Odometar s kamenčićima

Leonardo je izmislio prvi *odometar*: kolica koja mjere prijeđenu udaljenost tako da baca kamenčiće kako se kolica kreću. Brojeći kamenčiće znamo koliko su se puta okrenuo kotač, što nam omogućuje da izračunamo prijeđenu udaljenost. Kao računalni znanstvenici dodali smo programsku podršku odometru i na taj način proširili njegove funkcionalnosti. Vaš zadatak je da napišete program koji će kontrolirati odometar pod dolje zadanim uvjetima.

Polje kretanja

Odometar se kreće po imaginarnom kvadratnom polju veličine 256 x 256 ćelija. Svaka ćelija može sadržavati najviše 15 kamenčića i određena je s parom koordinata (red, stupac), gdje je svaka koordinata označena cijelim brojem od 0 do 255, uključivo. Za neku ćeliju (i, j), ćelije (i - 1, j), (i + 1, j), (i, j - 1) i (i, j + 1) su susjedne (ukoliko postoje). Bilo koju ćeliju koja leži na prvom ili zadnjem retku, odnosno prvom ili zadnjem stupcu nazivamo *rubnom*. Odometar uvijek počinje na ćeliji (0, 0) (sjeverno-zapadni ugao) i usmjeren je prema sjeveru.

Osnovne naredbe

Odometar možemo programirati da prati sljedeće naredbe.

- `left` — okreni se 90 stupnjeva ulijevo i ostani u trenutnoj ćeliji. Npr. ako je odometar prije bio usmjeren prema jugu, sad je usmjeren prema istoku.
- `right` — okreni se 90 stupnjeva udesno i ostani u trenutnoj ćeliji. Npr. ako je odometar prije bio usmjeren prema zapadu, sad je usmjeren prema sjeveru.
- `move` — pokreni se u susjednu ćeliju u trenutnom smjeru odometra. Ako takva ćelija ne postoji (tj. odometar je već na rubnoj ćeliji u tom smjeru), ova naredba nema efekta.
- `get` — makni jedaj kamenčić sa trenutne ćelije. Ako trenutna ćelija nema kamenčića, ova naredba nema efekta.
- `put` — dodaj jedan kamenčić na trenutnu ćeliju. Ako trenutna ćelija već ima 15 kamenčića, ova naredba nema efekta. Također, odometru nikad ne može ponestati kamenčića.
- `halt` — završi izvođenje.

Odometar izvodi ove naredbe redom kojim su zadane u programu. Program se mora sastojati od najviše jedne naredbe po liniji (prazne linije su ignorirane). Znak # označava komentar; bilo koji tekst koji prati taj znak do kraja reda je ignoriran. Ako odometar izvrši posljednju naredbu (ne nužno `halt`), program završava izvođenje.

Primjer 1

Promotrite sljedeći program za odometar. Program vodi odometar na ćeliju (0, 2) i ostavlja ga usmjerenog prema istoku. Prvi `move` korak je ignoriran pošto je odometar na početku usmjeren prema sjeveru.

```
move  # nema efekta
right
# odometar je sad usmjeren prema istoku
move
move
```

Oznake, rubovi i kamenčići

Kako bi mogli kontrolirati tok programa ovisno o trenutnom stanju možete koristiti oznake, odnosno stringove (osjetljive na velika i mala slova) čija je maksimalna duljina 128. Oznake se sastoje isključivo od malih i velikih engleskih slova `a`, …, `z`, `A`, …, `Z` te znamenki `0`, …, `9`. Nove naredbe koje se tiču oznaki su objašnjene u nastavku. U tim objašnjenjima, *`L`* označava bilo koju ispravnu oznaku.

- *`L`*`:` (tj. **`L`** popraćeno znakom '`:`') — označava trenutnu lokaciju u programu s oznakom L. Sve deklarirane oznake se moraju međusobno razlikovati. Deklaracija oznake nikako ne utječe na odometar.
- `jump` *`L`* — nastavi izvođenje s neuvjetovanim skokom na oznaku *`L`*.
- `border` *`L`* — ako se odometar nalazi na rubnom polju tako da je usmjeren prema rubu (tj. ako u tom trenutku naredba `move` ne bi imala učinka), nastavi izvođenje sa skokom na oznaku *`L`*. U protivnom ova naredba nema učinka.
- `pebble` *`L`* — ako se na trenutnoj ćeliji nalazi barem jedan kamenčić, nastavi izvođenje sa skokom na oznakom *`L`*. U protivnom ova naredba nema učinka.

