Kamínkový odometr

Leonardo vynalezl nejstarší měřič vzdáleností: vozík, ze kterého při každém otočení kola vypadl kamínek. Počet vypadlých kamínků udával počet otočení kola, z něhož bylo možné určit ujetou vzdálenost. Do tohoto zařízení jsme přidali sofistikovaný software rozšiřující jeho schopnosti. Vaším úkolem je psát programy pro takto vylepšený odometr v níže popsaném programovacím jazyce.

Pracovní pole

Odometr se pohybuje po čtvercové mřížce o rozměrech 256 × 256. Na každém políčku této mřížky může být nejvýše 15 kamínků. Políčka jsou určena souřadicemi ve tvaru (řádek, sloupec), kde čísla řádků i sloupců jsou v rozsahu od 0 do 255 včetně. S políčkem (i, j) tedy sousedí políčka (i - 1, j), (i + 1, j), (i, j - 1) a (i, j + 1), jestliže existují (obě jejich souřadnice jsou v povoleném rozsahu). Políčka v prvním a posledním řádku a v prvním a poslením sloupci nazýváme hraniční. Odometr vždy začíná na políčku (0, 0), což je severozápadní roh mřížky, a směřuje na sever.

Základní příkazy

Odometr lze řídit pomocí následujících příkazů:

- `left` — odometr se otočí o 90 stupňů doleva (proti směru hodinových ručiček, například směřoval-li na jih, bude směřovat na východ) a zůstane přitom na původním políčku.
- `right` — odometr se otočí o 90 stupňů doprava (po směru hodinových ručiček, například směřoval-li na západ, bude směřovat na sever) a zůstane přitom na původním políčku.
- `move` — odometr se posune o jedno políčko ve směru, kterým je natočen. Jestliže toto políčko neexistuje (odometr narazil na hranici), tento příkaz nic nedělá.
- `get` — odstraní jeden kamínek z políčka, na kterém odometr stojí. Není-li tam žádný kamínek, tento příkaz nic nedělá.
- `put` — přidá jeden kamínek na políčko, na kterém odometr stojí. Je-li tam už 15 kamínků, tento příkaz nic nedělá. Odometr má neomezenou zásobu kamínků.
- `halt` — ukončí běh programu.

Odometr vykonává příkazy v pořadí, v němž jsou zadány v programu. Každý řádek programu může obsahovat pouze jeden příkaz. Prázdné řádky se ignorují. Symbol # označuje komentář a jakýkoliv text za ním až do konce řádky je ignorován. Po dosažení konce programu výpočet končí.

Příklad 1

Následující program otočí odometr směrem na východ a posune ho na políčko (0, 2). První příkaz `move` je ignorován, jelikož odometr je na severní hranici a směřuje na sever.

```
move  # nic nedělá
right
# nyní odometr směřuje na východ
move
move
```

Návěští, hranice a kamínky

Běh programu lze ovlivnit v závislosti na stavu odometru. Návěští je řetězec nanejvýš 128 znaků `a`, …, `z`, `A`, …, `Z`, `0`, …, `9`. Rozlišují se v něm velká a malá písmena. V popisu následujících přikazů *`L`* označuje jakékoliv návěští.

- *`L`*`:` (tedy návěští *`L`* následované dvojtečkou '`:`') — deklaruje návěští *`L`* a označí jím přislušnou pozici v programu. Všechna deklarovaná návěští musí být navzájem různá. Deklarace návěští nijak neovlivňuje stav odometru.
- `jump` *`L`* — výpočet programu pokračuje od řádku, na němž je deklarováno návěští *`L`*.
- `border` *`L`* — jestliže je odometr na hraničním políčku a směřuje ven (tedy příkaz `move` by nic nedělal), pak výpočet programu pokračuje od řádku, na němž je deklarováno návěští *`L`*. Jinak výpočet pokračuje následujícím přikazem jako normálně, t.j. příkaz `border` v takovém případě nic nedělá.
- `pebble` *`L`* — jestliže je na aktuálním políčku alespoň jeden kamínek, pak výpočet programu pokračuje od řádku, na němž je deklarováno návěští *`L`*. Jinak výpočet pokračuje následujícím přikazem jako normálně, t.j. příkaz `pebble` v takovém případě nic nedělá.

Příklad 2

Následující program nalezne první (nejzápadnější) kamínek v řádku 0 a zastaví na příslušném políčku odometr. Nejsou-li v řádku 0 žádné kamínky, odometr se zastaví na východní hranici. Program používá dvě návěští `leonardo` a `davinci`.

```
right
leonardo:
pebble davinci  # nalezen kamínek
border davinci  # konec řádku
move
jump leonardo
davinci:
halt
```

Odometr se nejprve otočí doprava. Poté vstoupí do smyčky začínající návěštím `leonardo:` a končící příkazem `jump leonardo`. Ve smyčce odometr kontroluje, zda narazil na kamínek či hranici, a v tom případě skončí. Jinak se vykonáním příkazu `move` posune z aktuálního políčka (0, j) na sousední políčko (0, j + 1). Příkaz `halt` na konci není vlastně potřeba, jelikož by se i bez něj program ukončil.

Zadání

Odevzdejte programy v jazyce odometru, řešící níže popsané úlohy. V každé podúloze je přesně popsáno, jak se má odometr chovat a jaká omezení musí váš program splňovat. Tato omezení se týkají následujících vlastností programu.

- *Délka programu* — program by měl být dostatečně krátký. Délka programu je počet jeho příkazů. Deklarace návěští, prázdné řádky a komentáře se nepočítají.
- *Doba běhu* — běh programu musí skončit dostatečně rychle. Doba běhu je počet vykonaných příkazů. Počitají se i přikazy, které nic nedělaly. Deklarace návěští, prázdné řádky a komentáře se nepočítají.

