Odometar s kamenčićima

Upravo Leonardo je izmislio prvi *odometar*: kolica koja mjere pređenu udaljenost tako da baca kamenčiće kako se kolica kreću. Brojeći kamenčiće znamo koliko su se puta okrenuo točak, što nam omogućava da izračunamo pređenu udaljenost. Kao računarski naučnici dodali smo programsku podršku odometru i na taj način proširili njegove funkcionalnosti. Vaš zadatak je da napišete program koji će kontrolisati odometar pod dole zadanim uslovima.

Polje kretanja

Odometar se kreće po zamišljenom kvadratnom polju veličine 256 x 256 ćelija. Svaka ćelija može sadržavati najviše 15 kamenčića i određena je parom koordinata (red, kolona) 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 redu, odnosno prvoj ili zadnjoj koloni nazivamo *ivičnom*. Odometar uvijek počinje na ćeliji (0, 0) (sjeverozapadni ugao) i usmjeren je prema sjeveru.

Osnovne naredbe

Odometar se može programirati sljedećim naredbama.

- `left` — okreni se 90 stepeni ulijevo (suprotno smijeru kazaljke na satu) i ostani u trenutnoj ćeliji. Npr. ako je odometar prije bio usmjeren prema jugu, sad je usmjeren prema istoku.
- `right` — okreni se 90 stepeni udesno (u smijeru kazaljke na satu) i ostani u trenutnoj ćeliji. Npr. ako je odometar prije bio usmjeren prema zapadu, sad je usmjeren prema sjeveru.
- `move` — pomjeri se u susjednu ćeliju u trenutnom smjeru odometra. Ako takva ćelija ne postoji (tj. odometar je već na ivičnoj ćeliji u tom smjeru), ova naredba nema efekta.
- `get` — ukloni jedan kamenčić iz 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. Odometru nikad ne može ponestati kamenčića.
- `halt` — završi izvođenje.

Odometar izvodi ove naredbe redom kojim su zadane u programu. Program smije sadržavati najviše jednu naredbu po liniji. Znak # označava komentar; bilo koji tekst koji prati taj znak do kraja reda je ignorisan. Ako odometar izvrši posljednju naredbu, program završava izvođenje.

Primjer 1

Posmatrajmo sljedeći program za odometar. Program vodi odometar na ćeliju (0, 2) i ostavlja ga usmjerenog prema istoku. Prvi `move` korak je ignorisan 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 kontrolisati tok programa ovisno o trenutnom stanju, možete koristiti oznake koje predstavljaju stringove (osjetljive na velika i mala slova) čija je maksimalna dužina 128. Oznake se sastoje isključivo od malih i velikih slova engleskog alfabeta `a`, …, `z`, `A`, …, `Z` i cifara `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 deklarisane oznake se moraju međusobno razlikovati. Deklaracija oznake nikako ne utiče na odometar.
- `jump` *`L`* — nastavi izvođenje s bezuslovnim skokom na liniju označenu oznakom *`L`*.
- `border` *`L`* — ako se odometar nalazi na ivičnom polju tako da je usmjeren prema ivici (tj. ako u tom trenutku naredba `move` ne bi imala efekta), nastavi izvođenje sa skokom na liniju označenu oznakom *`L`*. U protivnom, ova naredba nema efekta.
- `pebble` *`L`* — ako se na trenutnoj ćeliji nalazi barem jedan kamenčić, nastavi izvođenje sa skokom na oznaku *`L`*. U protivnom, ova naredba nema efekta.

Primjer 2

Sljedeći program pronalazi najljevlji kamenčić u redu 0 i u toj ćeliji prekida izvođenje. Ako u redu 0 nema kamenčića, program prekida izvođenje na ivici 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 postojanje kamenčića ili kraja reda i tada prekida izvođenje skokom na labelu `davinci`, a u protivnom se kreće naredbom `move` iz ćelije (0, j) u susjednu ćeliju (0, j + 1) obzirom da ona postoji. (Naredba `halt` je tu nepotrebna pošto bi program ionako završio izvođenje).

Postavka

Vaš zadatak je napisati program u gore određenom programskom jeziku odometra koji će upravljati odometrom na određen način. Svaki podzadatak (vidi dole) 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.

- *Dužina izvršavanja* — program mora završiti izvođenje dovoljno brzo. Dužina izvršavanja određena je brojem koraka: svaka naredba se broji kao korak, bez obzira ima li efekta ili ne. Deklaracije oznaka, komentari i prazne linije se ne broje kao koraci.

U prvom primjeru, veličina programa je 4 i trajanje izvođenja 4 koraka. U drugom primjeru, dužina 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`) i na kraju, `pebble davinci` i `halt`.

Podzadatak 1 [9 bodova]

Na početku postoji x kamenčića na ćeliji (0, 0) i y na ćeliji (0, 1), dok su sve ostale ćelije prazne. Napomenimo da može biti najviše 15 kamenčića na jednoj ćeliji. Napišite program koji završava kada je odometar u ćeliji (0, 0) ako je $x \leq y$, odnosno u ćeliji (0, 1) u suprotnom. (Smjer u kojem je odometar okrenut na kraju nije bitan. Osim toga, nije bitno koliko je kamenčića na kraju na polju niti gdje se oni nalaze.)

Ograničenja: veličina programa ≤ 100, dužina izvršavanja ≤ 1 000.