Prvi primjer

Sljedeći program pronalazi prvi (najzapadniji) kamenčić u redu 0 i u toj ćeliji prekida izvođenje. Ako u redu 0 nema kamenčića, program prekida izvođenje na rubu reda 0. Program koristi dvije oznake `leonardo` i `davinci`.

```
right
leonardo:
pebble davinci  # kamenčić je pronađen
border davinci  # kraj reda
move
jump leonardo
davinci:
halt
```

Odometar počinje svoje izvođenje sa okretom udesno. Početak petlje je određen s deklaracijom oznake `leonardo:` i prestaje s naredbom `jump leonardo`. U petlji, odometar provjerava za prisutnost kamenčića ili kraja reda i tada prekida izvođenje skokom na oznaku `davinci`, a u protivnom se kreće naredbom `move` iz ćelije (0, j) u susjednu ćeliju (0, j + 1). (Naredba `halt` je tu

nepotrebna pošto bi program ionako završio izvođenje).

Zadatak

Vaša je zadatak napisati program u gore određenom programskom jeziku odometra, koji će upravljati odometrom na određen način. Svaki podzadatak (vidi dolje) određuje kako se odometar mora ponašati i koja ograničenja mora ispuniti. Dvije su vrste ograničenja:

- *Veličina programa* — program mora biti dovoljno kratak. Veličina programa je određena brojem naredbi u programu, ne brojeći deklaracije oznaki, komentare i prazne linije.
- *Trajanje izvođenja* — program mora završiti izvođenje dovoljno brzo. Trajanje izvođenja je određeno brojem koraka: svaka naredba se broji kao korak, bez obzira ima li učinka ili ne. Također, deklaracije oznaki, komentari i prazne linije se ne broje kao koraci.

U prvom primjeru, veličina programa je 4 i trajanje izvođenja 4 koraka. U drugom primjeru, duljina programa je 6, dok je trajanje izvođenja 43 koraka ako postoji samo jedan kamenčić u ćeliji (0, 10): `right`, zatim 10 iteracija petlja po 4 koraka (`pebble davinci`; `border davinci`; `move`; `jump leonardo`) te konačno, `pebble davinci` i `halt`.

Podzadatak 1 [9 bodova]

Na početku postoji x kamenčića u ćeliji (0, 0) i y kamenčića u čeliji (0, 1), dok su sve ostale ćelije prazne. Nemojte zaboraviti da u svakoj ćeliji može biti najviše 15 kamenčića u bilo kojem trenutku. Napišite program koji prekida izvođenje s odometrom u ćeliji (0, 0) ako je $x \leq y$, a inače u ćeliji (0, 1). Smjer odometra na kraju nije bitan, kao ni konačan broj kamenčića u bilo kojoj ćeliji polja.

Ograničenja: veličina programa ≤ 100, trajanje izvođenja ≤ 1 000.

Podzadatak 2 [12 bodova]

Isti zadatak kao i podzadatak 1, samo na kraju izvođenja ćelija (0, 0) mora imati točno x kamenčića, a ćelija (0, 1) točno y kamenčića.

Ograničenja: veličina programa ≤ 200, trajanje izvođenja ≤= 2 000.

Podzadatak 3 [19 bodova]

Postoje točno dva kamenčića u redu 0: jedan u ćeliji (0, x) i jedan u ćeliji (0, y). Pritom su x i y različiti brojevi i x + y je paran broj. Napišite program koji prekida izvođenje s odometrom u ćeliji (0, (x + y) / 2), tj. točno između dva kamenčića. Konačno stanje polja nije bitno.

Ograničenja: veličina programa ≤ 100, trajanje izvođenja ≤ 200 000.

