Одометр с камнями

Леонардо изобрел оригинальный одометр, представляющий собой тележку, которая может измерять расстояния путем бросания камней в тех местах, где она повернулась. Посчитав количество камней, мы получим количество поворотов колес тележки, которое позволит пользователю вычислить расстояние, которое проехал одометр. Как настоящие программисты, мы добавили программный контроль к одометру, расширив его функциональность. Требуется написать программу, соответствующую заданным ниже правилам.

Рабочая поверхность

Одометр перемещается по воображаемой квадратной сетке размером 256 × 256 клеток. В каждой клетке не может находиться более 15 камней. Каждая клетка задается парой координат (строка, столбец), где каждая координата принадлежит диапазону от 0 до 255 включительно. Для клетки с координатами (i, j) соседними являются клетки с координатами (i - 1, j), (i + 1, j), (i, j - 1) и (i, j + 1), если они существуют. Клетки, находящиеся на первой или последней строке или на первом или последнем столбце, называются границей. Изначально одометр всегда находится в клетке с координатами (0, 0) (северо-западный угол) и ориентирован на север.

Основные команды

Одометр может быть запрограммирован с помощью следующих команд.

- `left` — повернуться на 90 градусов налево (против часовой стрелки) и остаться в текущей клетке; например, если одометр был ориентирован на юг, то после выполнения этой команды он будет ориентирован на восток.
- `right` — повернуться на 90 градусов вправо (по часовой стрелке) и остаться в текущей клетке; например, если одометр был ориентирован на запад, то после выполнения этой команды он будет ориентирован на север.
- `move` — переместиться вперед, т.е в том направлении, в котором ориентирован одометр, в соседнюю клетку. Если такой клетки нет (например, граница в направлении ориентации одометра уже достигнута), то команда ничего не делает.
- `get` — удалить один камень из текущей клетки. Если в текущей клетке нет камней, то команда ничего не делает.
- `put` — добавить один камень в текущую клетку. Если в текущей клетке уже есть 15 камней, то команда ничего не делает. У одометра никогда не кончаются камни.

- `halt` — завершить выполнение программы.

Одометр выполняет команды в том порядке, в котором они заданы в программе. Программа должна содержать не более одной команды в строке. Пустые строки игнорируются. Символ # обозначает начало комментария; любой текст до конца строки после него игнорируется. Если одометр достигает конца программы, выполнение программы завершается.

Пример 1

Расотрим следующую программу для одометра. После её выполнения, одометр оказывается ориентированным на восток в клетке (0, 2). Обратите внимание, что первая команда `move` будет проигнорирована, так как одометр находится в северо-западном углу и ориентирован на север.

```
move  # не делает ничего
right
# сейчас одометр ориентирован на восток
move
move
```

Метки, границы и камни

Для того, чтобы изменять порядок выполнения команд в зависимости от текущего состояния, можно использовать метки. Метка это регистрозависимая строка, состоящая не более чем из 128 следующих символов: `a`, …, `z`, `A`, …, `Z`, `0`, …, `9`. Новые команды для работы с метками перечислены ниже. Во всех описаниях *L* обозначает любую корректную метку.

- *L*`:` (то есть *L*, за которым следует двоеточие ‘`:`’) — определяет положение метки *L* в программе. Все объявленные метки должны быть уникальными. Объявление метки не оказывает влияния на одометр.

- `jump` *L* — продолжить выполнение, в любом случае перейдя на строку с меткой *L*.

`border` *L* — продолжить выполнение путем перехода на строку с меткой *L*, если одометр находится на границе, причем он ориентирован в сторону границы поля, то есть инструкция `move` ничего не сделает. В противном случае выполнение программы продолжается в обычном порядке и эта команда ничего не делает.

- `pebble` *L* — продолжить выполнение путем перехода на строку с меткой *L*, если в текущей клетке есть хотя бы один камень. В противном случае выполнение программы продолжается в обычном порядке и эта команда ничего не делает.

Пример 2

В результате выполнения следующей программы одометр определяет местоположение первого (самого западного) камня в строке 0 и останавливается там. Если в строке 0 нет камней, то одометр останавливается на границе в конце строки. В этой программе используется две метки: `leonardo` и `davinci`.

```
right
leonardo:
pebble davinci  # найден камень
border davinci  # конец строки
move
jump leonardo
davinci:
halt
```

Сначала одометр поворачивается направо. Цикл начинается с объявления метки `leonardo:` и заканчивается командой `jump leonardo`. В этом цикле одометр проверяет наличие камня в текущей клетке или то, что он достиг границы. Если это не так, то одометр выполняет команду `move` и перемещается из клетки (0, j) в клетку (0, j + 1). Здесь можно обойтись без команды `halt`, так как все равно программа закончит выполнение после достижения последней строки.

Условие

Вы должны отправить программу на языке одометра, который описан выше. В каждой подзадаче, которые определены ниже, требуется реализовать определенное поведение одометра, при этом должны быть соблюдены два следующих ограничения.

- *Размер программы* — программа должна быть достаточно короткой. Размер программы - это количество команд в ней. Объявления меток, комментарии и пустые строки не учитываются при вычислении размера программы.
- *Количество операций* — это программа должна быть достаточно быстрой. Количество операций - это количество выполненных шагов: каждая отдельная команда считается как один шаг, вне зависимости от того, имела ли эффект эта команда или нет. Объявления меток, комментарии и пустые строки не считаются шагом.

