Guličkový krokomer

Leonardo vynašiel prvý známy *krokomer*: vozidlo, ktoré meralo vzdialenosť pomocou zhadzovania guličiek počas otáčania jeho kolies. Po spočítaní guličiek sme dostali počet otočení kolesa a z toho sme už ľahko mohli dopočítať vzdialenosť. Obyčajné počítanie vzdialenosti je pre teplých, poriadni informatici do krokomeru pridali počítač, ktorý sa dá programovať a rozširuje jeho funkcionalitu. Vašou úlohou bude programovať tento vylepšený krokomer podľa nasledujúcich pravidiel.

Mriežka

Krokomer sa pohybuje po štvorcovej mriežke rozmerov 256 * 256 políčok. Každé políčko obsahuje najviac 15 guličiek a je identifikované párom súradníc (riadok, stĺpec), kde každé z čísel je z rozsahu 0, …, 255. S políčkom (i, j) susedia políčka (ak existujú) (i - 1, j), (i + 1, j), (i, j - 1) a (i, j + 1). Políčko, ktoré leží v prvom alebo poslednom riadku, alebo v prvom alebo poslednom stĺpci sa nazýva *okraj*. Krokomer vždy štartuje z políčka (0, 0) (severozápadný roh), otočený na sever.

Základné príkazy

Krokomer sa programuje pomocou nasledujúcich príkazov.

- `left` — otoč sa o 90 stupňov doľava (proti smeru ručičiek) a zostaň na súčasnom poličku (napr. ak sa krokomer pozerá na juh, po vykonaní príkazu sa bude pozerať na východ).
- `right` — otoč sa o 90 stupňov doprava (v smere ručičiek) a zostaň na súčasnom poličku (napr. ak sa krokomer pozerá na západ, po vykonaní príkazu sa bude pozerať na sever).
- `move` — pohni sa o krok dopredu na susedné políčko (v smere kam sa krokomer pozerá). Ak také políčko neexistuje, príkaz nemá žiadny efekt.
- `get` — zober jednu guličku zo súčasného políčka. Ak je súčasné políčko prázdne, príkaz nemá žiadny efekt.
- `put` — pridá jednu guličku na súčasné políčko. Ak súčasné políčko obsahuje 15 guličiek, príkaz nemá žiadny efekt. Krokomeru sa guličky nikdy neminú.
- `halt` — ukončí beh programu.

Krokomer vykonáva príkazy v poradí v akom sú v programe. Program musí obsahovať v každom riadku maximálne jeden príkaz. Prázdne riadky sú ignorované. Symbol # označuje komentár, všetko za týmto znakom až po koniec riadku bude ignorované. Pokiaľ krokomer dosiahne koniec programu, jeho beh sa ukončí.

Príklad 1

Tento program pre krokomer ho presunie na políčko (0, 2), kde sa bude pozerať na východ. (Všimnite si, že prvý príkaz `move` je ignorovaný, lebo krokomer je v severozápadnom rohu a pozerá sa na sever).

```
move  # žiadny efekt
right
# teraz sa krokomer pozerá na východ
move
move
```

Značky, okraje a guličky

Na riadenie behu programu v závislosti na aktuálnom stave okolitého prostredia, môžete použiť značky (case-sensitive reťazce z maximálne 128 znakov `a`, …, `z`, `A`, …, `Z`, `0`, …, `9`). Príkazy, ktoré používajú značky sú popísané nižšie. V nasledujúcom texte *`L`* označuje ľubovoľnú platnú značku.

- *`L`*`:` (napr. *`L`* nasledované dvojbodkou '`:`') — deklaruje miesto v programe so značkou *`L`*. Všetky deklarované značky musia byť odlišné. Deklarovanie značky nemá žiadny efekt na krokomer.
- `jump` *`L`* — pokračuje v behu programu skokom na značku *`L`*.
- `border` *`L`* — pokračuje v behu programu skokom na značku *`L`*, ak je krokomer na okraji mriežky a pozerá sa smerom von (ak by príkaz `move` nemal žiadny efekt). Inak program pokračuje ďalej a tento príkaz nemá žiadny efekt.
- `pebble` *`L`* — pokračuje v behu programu skokom na značku *`L`*, ak súčasné políčko obsahuje aspoň jednu guličku. Inak program pokračuje ďalej a tento príkaz nemá žiadny efekt.

