Οδόμετρο με χαλίκια

Ο Leonardo εφεύρε το αρχικό *οδόμετρο*: ένα καρότσι που μπορούσε να μετρήσει αποστάσεις ρίχνοντας χαλίκια καθώς γύριζαν οι ρόδες του. Στη συνέχεια μετρώντας τα χαλίκια έβρισκε τον αριθμό των στροφών που έκανε η ρόδα, δίνοντας τη δυνατότητα στο χρήστη να υπολογίζει την απόσταση που διανύθηκε από το οδόμετρο. Σαν επιστήμονες της Πληροφορικής, προσθέσαμε ένα λογισμικό ελέγχου στο οδόμετρο, για να επεκτείνει τις λειτουργίες του. Το πρόβλημά σας είναι να προγραμματίσετε το οδόμετρο χρησιμοποιώντας τους πιο κάτω κανόνες.

Πλέγμα λειτουργίας

Το οδόμετρο κινείται σε ένα φανταστικό τετραγωνικό πλέγμα αποτελούμενο από 256 × 256 κελιά. Κάθε κελί μπορεί να περιέχει το πολύ 15 χαλίκια και προσδιορίζεται από ένα ζεύγος συντεταγμένων (σειρά, στήλη), όπου κάθε συντεταγμένη βρίσκεται στο διάστημα 0, …, 255. Δεδομένου ενός κελιού (i, j), τα γειτονικά κελιά (εάν υπάρχουν) είναι (i - 1, j), (i + 1, j), (i, j - 1) και (i, j + 1). Οποιοδήποτε κελί που βρίσκεται στην πρώτη ή τελευταία σειρά, ή στην πρώτη ή τελευταία στήλη, καλείται *σύνορο*. Το οδόμετρο αρχίζει πάντα από το κελί (0, 0) (η βορειοδυτική γωνία), και κοιτάζει προς το Βορρά.

Βασικές εντολές

Το οδόμετρο μπορεί να προγραμματιστεί χρησιμοποιώντας τις ακόλουθες εντολές.

- `left` — στρίψε 90 μοίρες αριστερά (αριστερόστροφα) παραμένοντας στο τρέχον κελί (π.χ. μετά από την εκτέλεση της εντολής, εάν προηγουμένως αντίκρυζε το νότο, μετά θα αντικρύζει την ανατολή).
- `right` — στρίψε 90 μοίρες δεξιά (δεξιόστροφα) παραμένοντας στο τρέχον κελί (π.χ. μετά από την εκτέλεση της εντολής, εάν προηγουμένως αντίκρυζε τη δύση, μετά θα αντικρύζει το Βορρά).
- `move` — μετακινήσου μια μονάδα προς τα εμπρός (στην κατεύθυνση που βλέπει το οδόμετρο) σε ένα γειτονικό κελί. Εάν δεν υπάρχει κελί (δηλαδή το οδόμετρο έχει ήδη φτάσει στα σύνορα σε εκείνη την κατεύθυνση) αυτή η εντολή δεν έχει κανένα αποτέλεσμα.
- `get` — αφαίρεσε ένα χαλίκι από το τρέχον κελί. Εάν το τρέχον κελί δεν έχει κανένα χαλίκι, τότε αυτή η εντολή δεν έχει κανένα αποτέλεσμα.
- `put` — πρόσθεσε ένα χαλίκι στο τρέχον κελί. Εάν το τρέχον κελί περιέχει 15 χαλίκια, τότε η εντολή δεν έχει κανένα αποτέλεσμα. Από το οδόμετρο δεν θα λείψουν ποτέ τα χαλίκια.
- `halt` — τερμάτισε την εκτέλεση.

Το οδόμετρο εκτελεί τις εντολές με τη σειρά που δίνονται στο πρόγραμμα, μια εντολή για κάθε γραμμή. Άδειες γραμμές αγνοούνται. Το σύμβολο `#` σημαίνει σχόλιο, ό,τι κείμενο ακολουθεί, μέχρι το τέλος της γραμμής, δεν λαμβάνεται υπόψη. Αν το οδόμετρο φτάσει στο τέλος του προγράμματος, η εκτέλεση σταματά.