В примере 1 размер программы равен 4 и количество операций равно 4. В примере 2 размер программы равен 6. В случае запуска программы с одним камнем в клетке (0, 10) количество операций будет равно 43: `right`, 10 итерациям цикла, каждая итерация занимает 4 шага (`pebble davinci`; `border davinci`; `move`; `jump leonardo`), и наконец, исполняются команды `pebble davinci` и `halt`.

Подзадача 1 [9 баллов]

В начале в клетке (0, 0) находится x камней, и y камней в клетке (0, 1), в то время, как все остальные клетки пустые. Обратите внимание, что в любой клетке может быть не более 15 камней. Требуется написать программу, после выполнения которой одометр будет находится в клетке (0, 0), если $x \leq y$, и в клетке (0, 1) - в противном случае. Нас не интересует направление ориентации одометра; нас так же не волнует количество камней и их расположение.

Ограничения: размер программы ≤ 100, количество операций ≤ 1 000.

Подзадача 2 [12 баллов]

Такая же задача, как вышеописанная, но после завершении программы клетка (0, 0) должна содержать ровно x камней и клетка (0, 1) должен содержать ровно y камней.

Ограничения: размер программы ≤ 200, количество операций ≤ 2 000.

Подзадача 3 [19 баллов]

В строке с индексом 0 находится ровно два камня. Один - в клетке (0, x), другой - в клетке (0,y). x и y различны, (x + y) - четно. Требуется написать программу, в результате выполнения которой одометр окажется в клетке (0, (x + y) / 2), то есть, ровно по середине между двумя клетками, которые содержали камни. Конечное расположение камней не имеет значения.

Ограничения: размер программы ≤ 100, количество операций ≤ 200 000.

Подзадача 4 [до 32 баллов]

На доске находится не более 15 камней, никакие две из них не находятся в одной клетке. Требуется написать программу, которая собирает все камни в северо-западном углу. Если быть более точным, то, если исходно на доске было x камней, тогда после завершения программы в клетке (0, 0) должно находиться ровно x камней, а в остальных клетках камней не должно быть.

Количество баллов за эту подзадачу зависит от количества операций, совершенных посланной на проверку программой. В частности, если L - это максимальное количество операций для различных входных данных, соответствующих этой подгруппе, то вы получите следующее количество баллов:

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

Ограничения: размер программы ≤ 200.

Подзадача 5 [до 28 баллов]

В каждой клетке может быть любое количество камней (разумеется, от 0 до 15). Требуется написать программу, которая находит минимум, то есть после её завершения одометр находится в такой клетке (i, j), что любая другая клетка содержит как минимум столько же камней, сколько и клетка (i, j). После выполнения программы количество камней в каждой клетке должно быть таким же, как и до запуска программы.

Количество баллов за эту подзадачу зависит от размера программы P. В частности, вы получите следующее количество баллов:

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

Ограничения: количество операций ≤ 44 400 000.

Детали реализации

Вы должны отправить на проверку ровно один файл по каждой подзадаче, составленный по вышеописанным правилам. Размер каждого файла не должен превосходить 5 мебибайт. Для каждой подзадачи ваша программа для одометра будет протестирована на нескольких наборах входных данных, и вы получите некоторую информацию о ресурсах, использованных вашей программой. В том случае, если ваш код не является синтаксически корректным, то вы получите информацию о виде синтаксической ошибки.

Вам не обязательно одновременно посылать на проверку все программы для всех подзадач. Если ваша текущая посылка не содержит программы для подзадачи X, то самая последняя посылка по этой подзадаче будет включена в эту посылку. Если вы не посылали на проверку программу для решения этой подзадачи, то эта подзадача в этой посылке будет оценена нулем баллов.

Как обычно, количество баллов за посылку равно сумме баллов за все подзадачи, и итоговый балл за задачу будет равен максимальному баллу среди release-tested посылок и последней посылки.

Симулятор

Для отладки вам будет предоставлен симулятор одометра, которому необходима программа и входная сетка. Программа для одометра должна удовлетворять тому же формату, который используется для посылок, который описан выше.

Сетка должна быть описана в следующем формате: каждая строка файла должна содержать три числа, R, C, P, обозначающих клетку в строке R и столбце C, содержащую P камней. Подразумевается, что все не упомянутые в файле клетки не содержат камней. Для примера рассмотрим следующий файл:

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

Сетка, описанная в этом файле, содержит 15 камней: 3 в ячейке (0, 10) и 12 в ячейке (4, 5).

Вы можете запустить симулятор путем вызова программы `simulator.py` в папке с задачей, передав ей файл с программой в качестве аргумента. Программа-симулятор будет принимать следующие параметры командной строки:

- `-h` выведет вам краткий обзор всех возможных опций;
- `-g GRID_FILE` загрузит описание сетки из файла `GRID_FILE` (по умолчанию сетка пуста);
- `-s GRID_SIDE` устанавливает размер сетки `GRID_SIDE x GRID_SIDE` (значение по умолчанию равно 256, как указано в условии задачи); использование меньших сеток может быть полезно для отладки программы;
- `-m STEPS` ограничивает количество исполняемых симулятором шагов значением `STEPS`;
- `-c` переходит в режим компиляции. В режиме компиляции симулятор возвращает такой же вывод, но вместо симуляции программы с использованием Python, он генерирует небольшую программу на C. Это вызывает увеличение задержки на старте, но позволяет получать результаты существенно быстрее. Мы советуем вам использовать этот параметр когда предполагается, что ваша программа будет выполнять больше чем примерно 10 000 000 шагов.

Количество посылок

Максимальное количество посылок по задаче равно 128.