Príklad 2

Nasledujúci program nájde najzápadnejšiu guličku v riadku 0 a zastaví sa na danom poličku. Ak v riadku nie sú žiadne guličky zastaví sa na okraji riadku. Používa dve značky `leonardo` a `davinci`.

```
right
leonardo:
pebble davinci  # gulička nájdená
border davinci  # koniec riadka
move
jump leonardo
davinci:
halt
```

Krokomer sa začne otáčať doprava. Cyklus začína deklaráciou značky `leonardo:` a končí príkazom `jump leonardo`. V cykle krokomer kontroluje prítomnosť guličky alebo koncu riadka a ak sa tak nestane, vykoná `move` zo súčasného políčka (0, j) na susedné políčko (0, j + 1), keďže určite existuje. (Príkaz `halt` nie je nutne potrebný, keďže program skončí tak, či onak.)

Úloha

Vašou úlohou je odovzdať program v jazyku krokomera (popísanom vyššie), ktorý sa správa ako je očakávané. Každá podúloha (pozri nižšie) špecifikuje správanie krokomera, ktoré treba splniť a ďalšie limity na riešenie. Limity sa týkajú nasledovných vecí.

- *Veľkosť programu* — program musí byť dostatočne krátky. Veľkosť programu je počet príkazov v ňom. Deklarácie značiek, komentáre a prázdne riadky sa nerátajú do veľkosti programu.
- *Dĺžka behu* — program musí skončiť dostatočne skoro. Dĺžka behu je počet krokov počas vykonávania programu. Každé jedno vykonanie každého príkazu sa počíta ako krok, nezávisle od toho, či má efekt alebo nie. Deklarácie značiek, komentáre a prázdne riadky sa nepočítajú ako kroky.

V príklade 1, veľkosť programu je 4 a dĺžka behu 4. V príklade 2, veľkosť programu je 6 a keď ho pustíme na mriežke s jednou guličkou na políčku (0, 10), dĺžka behu je 43 krokov: `right`, 10 iteráciu cyklu, každá iterácia trvá 4 kroky (`pebble davinci`; `border davinci`; `move`; `jump leonardo`), a nakoniec, `pebble davinci` a `halt`.

Podúloha 1 [9 bodov]

Na začiatku je x guličiek na políčku (0, 0) a y guličiek na políčku (0, 1), a všetky ostatné políčka sú prázdne. Pamätajte, že môže byť najviac 15 guličiek na jednom políčku. Napíšte program, ktorý skončí na políčku (0, 0) ak $x \leq y$ a na políčku (0, 1) inak. (Nezaujíma nás smer, ktorým sa krokomer na konci bude pozerať, takisto nás nezaujíma kde a koľko guličiek ostane nakonci v mriežke.)

Limity: veľkosť programu ≤ 100, dĺžka behu ≤ 1 000.

Podúloha 2 [12 bodov]

To isté, čo v predchádzajúcej úlohe, len navyše keď program skončí, políčko (0, 0) musí obsahovať x guličiek a políčko (0,1) musí obsahovať y guličiek.

Limity: veľkosť programu ≤ 200, dĺžka behu ≤ 2 000.

Podúloha 3 [19 bodov]

V mriežke sú presne dve guličky v riadku 0: jedna na políčku (0, x), druhá na políčku (0, y); x a y sú rôzne. Navyše x + y je párne číslo. Napíšte program, ktorý ponechá krokomer na políčku (0, (x + y) / 2), čiže presne v strede medzi políčkami obsahujúcimi guličky. Finálny stav mriežky nie je podstatný.

Limity: veľkosť programu ≤ 100, dĺžka behu ≤ 200 000.