Παράδειγμα 1

Ας υποθέσουμε το ακόλουθο πρόγραμμα για το οδόμετρο. Παίρνει το οδόμετρο στο κελί (0, 2) να αντικρύζει την ανατολή. (Σημειώστε ότι η πρώτη εντολή `move` αγνοείται, επειδή το οδόμετρο είναι στη βορειοδυτική γωνία και βλέπει προς το Βορρά.)

```
move  # κανένα αποτέλεσμα
right
# τώρα το οδόμετρο βλέπει ανατολικά
move
move
```

Ετικέτες, σύνορα και χαλίκια

Για να αλλάξετε τη ροή του προγράμματος ανάλογα με την παρούσα κατάσταση, μπορείτε να χρησιμοποιήσετε τις ετικέτες, που αποτελούνται από το πολύ 128 σύμβολα μεταξύ των `a`, …, `z`, `A`, …, `Z`, `0`, …, `9` και για τις οποίες τα μικρά και τα κεφαλαία γράμματα διαφέρουν. Οι νέες εντολές που αφορούν τις ετικέτες παρατίθενται πιο κάτω. Στις περιγραφές πιο κάτω, *`L`* παριστάνει οποιαδήποτε έγκυρη ετικέτα.

- *`L`*`:` (δηλαδή *`L`* που ακολουθείται από μια άνω και κάτω τελεία '`:`') — δηλώνει τη θέση μέσα στο πρόγραμμα μιας ετικέτας *`L`*. Όλες οι δηλωμένες ετικέτες πρέπει να είναι μοναδικές. Η δήλωση μιας ετικέτας δεν έχει καμία επίδραση στο οδόμετρο.

- `jump` *`L`* — συνέχισε την εκτέλεση με άλμα χωρίς συνθήκη (unconditional jump) στη γραμμή με την ετικέτα *`L`*.

- `border` *`L`* — συνέχισε την εκτέλεση με άλμα στη γραμμή με την ετικέτα *`L`*, εάν το οδόμετρο βρίσκεται στα σύνορα και κοιτάζει προς την άκρη του πλέγματος (δηλαδή η οδηγία `move` δεν θα είχε κανένα αποτέλεσμα). Διαφορετικά, η εκτέλεση συνεχίζεται κανονικά και αυτή η εντολή δεν έχει κανένα αποτέλεσμα.

- `pebble` *`L`* — συνέχισε την εκτέλεση με άλμα στη γραμμή με την ετικέτα *`L`*, εάν το τρέχον κελί περιέχει τουλάχιστον ένα χαλίκι. Διαφορετικά, η εκτέλεση συνεχίζεται κανονικά και αυτή η εντολή δεν έχει κανένα αποτέλεσμα.

Παράδειγμα 2

Το ακόλουθο πρόγραμμα βρίσκει το πρώτο (δυτικότερο) χαλίκι στη σειρά 0 και σταματά εκεί. Εάν δεν υπάρχει κανένα χαλίκι στη σειρά 0, σταματά στα σύνορα στο τέλος της σειράς. Χρησιμοποιεί δύο ετικέτες `leonardo` και `davinci`.

```
right
leonardo:
pebble davinci  # βρέθηκε χαλίκι
border davinci  # τέλος γραμμής
move
jump leonardo
davinci:
halt
```

Το οδόμετρο αρχίζει στρίβοντας δεξιά. Ο βρόχος αρχίζει με τη δήλωση της ετικέτας `leonardo:` και τελειώνει με την εντολή `jump leonardo`. Μέσα στο βρόχο, το οδόμετρο ελέγχει για την παρουσία ενός χαλικιού ή για τα σύνορα στο τέλος της σειράς. Αν κανένα από τα δύο δε συμβαίνει, το οδόμετρο εκτελεί `move` από το τρέχον κελί (0, j) στο γειτονικό κελί (0, j + 1) αφού έχει βεβαιωθεί ότι αυτό υπάρχει. (Η εντολή `halt` δεν είναι αναγκαία καθώς το πρόγραμμα τερματίζει ούτως ή άλλως.)

