# Одометър с камъчета

Леонардо изобретил оригинален одометър: количка, с която може да се измерват разстояния чрез оставяне на камъчета, когато колелата на количката се завъртат. Преброяването на камъчетата дава броя на завоите, направени от количката. Това позволява да пресметнем изминатото разстояние от одометъра. Вие трябва да направите софтуер, който да подобри функционалността на одометъра. Вашата задача е да програмирате одометъра пи спазване на правилата, дадени по-долу.

**Мрежа**

Одометърът се движи по въображаема квадратна мрежа от 256 × 256 единични клетки. Всяка клетка може да съдържа най-много 15 камъчета и се идентифицира с двойка координати (ред, стълб), където всяка координата е диапазона 0, …, 255. За дадена клетка (i, j), съседите й са (ако съществуват): (i - 1, j), (i + 1, j), (i, j - 1) и (i, j + 1). Всяка клетка, намираща се на първия или на последния рес, или на първия или на последния стълб, се нарича гранична. Одометърът винаги започва от клетката (0, 0) (североизточния ъгъл) и е насочен на север.

**Основни команди**

Одометърът може да бъде програмиран чрез следните команди:

- `left` — завърта го на 90 градуса на лявот (по посока на часовниковата стрелка) и остава в текущата клетка (т.е. ако е сочил на юг, сега ще сочи на изток).
- `right` — завърта го на 90 градуса на дясно (по посока на часовниковата стрелка) и остава в текущата клетка (т.е. ако е сочил на запад, сега ще сочи на север).
- `move` — премества го на единица разстояние напред (в посоката, която сочи) в съседна клетка. Ако няма такава клетка, командата няма ефект.
- `get` — премахва по едно камъче от текущата клетка. Ако в текущата клетка няма камъчета, командата няма ефект..
- `put` — добавя по едно камъче в текущата клетка. Ако текущата клетка съдържа 15 камъчета, командата няма ефект.
- `halt` — завършва изпълнението.

Одометърът изпълнява командите в реда, даден в програмата - една команда на ред. Програмата трябва да съдържа по една команда (най-много) на ред. Празните редове се

игнорират. Знакът # показва началото на коментар, който завършва на края на реда. Ако одометърът достигне края на програмата, изпълнението завършва.

**Пример 1**

Разглеждаме следната програма за одометър. Тя придвижва одометъра в клетка (0, 2) и го оставя насочен на изток. (Отбележете, че първото `move` се игнорира, защото одометърът е на северозападния ъгъл и сочи на север.)

```
move  # няма ефект
right
# сега одометърът е насочен на изток
move
move
```

**Етикети, граници и камъчета**

За да се промени протичането на програмата в зависимост от текущия статус, може да се използват етикети, които са низосе от най-много 128 знака измежду `a`, …, `z`, `A`, …, `Z`, `0`, …, `9` (малките и главните букви се различават). Командите с етикети са показани, като *`L`* означава валиден етикет.

- *`L`*`:` (т.е. *`L`*, следван от двоеточие ‘`:`’) — деклара място в програмата за етикет *`L`*. Етикетите трябва да са уникални. Деклариране на етикет няма ефект за одометъра.
- `jump` *`L`* — продължава изпълнението с безусловен скок на реда с етикет *`L`*.
- `border` *`L`* — продължава изпълнението със скок на реда с етикет *`L`*, ако одометърът е на границата (i.e. командата `move` няма ефект); в противен случай изпълнението продължава нормално и тази команда няма ефект.
- `pebble` *`L`* — продължава изпълнението със скок на реда с етикет *`L`*, ако текущата клетка съдържа поне поне едно камъче; в противен случай изпълнението продължава нормално и командата няма ефект.

**Пример 2**

Следващата програма намира първото (най-западно) камъче в ред 0 и спира там; ако няма камъчета в ред 0, спира на границата на края на реда. Използва два етикета `leonardo` и `davinci`.

```
right
leonardo:
pebble davinci  # камъчето е намерено
border davinci  # край на реда
move
jump leonardo
davinci:
halt
```

Одометърът стартира чреа завъртане на дясно. Цикълът започва с етикет `leonardo:` и завършва с команда `jump leonardo`. В цикъла одометърът проверява наличие на

камъче или граница в края на реда. Ако не е така, одометърът прави `move` от текущата позиция (0, j) към съседната клетка (0, j + 1), понеже тя съществува. (командата `halt` не е строго необходима тук, понеже програмата завършва.)

# Задача

Вие трябва да изпратите програма на езика на одометъра, която прави движението на одометъра, както се очаква. Всяка подзадача определя поведение на одометъра, такова че да изпълни ограниченията. Ограниченията са от два вида:

- *Program size* — програмата трябва да е възможно най-къса. Размер на прогромата се нарича броят на командите. Декларациите на етикети и коментариите не се броят в размера.
- *Execution length* — Програмата трябва да завърши възможно най-бързо. Дължината на изпълнението е броят на изпълнените стъпки: изпълненото на всяка единична команда се брои като стъпка, без значение дали командата има ефект, или - не; Декларациите на етикети, коментариите и празните редове не се броят за стъпки.