Podúloha 4 [najviac 32 bodov]

Do mriežky bolo umiestnených najviac 15 guličiek a každá do iného políčka. Napíšte program, ktorý ich všetky pozbiera do severozápadného rohu, presnejšie ak bolo na začiatku x guličiek v mriežke, tak na konci musí byť x guličiek na políčku (0, 0) a žiadne guličky na iných políčkach.

Počet bodov za túto úlohy závisí od dĺžky behu vášho programu. Presnejšie ak L je maximálny počet krokov vášho programu, vaše skóre bude:

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

Limity: veľkosť programu ≤ 200.

Podúloha 5 [najviac 28 bodov]

V každom políčku môže byť ľubovoľne veľa guličiek (samozrejme medzi 0 a 15). Napíšte program, ktorý nájde minimum, čiže skončí na políčku (i,j) takom, že všetky ostatné políčka obsahujú aspoň toľko guličiek ako (i,j). Po skončení programu musí byť počet guličiek v každom políčku rovnaký ako na začiatku.

Počet bodov za túto úlohy závisí od veľkosti P vášho programu. Presnejšie vaše skóre bude:

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

Limity: dĺžka behu ≤ 44 400 000.

Detaily implementácie

Odovzdajte práve jeden súbor pre každú podúlohu, napísaný podľa pravidiel uvedených vyššie. Každý odovzdaný súbor môže mať maximálne veľkosť 5 MiB. Pre každú podúlohu, váš program pre krokomer bude otestovaný na niekoľkých testcasoch a dostanete feedback o prostriedkoch použitých vaším kódom. V prípade, že váš kód je syntaktický nesprávny a v dôsledku toho je ho nemožné otestovať, dostane informáciu o syntaktickej chybe.

Nie je nutné aby ste zakaždým submitovali program pre každú podúlohu. Pokiaľ váš submit neobsahuje program pre podúlohu X, automaticky sa pre ňu použije najnovší z vašich predchádzajúcich submitov pre podúlohu X. (Pokiaľ taký program neexistuje, za danú podúlohu dostanete 0 bodov.)

Ako obvykle, počet bodov za submit je súčtom počtu bodov získaných za každú podúlohu a finálny počet bodov za úlohu je maximum bodov z release-tested submitov a posledného submitu.

Simulátor

Pre účely testovania dostanete simulátor krokomera, ktorý môžete nakŕmiť vaším programom a začiatočným stavom mriežky. Programy pre krokomer sa píšu v rovnakom formáte ako tie, ktoré submitujete.

Popisy mriežky sa zadávajú v nasledovnom formáte: každý riadok súboru musí obsahovať tri čísla: R, C a P, ktoré znamenajú, že políčko v riadku R a stĺpci C obsahuje P guličiek. O políčkach, ktoré nie sú popísané sa predpokladá, že sú prázdne. Napríklad, zoberme si súbor:

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

Mriežka popísaná týmto súborom obsahuje 15 guličiek: 3 na políčku (0, 10) a 12 na políčku (4, 5).

Simulátor môžete spustiť príkazom `simulator.py` v adresári s úlohou, pričom meno programu mu predáte ako argument. Simulátor akceptuje nasledovné možnosti z príkazového riadku:

- `-h` dá stručný popis dostupných možností;
- `-g GRID_FILE` načíta súbor s mriežkou zo súboru `GRID_FILE` (defaultne sa používa prázdna mriežka);
- `-s GRID_SIDE` nastaví veľkost mriežky na `GRID_SIDE x GRID_SIDE` (default: 256, presne ako v zadaní úlohy); použitie menšej mriežky môže byť užitočné pri debuggovaní;
- `-m STEPS` obmedzí počet krokov simulácie na najviac `STEPS`;
- `-c` spustí kompilačný mód, simulátor vráti presne rovnaký výstup, ale miesto toho, aby robil simuláciu v Pythone, zgeneruje a skompiluje malý Cčkový program. Toto síce stojí veľa roboty pred spustením simulácie, ale samotná simulácia je o poznanie rýchlejšia. Odporúčané použiť, ak očakávaný počet krokov presiahne 10 000 000;

Počet submitov

Maximálny počet submitov v tejto úlohe je obmedzený na 128. The maximum number of submissions allowed for this task is 128.