Одометар со камчиња

Леонардо го измислил оригиналниот *одометар*: количка која може да измери растојанија со испуштање на камчиња дури се вртат тркалата на количката. Со броење на камчињата се утврдувал бројот на вртежи на тркалото, па можело да се утврди изоденото растојание со одометарот. Како компјутерции, ние додадовме софтверска контрола на одометарот, со проширување на неговите функционалности. Ваша задача е да го програмирате одометарот според правилата дадени подолу.

Шема (мрежа) на оперирање

Одометарот се движи на замислена квадратна шема од 256 × 256 единечни клетки. Секоја клетка може да содржи најмногу 15 камчиња и се идентификува со пар координати (редица, колона), каде секоја координата е во рангот 0, …, 255. За дадена клетка (i, j), соседни клетки до неа се (ако постојат) (i - 1, j), (i + 1, j), (i, j - 1) и (i, j + 1). Секоја клетка која се наоѓа на првата или последната редица, или на првата или последната колона, се нарекува *граница*. Одометарот секогаш започнува на клетката (0, 0) (аголот на северо-запад), свртен накај север.

Основни команди

Одометарот може да се програмира до користење на следните команди:

- `left` — сврти 90 степени во лево (спротивно од движење на стрелките на часовникот) и остани во актуелната клетка (на пример, ако бил свртен кон југ, ќе биде свртен кон исток после командата).
- `right` — сврти 90 степени во десно (во правец на движење на стрелките на часовникот) и остани во актуелната клетка (на пример, ако бил свртен кон запад, ќе биде свртен кон север после командата).
- `move` — придвижи се еден чекор (една клетка) нанапред (во насока во која е свртен одометарот) во соседната клетка. Ако таква клетка не постои (т.е. веќе е достигната границата во таа насока) тогаш оваа команда нема ефект.
- `get` — отстрани едно камче од актуелната клетка. Ако актуелната клетка нема камчиња, тогаш командата нема ефект.
- `put` — додај едно камче во актуелната клетка. Ако конкретната клетка веќе содржи 15 камчиња, тогаш командата нема ефект. Одометарот секогаш има доволно камчиња за да може да остави едно.

- `halt` — прекини го извршувањето.

Одометатор ги извршува командите во редослед како што се дадени во програмата. Програмата смее да има најмногу една команда во една линија. Празните линии се игнорираат. Симболот # индицира коментар; се игнорира текстот кој што следи по него, се до крајот на линијата. Ако одометарот дојде до крајот на програмата, извршувањето се прекинува.

Пример 1

Разгледајте ја следната програма за одометарот. Таа го донесува одометарот до клетката (0, 2), свртен на исток. (Забележете дека првиот `move` се игнорира, бидејќи одометарот е на северо-западниот ќош и е свртен накај север.)

```
move  # нема ефект
right
# сега одометарот е свртен кон исток
move
move
```

Етикети (лабели), граници и камчиња

За да го измените текот на програмата, во зависност од актуелниот статус, можете да користите лабели кои се стрингови (кај кои е значајно дали буквите се големи или мали, case-sensitive) и кои се состојат од најмногу 128 симболи кои се некои од следниве `a`, …, `z`, `A`, …, `Z`, `0`, …, `9`. Новите команди кои се однесуваат на лабелите се дадени подолу. Во описот подолу, *`L`* ја означува која било валидна лабела.

- *`L`*`:` (т.е. *`L`* проследено со знакот две-точки ‘`:`’) — ја декларира локацијата во програмата со лабела *`L`*. Сите декларирани лабели мора да се единствени. Декларирањето лабела нема ефект на одометарот.

- `jump` *`L`* — продолжи со извршувањето со безусловен скок на линијата со лабела *`L`*.

- `border` *`L`* — продолжи го извршувањето со скок на линијата со лабела *`L`*, ако одометарот е на граница, свртен кон работ на шемата (т.е. кога инструкцијата `move` нема да има ефект); во спротивно, извршувањето продолжува нормално и оваа команда нема ефект.

