 000 βήματα χωρίς να τερματίσει το πρόγραμμά του.

Διαφορετικά, $e[r][c]$ να είναι η κατάσταση του κελιού (r, c) μετά τον τερματισμό του προγράμματος του Pulibot. Ο ενδεικτικός Grader εκτυπώνει H γραμμές με την ακόλουθη μορφή:

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

Εργαλείο εμφάνισης

Στα επισυναπτόμενα αυτού του προβλήματος περιέχει ένα αρχείο με όνομα `display.py`. Όταν καλείται, αυτό το script της Python εμφανίζει τις ενέργειες του Pulibot στο λαβύρινθο που

περιγράφεται από την είσοδο του ενδεικτικού Grader. Για να γίνει αυτό πρέπει το δυαδικό αρχείο `robot.bin` να βρίσκεται στον κατάλογο εργασίας.

Για να κληθεί το script, εκτελέστε την ακόλουθη εντολή.

```
python3 display.py
```

Εμφανίζεται ένα απλό γραφικό περιβάλλον. Τα κύρια χαρακτηριστικά του είναι τα επόμενα:

- Μπορείτε να παρατηρήσετε την κατάσταση ολόκληρου του λαβυρίνθου. Η τρέχουσα θέση του Pulibot επισημαίνεται από ένα ορθογώνιο.
- Μπορείτε να περιηγηθείτε στα βήματα του Pulibot κάνοντας κλικ στα κουμπιά με τα βέλη ή πατώντας τα πλήκτρα συντόμευσής τους. Μπορείτε επίσης να μεταβείτε σε ένα συγκεκριμένο βήμα.
- Το επερχόμενο βήμα στο πρόγραμμα του Pulibot εμφανίζεται στο κάτω μέρος. Δείχνει τον πίνακα της τρέχουσας κατάστασης και την εντολή που θα εκτελέσει. Μετά το τελευταίο βήμα, εμφανίζει είτε ένα από τα μηνύματα σφάλματος του Grader, είτε Terminated αν το πρόγραμμα έχει τερματιστεί επιτυχώς.
- Σε κάθε αριθμό που αντιπροσωπεύει ένα χρώμα, μπορείτε να αντιστοιχίσετε ένα οπτικό χρώμα φόντου, καθώς και ένα κείμενο εμφάνισης. Το κείμενο εμφάνισης είναι μια σύντομη συμβολοσειρά που θα εμφανίζεται σε κάθε κελί που έχει αυτό το χρώμα. Μπορείτε να εκχωρήσετε χρώματα φόντου και κείμενα εμφάνισης με έναν από τους ακόλουθους τρόπους:
 - Να τα ορίσετε σε ένα παράθυρο διαλόγου αφού κάνετε κλικ στο κουμπί Colors.
 - Να επεξεργαστείτε τα περιεχόμενα του αρχείου `colors.txt`.
- Για να επαναφορτώσετε (reload) το `robot.bin`, χρησιμοποιήστε το κουμπί Reload. Είναι χρήσιμο εάν τα περιεχόμενα του αρχείου `robot.bin` έχουν αλλάξει.