Πρόβλημα

Πρέπει να υποβάλετε ένα πρόγραμμα στη γλώσσα του οδόμετρου, όπως περιγράφεται παραπάνω, που να κάνει το οδόμετρο να συμπεριφέρεται όπως ζητείται. Κάθε υποπρόβλημα (δείτε παρακάτω) περιγράφει μία συμπεριφορά που πρέπει να έχει το οδόμετρο και τους περιορισμούς που πρέπει να ικανοποιεί η υποβληθείσα λύση. Οι περιορισμοί αφορούν στα εξής.

- *Μέγεθος προγράμματος* — το πρόγραμμα πρέπει να είναι αρκετά μικρό. Το μέγεθος του προγράμματος είναι το πλήθος των εντολών σε αυτό. Οι δηλώσεις ετικετών, τα σχόλια και οι κενές γραμμές *δε μετρούν* στο μέγεθος.

- *Μήκος εκτέλεσης* — το πρόγραμμα πρέπει να τελειώνει αρκετά γρήγορα. Το μήκος εκτέλεσης είναι το πλήθος των *βημάτων* που εκτελούνται: κάθε εκτέλεση μίας εντολής μετρά ως ένα βήμα, ανεξάρτητα αν η εντολή έχει ή όχι αποτέλεσμα. Οι δηλώσεις ετικετών, τα σχόλια και οι κενές γραμμές δε μετρούν ως βήματα.

Στο Παράδειγμα 1, το μέγεθος του προγράμματος είναι 4 και το μήκος εκτέλεσης είναι 4. Στο Παράδειγμα 2, το μέγεθος του προγράμματος είναι 6 και, όταν εκτελείται σε ένα πλέγμα με ένα μόνο χαλίκι στο κελί (0, 10), το μήκος εκτέλεσης είναι 43 βήματα: `right`, 10 επαναλήψεις του βρόχου, κάθε επανάληψη απαιτεί 4 βήματα (`pebble davinci`; `border davinci`; `move`; `jump leonardo`), και τέλος, `pebble davinci` και `halt`.

Υποπρόβλημα 1 [9 βαθμοί]

Στην αρχή υπάρχουν x χαλίκια στο κελί (0, 0) και y στο κελί (0, 1), ενώ όλα τα άλλα κελιά είναι κενά. Θυμηθείτε ότι μπορούν να υπάρχουν το πολύ 15 χαλίκια σε κάθε κελί. Γράψτε ένα πρόγραμμα που τερματίζει με το οδόμετρο στο κελί (0, 0) αν $x \leq y$, και στο κελί (0, 1) διαφορετικά. (Δε μας ενδιαφέρει η κατεύθυνση στην οποία κοιτάζει το οδόμετρο στο τέλος. Επίσης, δε μας ενδιαφέρει πόσα χαλίκια υπάρχουν στο τέλος πάνω στο πλέγμα ή πού βρίσκονται αυτά.)

Όρια: μέγεθος προγράμματος ≤ 100, μήκος εκτέλεσης $\leq 1\,000$.

Υποπρόβλημα 2 [12 βαθμοί]

Το ίδιο όπως το παραπάνω υποπρόβλημα, αλλά όταν το πρόγραμμα τελειώσει, το κελί (0, 0) πρέπει να περιέχει ακριβώς x χαλίκια και το κελί (0, 1) πρέπει να περιέχει ακριβώς y χαλίκια.

Όρια: μέγεθος προγράμματος ≤ 200, μήκος εκτέλεσης ≤ 2 000.