- `pebble` *`L`* — продолжи го извршувањето со скок на линијата со лабела *`L`*, ако актуелната клетка има барем едно камче; во спротивно, извршувањето продолжува нормално и оваа команда нема ефект.

Пример 2

Следната програма го лоцира првото (најзападното) камче во редицата 0 и застанува таму; ако нема камчиња во редицата 0, застанува на границата на крајот на редицата. Користи две лабели `leonardo` и `davinci`.

```
right
leonardo:
pebble davinci  # најдено е камче
border davinci  # крај на редицата
move
jump leonardo
davinci:
halt
```

Одометарот започнува со вртење во десно. Циклусот започнува со декларацијата на лабелата `leonardo:` и завршува со командата `jump leonardo`. Во циклусот, одометарот го проверува присуството на камче или на граница на крајот од редицата. Во случај да не е најдено одометарот прави `move` од актуелната клетка (0, j) кон соседната клетка (0, j + 1) бидејќи истата постои. (Командата `halt` не е стриктно неопходна бидејќи програмата секако завршува.)

Задача

Вие треба да предадете програма во јазикот на одометарот, според описот погоре, таква што одометарот ќе го натера да се однесува според очекувањето. Секоја подзадача (видете подолу) специфицира однесување кое одометарот треба да го исполни и ограничувањата кои мора да се задоволат од предаденото решение. Ограничувањата се однесуваат на следните две работи:

- *Големина на програмата* — програмата мора да е доволно кратка. Големина на програмата е бројот на команди во неа. Декларациите на лабели, коментарите и празните линии *не се пресметуваат* во големината.

- *Должина на извршување* — програмата мора доволно брзо да заврши. Должина на извршување е бројот на *чекори*: како чекор се брои секое извршување на команда, без разлика дали командата имала ефект или не; Декларациите на лабели, коментарите и празните линии не се сметаат за чекори.

Во пример 1, големината на програмата е 4, а должината на извршување е 4. Во пример 2, големината на програмата е 6 и, при извршување на шема со едно единствено камче на клетката (0, 10), должината на извршување е 43 чекори: `right`, 10 повторувања на циклусот, секоја итерација зазема 4 чекора (`pebble davinci`; `border davinci`; `move`; `jump leonardo`), и конечно, `pebble davinci` и `halt`.

Подзадача 1 [9 поени]

На почетокот има x камчиња во клетката (0, 0) и y во клетката (0, 1), додека сите останати клетки се празни. Да се потсетиме дека има најмногу 15 камчиња во секоја клетка. Напиши програма која завршува со одометарот во клетката (0, 0) ако $x \leq y$, и во клетката (0, 1) во спротивно. (Не ни е гајле за насоката на одометарот кон која е свртен на крајот; не ни е гајле ни колку камчиња на крајот има на шемата и каде се истите поставени.)

Ограничувања: големина на програмата ≤ 100, должина на програмата $\leq 1\,000$.

Подзадача 2 [12 поени]

Истата задача како погоре, со тоа што, кога програмата ќе заврши, клетката (0, 0) мора да содржи точно x камчиња, а клетката (0, 1) мора да содржи точно y камчиња.

Ограничувања: големина на програмата ≤ 200, должина на програмата ≤ 2 000.

Подзадача 3 [19 поени]

Има тично две камчиња некаде во редицата 0: едно е во клетката (0, x), другото во (0, y); x и y се различни, и x + y е парен број. Напишете програма која го остава одометарот во клетката (0, (x + y) / 2), т.е., точно на половина помеѓу двете клетки кои содржат камчиња. Финалната состојба на шемата не е значајна.

Ограничувања: големина на програмата ≤ 100, должина на програмата ≤ 200 000.

Подзадача 4 [до 32 поена]

Има најмногу 15 камчиња во шемата, ниедни две не се во иста клетка. Напишете програма која ги собира сите нив во северо-западниот агол; поточно, ако има x камчиња во шемата на почетокот, на крајот мора да има точно x камчиња во клетката (0, 0), и никакви камчиња на друго место.