В пример 1, размерът на програмата е 4 и размерът на изпълнението е 4. В пример 2, размерът на програмата е 6 и когато се изпълнява в мрежа с едно камъче в клетка (0, 10), размерът на изпълненото е 43 стъпки: `right`, 10 итерации на цикъла; всяка итерация има 4 4 стъпки (`pebble davinci`; `border davinci`; `move`; `jump leonardo`), и накрая `pebble davinci` и `halt`.

# Subtask 1 [9 points]

В началото има x камъчета в клетка (0, 0) и y в клетка (0, 1), като другите клетки са празни. Да отбележим, че може да има най-много 15 камъчета в една клетка. Напишете програма, която завършва с одометъра в клетка (0, 0), ако $x \leq y$, и в клетка (0, 1) е противен случай. (Не се интересуваме от посоката на одометъра; също не се интересуваме колко камъчета има на края в мрежата, или къде са те.)

*Limits:* размер на програмата ≤ 100, размер на изпълнението ≤ 1 000.

# Subtask 2 [12 points]

Същата задача, като по-горе, но при завършване на програмата, клетката (0, 0) трябва да съдържа точно x камъчета и клетката (0, 1) трябва да съдържа точно y камъчета.

*Limits:* размер на програмата ≤ 200, размер на изпълнението ≤ 2 000.

# Subtask 3 [19 points]

Има точно 2 камъчета в ред 0: едно камъче в клетка (0, x), едно камъче - в клетка (0, y); x и y са различни, и x + y е четно. Напишете програма, която оставя одометъра в клетка (0, (x + y) / 2), т.е. точно в средата между двете клетки, съдържащи камъчета. Крайното

състеояние на мрежата не е от значение.

*Limits:* размер на програмата ≤ 100, размер на изпълнението ≤ 200 000.

# Subtask 4 [up to 32 points]

Има най-много 15 камъчета в мрежата, никои две не са в една и съща клетка. Напишете програма, която ги събира в северозападния ъгъл. По-точно: ако е имало x камъчета в началото, в края трябва да има x камъчета в клетка (0, 0) и да няма камъчета в други клетки.

Точките за тази подзадача зависят от дължината за изпълнение на изпратената програма. По-точно, ако L е максимумът от дължините за изпълнение, вашият резултат ще е:

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

*Limits:* размер на програмата ≤ 200.

# Subtask 5 [up to 28 points]

Може да има различен брой камъчета във всяка клетка на мрежата (разбира се, между 0 и 15). Напишете програма, която намира минимумът, т.е. която завършва с одометъра в клетка (i, j) така, че всяка друга клетка да съдържа поне толкова камъчета, както (i, j). След завършване на програмата, броят на камъчетата във всяка клетка трябва да е същият от преди стартирането на програмата.

Точките за тази подзадача зависят от размера на програмата P. По-точно, вашият резултат е:

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

*Ограничения:* дължина на изпълнението ≤ 44 400 000.

# Детайли на имплементацията

Трябва да събмитнете точно по един файл за подзадача. Всеки събмитнат файл трябва да има размер най-много 5 MiB. За всяка подзадача, вашата програма за одометъра ще бъде тествана с няколко теста и вие ще получите някакъв feedback. Ако кодът не е синктактически коректен, ще получите информация.

Не е необходимо вашите събмити да съдържат програма на одометъра за всички подзадачи. Ако текущият събмит не съдържа програма на одометъра за подзадача X, вашият последен събмит за подзадача X автоматично се включва; ако не съществува такава програма, подзадачата ще има нула точки за този събмит.

Както обичкновенно, точкит при събмит е сумата от точките получени при всяка подзадача; крайните точки за задачата са максимума от точките на събмитите.

**Симулатор**

С цел тестване, вие ще разполагате със симулатор на одометър, който може да захранвате с вашите програми и мрежи.

Описание на мрежата ще бъде дадено в следния формат: всеки ред от входа съдържа три числа: R, C и P, означаващи, че клетката от ред R и стълб C съдържа P камъчета. Всички клетки, които не са описани, съдържат по 0 камъчета. Например:

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

Решетката, описана в този файл съдържа 15 камъчета: 3 в клетката (0, 10) и 12 в клетката (4, 5).

Може да извикате симулатора за тестване - `simulator.py` в папката на задачата, задавайки му като аргумент името на програмата. Симулаторът приема следните команди и опции:

- `-h` прави кратък преглед на допустимите опции;
- `-g GRID_FILE` зарежда описанието на решетката от файла `GRID_FILE` (по премълчаване: празна решетка);
- `-s GRID_SIDE` задава размера на решетката `GRID_SIDE x GRID_SIDE` (по премълчаване: 256, както е в условието на задачата); използването на по-малка решетка може да бъде полезно при тестване на програмата;
- `-m STEPS` ограничава броя на изпълняваните симулационни стъпкидо най-много `STEPS`;
- `-c` влиза в режим на компилация; в този режим, симулаторът връща същия изход, но вместо да извършва симулацията с Python, генерира и компилира малки програми на C. Това причинява голямо натоварване по време на стартирането, но след това дава значително по-бърз резултат; съветваме Ви да използвате този режим, когато очаквате програмата Ви се очаква да извърши повече от 10 000 000 стъпки.

**Брой на събмитите**

Максималният брой позволени събмити за тази задача са 128.