Υποπρόβλημα 3 [19 βαθμοί]

Υπάρχουν ακριβώς δύο χαλίκια κάπου στη γραμμή 0: το ένα είναι στο κελί (0, x), το άλλο στο κελί (0, y); τα x και y είναι διαφορετικά, και το x + y είναι ζυγός αριθμός. Γράψτε ένα πρόγραμμα που αφήνει το οδόμετρο στο κελί (0, (x + y) / 2), δηλαδή ακριβώς στο μέσο ανάμεσα στα δύο κελιά που περιέχουν τα χαλίκια. Η τελική κατάσταση του πλέγματος δε μας ενδιαφέρει.

Όρια: μέγεθος προγράμματος ≤ 100, μήκος εκτέλεσης ≤ 200 000.

Υποπρόβλημα 4 [μέχρι 32 βαθμοί]

Υπάρχουν το πολύ 15 χαλίκια στο πλέγμα και δεν υπάρχουν δύο ή περισσότερα στο ίδιο κελί. Γράψτε ένα πρόγραμμα που συλλέγει όλα τα χαλίκια στη βορειοδυτική γωνία. Πιο συγκεκριμένα, αν στην αρχή υπάρχουν στο πλέγμα x χαλίκια, στο τέλος πρέπει να υπάρχουν ακριβώς x χαλίκια στο κελί (0, 0) και κανένα χαλίκι πουθενά αλλού.

Η βαθμολογία για αυτό το υποπρόβλημα εξαρτάται από το μήκος εκτέλεσης του προγράμματος που θα υποβάλετε. Ακριβέστερα, αν L είναι το μέγιστο μήκος εκτέλεσης των διαφόρων περιπτώσεων ελέγχου, η βαθμολογία σας θα είναι:

- 32 βαθμοί αν $L \leq 200\,000$;
- $32 - 32 \log_{10}(L / 200\,000)$ βαθμοί αν $200\,000 < L < 2\,000\,000$;
- 0 βαθμοί αν $L \geq 2\,000\,000$.

Όρια: μέγεθος προγράμματος ≤ 200.

Υποπρόβλημα 5 [μέχρι 28 βαθμοί]

Μπορεί να υπάρχουν οσαδήποτε χαλίκια σε κάθε κελί του πλέγματος (φυσικά, μεταξύ 0 και 15). Γράψτε ένα πρόγραμμα που βρίσκει το ελάχιστο, δηλαδή που τερματίζει με το οδόμετρο σε ένα κελί (i, j) τέτοιο ώστε κάθε άλλο κελί να περιέχει τουλάχιστον τόσα χαλίκια όσα και το (i, j). Μετά την εκτέλεση του προγράμματος, το πλήθος των χαλικιών σε κάθε κελί πρέπει να είναι το ίδιο που ήταν και πριν την εκτέλεση του προγράμματος.

Η βαθμολογία για αυτό το υποπρόβλημα εξαρτάται από το μέγεθος του προγράμματος που θα υποβάλετε. Πιο συγκεκριμένα, η βαθμολογία σας θα είναι:

- 28 βαθμοί αν $P \leq 444$,

- $28 - 28 \log_{10}(P / 444)$ βαθμοί αν $444 < P < 4\,440$,
- 0 βαθμοί αν $P \geq 4\,440$.

Όρια: μήκος εκτέλεσης ≤ 44 400 000.

Λεπτομέρειες υλοποίησης

Πρέπει να υποβάλετε ακριβώς ένα αρχείο ανά υποπρόβλημα, γραμμένο σύμφωνα με τους συντακτικούς κανόνες που περιγράφηκαν παραπάνω. Κάθε υποβαλόμενο αρχείο μπορεί να έχει μέγιστο μήκος 5 MiB. Για κάθε υποπρόβλημα, ο κώδικας του οδομέτρου σας θα ελεχθεί με μερικές περιπτώσεις ελέγχου και θα λάβετε ορισμένες πληροφορίες για τους πόρους που χρησιμοποιεί ο κώδικάς σας. Σε περίπτωση που ο κώδικας δεν είναι συνακτικά σωστός και άρα δεν είναι δυνατό να ελεγχθεί, θα λάβετε πληροφορίες σχετικές με το συντακτικό σφάλμα.