Резултатот за оваа подзадача зависи од должината на извршување на предадената програма. Поточно, ако L е максималното време на извршување на различните тест случаи, вешиот резултат ќе биде:

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

Ограничувања: големина на програмата ≤ 200.

Подзадача 5 [до 28 поена]

Може да има колку било камчиња во секоја клетка на шемата (се разбира, од 0 до 15). Напишете програма што го наоѓа минимумот, т.е., што завршува со одометарот во клетката (i, j) која е таква што секоја друга клетка содржи толку камчиња колку што има (i, j). По извршувањето на програмата, бројот на камчиња во секоја клетка мора да е ист како пред извршувањето на програмата.

Поените на оваа подзадача зависат од големината P на предадената програма. Поточно, освоените поени ќе бидат:

- 28 поени ако P ≤ 444;

- 28 - 28 $\log_{10}$ (P / 444) поени ако 444 < P < 4 440;
- 0 поени ако P ≥ 4 440.

Ограничувања: должина на извршување ≤ 44 400 000.

Имплементациски детали

Мора да предадете точно една датотека по подзадача, напишана според синтаксните правила дадени погоре. Секоја предадена датотека може да има максимална големина од 5MB. За секоја подзадача, вашиот одометар ќе биде тестиран на неколку тест случаи и ќе добиете некаков фидбек за ресурсите искористени од вашиот код. Во случај кодот да не е синтаксно точен па и невозможо да се тестира, ќе добиете информација за специфичната синтаксна грешка.

Не е неопходно вашата субмисија да содржи програми за одометрите од сите подзадачи. Ако вашата актуелна задача не содржи програма за одометарот од подзадачата X, вашата најпоследна субмисија за подзадачата X се вклучува автоматски; ако таква програма не постои, за подзадачата ќе добиете 0 за таа субмисија.

Вообичаено, поените на една субмисија претставуваат сума од поените добиени на секоја подзадача, и финалните поени добиени за задачата е максимумот од тестираните (со рилис) субмисии и последната субмисија.

Симулатор

За тестирање, добивате симулатор на одометар, кој може да прима ваши програми и влезни шеми. Одометарските програми ќе бидат напишани во истиот формат како оној кој се користи за субмисиите (според објаснувањето погоре)

Описот на шемата ќе биде даден според следниот формат: Секоја линија на датотеката мора да содржи 3 броја,, R, C и P, со значење дека клетката во редица R и колона C содржи P камчиња. Сите клетки кои нема да се специфицирани во описот на шемата ќе се смета дека немаат камчиња. На пример, разгледајте ја датотеката:

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

Шемата опишана со ова ќе содржи 15 камчиња: 3 во клетката (0, 10) и 12 во клетката (4, 5).

Може да го повикате тест симулаторот со повик на програмата `simulator.py` во вашиот директориум за задачата, предавајќи го името на датотеката со програмата како аргумент. Симулаторот ќе ги прифати следните опции на командната линија:

- `-h` ќе дава преглед на расположивите опции;
- `-g GRID_FILE` вчитува опис на шема од датотеката `GRID_FILE` (default: празна шема);

- `-s GRID_SIDE` ја поставува големината на шемата на `GRID_SIDE x GRID_SIDE` (default: 256, како што е дадено во спецификацијата на проблемот); користењето на помали шеми може да е корисно за дебагирање на програмите;

- `-m STEPS` го ограничува бројот на чекорите на извршување до најмногу `STEPS`;

- `-c` влегува во мод на компајлирање; во мод за компајлирање, симулаторот го враќа истиот излез, но наместо да ја работи симулацијата со Python, генерира и компајлира мала C програма. Ова предизвикува поголем застој при стартување, но дава резултати значително побрзо; ве советуваме да ја користите кога очекувате дека вашата програма ќе работи подолго од 10 000 000 чекори.

Број на субмисии

Максималниот број на субмисии дозволени за оваа задача е 128.