V příkladu 1 je délka programu 4 a doba běhu 4. V příkladu 2 je délka programu 6. Je-li tento program spuštěn na pracovním poli s jediným kamínkem na políčku (0, 10), pak doba jeho běhu je 43: `right`, 10 iterací smyčky, v každé iteraci se vykonají 4 příkazy (`pebble davinci`; `border davinci`; `move`; `jump leonardo`), a na závěr `pebble davinci` a `halt`.

Podúloha 1 [9 bodů]

Na políčku (0, 0) je x kamínků, na políčku (0,1) je y kamínků a ostatní políčka jsou prázdná. Připomeňme, že na každém políčku může být nejvýše 15 kamínků. Napište program, po jehož ukončení bude odometr na políčku (0, 0) jestliže $x \leq y$, a na políčku (0, 1) jinak. (Nezáleží nám na směru, kterým bude odometr na konci směřovat. Také nám nezáleží na počtu a pozicích kamínků na konci výpočtu.)

Omezení: délka progamu ≤ 100, doba běhu ≤ 1 000.

Podúloha 2 [12 bodů]

Stejná úloha, ale po skončení běhu programu musí být na políčku (0, 0) přesně x kamínků a na políčku (0, 1) přesně y kamínků.

Omezení: délka progamu ≤ 200, doba běhu ≤ 2 000.

Podúloha 3 [19 bodů]

Někde na řádku 0 jsou právě dva kamínky: jeden je na políčku (0, x), druhý na políčku (0, y), kde x a y jsou různá čísla a jejich součet je sudý. Napište program, po jehož doběhnutí je odometr na políčku (0, (x + y) / 2), tedy přesně uprostřed mezi políčky, která na začátku obsahují kamínky. Na koncovém stavu pracovního pole nezáleží.

Omezení: délka progamu ≤ 100, doba běhu ≤ 200 000.

Podúloha 4 [až 32 bodů]

Na pracovním poli je nejvýše 15 kamínků, ležících na navzájem různých políčkách. Napište program, který všechny kamínky sesbírá a přenese je na políčko (0, 0). Je-li tedy na začátku na pracovním poli x kamínků, potom na konci je x kamínků na políčku (0, 0) a ostatní políčka jsou prázdná.

Počet bodů za tuto podúlohu závisí na době běhu vašeho programu. Je-li L maximum z dob běhů programu na testovacích vstupech, získáte následující počet bodů:

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

Omezení: délka programu ≤ 200.

Podúloha 5 [až 28 bodů]

Na každém políčku může být libovolný počet kamínků (mezi 0 a 15 včetně). Napište program, který nalezne minimum. Po skončení jeho běhu odometr bude na políčku (i, j) takovém, že každé jiné políčko obsahuje alespoň tolik kamínků jako (i, j). Počet kamínků na každém políčku musí na konci být stejný jako na začátku.

Počet bodů za tuto podúlohu závisí na délce P vašeho programu. Přesněji, váš program získá následující počet bodů:

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

Omezení: doba běhu ≤ 44 400 000.

Implementace

Pro každou podúlohu odevzdejte právě jeden program zapsaný dle výše popsaných syntaktických pravidel. Každý odevzdaný program může mít nejvýše 5 MiB. Váš program bude spuštěn na několika testovacích datech a dozvíte se informaci o době jeho běhu a jeho délce. Není-li odevzdaný program syntakticky správně, nemůže být otestován. V tomto případě dostanete odpovídající chybovou hlášku popisující konkrétní syntaktickou chybu.

Nemusíte naráz zasílat programy pro všechny podúlohy. Jestliže vámi zaslaný archív neobsahuje program pro podúlohu X, doplní se místo něj vaše poslední odevzdané řešení podúlohy X. Jestliže jste podúlohu X ještě neodevzdali, bude hodnocena 0 body.

Jako obvykle, počet bodů za celý submit je dán součtem bodů za podúlohy, a konečný počet bodů za úlohu je maximum z release-tested submitů a z posledního submitu.

Simulátor

Pro testování vašich programů máte k dispozici simulátor odometru.

Pracovní pole pro simulátor zadávejte v následujícím formátu. Každá řádka souboru obsahuje tři čísla R, C a P, která určují, že na políčku (R, C) leží P kamínků. Políčka, která v souboru nejsou zmíněna, neobsahují žádné kamínky. Například:

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

Pracovní pole popsané tímto souborem obsahuje 15 kamínků: 3 na políčku (0, 10) a 12 na políčku (4, 5).

Simulátor spustíte zavoláním programu `simulator.py` v adresáři úlohy. Jméno souboru s programem zadejte jako jeho argument. Simulátor má následující volby:

- `-h` stručný popis voleb.
- `-g GRID_FILE` nahraje popis pracovního pole ze souboru `GRID_FILE`. Bez této volby je pracovní pole prázdné.
- `-s GRID_SIDE` nastaví rozměry pracovního pole na `GRID_SIDE x GRID_SIDE`. Bez této volby jsou rozměry 256 x 256 jako v zadání. Menší pracovní pole mohou být užitečná při ladění vašich programů.
- `-m STEPS` omezí dobu běhu na nanejvýš `STEPS`;
- `-c` spustí simulátor v módu kompilace. V tomto módu simulátor vrací stejný výstup jako normálně, ale místo krokování vašeho programu ho nejprve zkompiluje do programu v C, který pak spustí. Spuštění simulátoru v módu kompilace je pomalejší, ale samotný program pak běží podstatně rychleji. Tento mód doporučujeme použít, pokud doba běhu vašeho programu je alespoň 10 000 000.

Počet submitů

Řešení této úlohy můžete odevzdat nejvýše 128x.