Podzadatak 4 [najviše 32 boda]

Na cijelom polju je najviše 15 kamenčića, a svaka ćelija sadrži najviše jedan. Napišite program koji će se kamenčiće skupiti u sjeverno-zapadni ugao: bolje rečeno, ako na početku postoji točno x kamenčića u cijelom polju, na kraju mora biti točno x kamenčića u ćeliji (0, 0) i nijedan kamenčić u svim ostalim ćelijama.

Broj bodova za ovaj podzadatak ovisi o trajanju izvođenja programa. Točnije, ako je L najveća duljina trajanja na nekom skupu test podataka, vaši bodovi će biti:

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

Ograničenja: veličina programa ≤ 200.

Potprogram 5 [najviše 28 bodova]

Na polju može biti proizvoljan broj kamenčića (u svakoj ćeliji maksimalno 15). Napišite program koji prekida izvođenje s odometrom u ćeliji s najmanje kamenčića, odnosno takvoj ćeliji da svaka druga ćelija sadrži najmanje toliko kamenčića. Nakon izvođenja programa, broj kamenčića u svakoj ćeliji mora biti jednak broju kamenčića prije izvođenja.

Broj bodova za ovaj podzadatak ovisi o duljini programa P vašeg programa:

- 28 bodova ako je $P \leq 444$;
- $28 - 28 \log_{10} (P / 444)$ ako je $444 < P < 4\,440$;
- 0 bodova ako je $P \geq 4\,440$.

Ograničenja: trajanje izvođenja ≤ 44 400 000.

Implementacija

Za svaki podzadatak morate napisati točno jednu datoteku, napisanu prema pravilima gore definiranog jezika. Svaka datoteka koju predate može biti najveće veličine 5 MiB. Za svaki podzadatak, vaš program za odometar će biti testiran na više test primjera, a vi ćete dobiti povratnu informaciju o resursima koje troši vaš program. Ukoliko program nije sintaksno točan, dobit ćete odgovarajuću poruku koja indicira grešku.

Nije nužno da svaki put kad šaljete rješenja, šaljete programe za sve podzadatke. Ako ne uključite rješenje za neki podzadatak X, vaše prethodno zadnje poslano rješenje za podzadatak X će i dalje biti aktivno. Ako takvog programa nema, imat ćete 0 bodova za taj podzadatak.

Kao i inače, broj bodova za neko rješenje je zbroj bodova dobivenih za svaki podzadatak. Konačni broj bodova je najveći broj bodova između svih rješenja za koje ste potrošili token ili zadnjeg poslanog rješenja.

Simulator

For testing purposes, you are provided with an odometer simulator, which you can feed with your programs and input grids. Odometer programs will be written in the same format used for submission (i.e., the one described above).

Polje opisujete u sljedećem formatu: svaka linija datoteke mora sadržavati tri broja R, C i P, što označava da ćelija (R, C) ima P kamenčića. Za sve ćelije koje nisu opisane se pretpostavlja da nemaju kamenčića. Npr. promotrite sljedeću datoteku:

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

Polje opisano ovom datotekom bi imalo ukupno 15 kamenčića: 3 u ćeliji (0, 10) i 12 u ćeliji (4, 5).

Simulator možete pokrenuti tako da pozovete program `simulator.py` u direktoriju zadatka. Pritom mu morate predati ime datoteke s programom kao argument. Simulator može primiti sljedeće opcije putem komandne linije:

- `-h` daje kratak opis mogućih opcija;
- `-g GRID_FILE` otvara opis polja iz datoteke `GRID_FILE` (default: prazno polje);
- `-s GRID_SIDE` postavlja veličinu polja na `GRID_SIDE x GRID_SIDE` (default: 256, kao u opisu zadatka); uporaba manjih veličina može biti korisna za testiranje;
- `-m STEPS` ograničava broj koraka programa na najviše `STEPS` koraka;
- `-c` koristi kompajlerski mod, u kojem simulator vraća isti izlaz, samo umjesto da koristi Python, prevodi program u C. To uzrokuje određeno početno usporenje, no može biti puno brže za programe za koje očekujute da će trajati više od 10 000 000 koraka.

Broj slanja rješenja

Rješenje možete slati najviše 128 puta.