Podzadatak 2 [12 bodova]

Isto kao i podzadatak iznad sa razlikom što kada se program završi, ćelija (0, 0) treba da sadrži tačno x kamečića, a ćelija (0, 1) tačno y kamečića.

Ograničenja: veličina programa ≤ 200, dužina izvršavanja ≤ 2 000.

Podzadatak 3 [19 bodova]

Postoje tačno dva kamenčića negdje u redu 0: jedan je u ćeliji (0, x), a drugi u ćeliji (0, y) gdje su x i y međusobno različiti i x + y je paran broj. Napišite program koji ostavlja odometar u ćeliji (0, (x + y) / 2), odnosno tačno na pola između dva polja koja inicijalno sadrže kamenčiće. Stanje polja po završetku nije bitno.

Ograničenja: veličina programa ≤ 100, dužina izvršavanja ≤ 200 000.

Podzadatak 4 [do 32 boda]

Postoji najviše 15 kamenčića u polju, nikoja dva u istoj ćeliji. Napišite program koji ih sve sakuplja u sjeverozapadni ugao. Preciznije, ako postoji x kamenčića u polju na početku, na kraju treba biti tačno x kamenčića u polju (0, 0) i niti jedan kamenčić u nekoj drugoj ćeliji.

Rezultat za ovaj podzadatak zavisi od dužine izvršavanja poslatog programa. Preciznije, ako je L maksimalna dužina izvršavanja između različitih testnih slučajeva, Vaš rezultat će biti:

- 32 boda 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.

Podzadatak 5 [do 28 bodova]

Može biti proizvoljan broj (naravno, između 0 i 15) kamenčića u svakoj ćeliji polja. Napišite program koji pronalazi minimum, odnosno završava sa odometrom na ćeliji (i, j) koja je takva da svaka druga ćelija u polju sadrži barem onoliko kamenčića koliko ih ima u ćeliji (i, j). Nakon izvršavanja programa, broj kamenčića u svakoj od ćelija treba biti isti kao i prije pokretanja programa.

Rezultat za ovaj podzadatak zavisi od veličine programa P poslatog programa. Preciznije, Vaš rezultat će biti:

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

Ograničenja: dužina izvršavanja $\leq 44\,400\,000$.

Implementacijski detalji

Trebate poslati tačno po jednu datoteku za svaki podzadatak, napisanu prema ranije navedenim sintaksnim pravilima. Svaki poslati fajl može biti najviše 5 MiB velik. Za svaki podzadatak, Vaš program za odometar bit će testiran na nekoliko testnih slučajeva i dobit ćete neke povratne informacije o resursima koje je iskoristio Vaš program. U slučaju kada poslati program nije sintaksno ispravan te ga zbog toga nije moguće testirati, dobit ćete informacije o specifičnoj sintaksnoj greški.

Nije neophodno da ono što pošaljete sadrži programe za sve podzadatke. Ako ne pošaljete program za podzadatak X, iskoristit će se ono što ste prethodno zadnje poslali za ovaj podzadatak; ako takav program ne postoji, ovaj podzadatak će biti ocijenjen sa 0 bodova.

Kao i obično, Vaš ukupni rezultat je suma bodova koje ste ostvarili po pojedinačnim podzadacima. Vaš finalni rezultat za ovaj zadatak je najbolji rezultat između svih slanja koja su testirana sa tokenom i Vašeg posljednjeg slanja.

Simulator

Za potrebe testiranja, bit će Vam dostupan simulator odometra kojem možete proslijediti svoje programe i ulazne podatke za polja. Programi za odometar trebaju biti napisani u istom formatu kao oni koji se šalju na ocjenjivanje, tj. u formatu koji je ranije opisan.

Opis polja bit će dat u sljedećem formatu: svaka linija datoteke treba da sadrži tri broja R, C i P sa značenjem da ćelija u redu R i koloni C sadrži P kamenčića. Za sve ćelije koje nisu navedene u opisu polja smatra se da ne sadrže kamenčiće. Na primjer, posmatrajmo sljedeću datoteku:

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

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

Simulator se može pozvati izvršavanjem programa `simulator.py` u folderu Vašeg zadatka uz proslijeđivanje argumenta sa nazivom datoteke koja sadrži program za odometar. Simulator prihvata sljedeće opcije na komandnoj liniji:

- `-h` će dati kratki pregled dostupnih opcija;
- `-g GRID_FILE` učitava opis polja iz datoteke `GRID_FILE` (ako se ne navede ova opcija, učitava se prazno polje);
- `-s GRID_SIDE` postavlja veličinu polja na `GRID_SIDE x GRID_SIDE` (ako se ne navede ova opcija, veličina polja je 256, kao što je opisano u postavci zadatka); upotreba manjih polja može biti korisna prilikom rješavanja problema u implementaciji;
- `-m STEPS` ograničava broj koraka izvršavanja u simulaciji na najviše `STEPS`;
- `-c` omogućava mod za kompajliranje; u modu za kompajliranje, simulator vraća potpuno isti izlaz ali umjesto simulacije upotrebom Python-a, generiše i kompajlira mali C program. Ovo uzrokuje nešto sporije pokretanje simulatora, ali daje rezultate značajno brže; preporučuje se da koristite ovaj mod kada očekujete da će Vaš program raditi duže od 10 000 000 koraka.

Broj slanja

Rješenja za ovaj zadatak smijete poslati najviše 128 puta.