Δεν είναι αναγκαίο οι υποβολές σας να περιέχουν προγράμματα του οδόμετρου για όλα τα υποπροβλήματα. Αν η τρέχουσα υποβολή δεν περιέχει πρόγραμμα για το υποπρόβλημα X, η πιο πρόσφατη υποβολή σας για το υποπρόβλημα X συμπεριλαμβάνεται αυτόματα. Αν δεν υπάρχει τέτοιο πρόγραμμα, το υποπρόβλημα αυτό θα λάβει μηδενική βαθμολογία για αυτή την υποβολή.

Όπως συνήθως, η βαθμολογία μίας υποβολής είναι το άθροισμα των βαθμολογιών κάθε υποπροβλήματος και η τελική βαθμολογία είναι η μέγιστη βαθμολογία μεταξύ των υποβολών που ελέγχθηκαν πλήρως και της τελευταίας υποβολής.

Προσομοιωτής

Για τον έλεγχο του προγράμματός σας, σας δίνεται ένας προσομοιωτής οδόμετρου, τον οποίον μπορείτε να τροφοδοτείτε με τα προγράμματα και τα αρχικά πλέγματα που θέλετε. Τα προγράμματα πρέπει να γράφονται στην ίδια μορφή που χρησιμοποιείται για τις υποβολές (δηλαδή, αυτή που περιγράφηκε παραπάνω).

Οι περιγραφές του πλέγματος θα δίνονται στην ακόλουθη μορφή: κάθε γραμμή του αρχείου θα περιέχει τρεις αριθμούς, R, C και P, που σημαίνουν ότι το κελί στη γραμμή R και τη στήλη C περιέχει P χαλίκια. Όλα τα κελιά που δεν αναφέρονται στην περιγραφή του πλέγματος θεωρούνται κενά. Για παράδειγμα, έστω το αρχείο:

```
0 10 3
4 5 12
```

Το πλέγμα που περιγράφεται από αυτό το αρχείο περιέχει 15 χαλίκια: 3 στο κελί (0, 10) και 12 στο κελί (4, 5).

Μπορείτε να καλέσετε τον προσομοιωτή εκτελώντας το πρόγραμμα `simulator.py` στο directory του προβλήματος, δίνοντάς του το όνομα του προγράμματος. Ο προσομοιωτής δέχεται τις εξής επιλογές στη γραμμή εντολών:

- `-h` δίνει μία μικρή σύνοψη των διαθέσιμων επιλογών,

- `-g GRID_FILE` φορτώνει την περιγραφή του πλέγματος από το αρχείο `GRID_FILE` (default: κενό πλέγμα),
- `-s GRID_SIDE` καθορίζει το μέγεθος του πλέγματος σε `GRID_SIDE x GRID_SIDE` (default: 256, όπως χρησιμοποιείται στην περιγραφή του προβλήματος); η χρήση μικρότερων πλεγμάτων μπορεί να είναι χρήσιμη για τον εντοπισμό σφαλμάτων,
- `-m STEPS` περιορίζει το πλήθος των βημάτων εκτέλεσης της προσομοίωσης σε το πολύ `STEPS` βήματα,
- `-c` εισάγει σε κατάσταση μεταγλώττισης. Στην κατάσταση αυτή, ο προσομοιωτής επιστρέφει ακριβώς την ίδια έξοδο, όμως αντί να κάνει την προσομοίωση με Python, γεννά και μεταγλωττίζει ένα μικρό πρόγραμμα σε C. Αυτό προκαλεί μία μεγάλη καθυστέρηση όταν αρχίζει, δίνει όμως αρκετά γρηγορότερα αποτελέσματα. Η χρήση του συνιστάται όταν το πρόγραμμά σας αναμένεται να εκτελέσει πάνω από 10 000 000 βήματα.

Αριθμός υποβολών

Ο μέγιστος αριθμός υποβολών που επιτρέπεται να γίνουν σε αυτό το πρόβλημα είναι 128.