# Библиотека квотирования

Молодой, но очень перспективный разработчик Олег к концу испытательного срока настолько преисполнился идеей аккаунтинга, лимитирования и квотирования запросов, что решил, что каждый его сервис будет поддерживать эти технологии. Проведя ночь с ноутбуком в обнимку, он написал целую библиотеку, которая позволяла бы ограничивать рейт запросов в свой сервис, основываясь на аутентификационной информации (проще говоря, токену).

Первый MVP библиотеки позволял отдельно взятому пользователю ходить в его сервис не более N*N* раз в секунду. Таким образом, если в момент времени t*t* пользователь u*u* отправил запрос, то он будет пропущен (принят), если пользователь u*u* сделал с момента времени t−1 сек.*t*−1 сек. меньше N*N* запросов.

Смекнув, что в случае нерабочего кода тут пахнет серьезными денежными потерями, Олег покрыл практически всю свою библиотеку unit-тестами.

Тесты подавали на вход записанный упрощенный лог запросов, вида:

1 10000

1676217286000 AYjcyMzY3ZDhiNmJkNTY

1676217286009 RjY2NjM5NzA2OWJjuE7c

1676217286010 AYjcyMzY3ZDhiNmJkNTY

1676217286010 RjY2NjM5NzA2OWJjuE7c

1676217286100 AYjcyMzY3ZDhiNmJkNTY

1676217286199 RjY2NjM5NzA2OWJjuE7c

1676217287012 RjY2NjM5NzA2OWJjuE7c

1676217287612 RjY2NjM5NzA2OWJjuE7c

...

1676217892348 Th7UHJdLswIYQxwSg29D

Где первая строка указывала рейт запросов в секунду и количество записей лога запроса, а последующие - лог запросов, в которых первое поле - расширенный unix timestamp с миллисекундной точностью, а второе - OAuth токен пользователя.

На выходе тесты проверяли статус запроса: будет ли он пропущен или нет. При этом выводились **только пропущенные** запросы:

1676217286000 AYjcyMzY3ZDhiNmJkNTY

1676217286009 RjY2NjM5NzA2OWJjuE7c

...

1676217892348 Th7UHJdLswIYQxwSg29D

К сожалению, утром за день до конца испытательного срока Олег узнал, что его библиотека умерла вместе с жестким диском, но зато остались живы тесты, которые он успел закоммитить. Олег просит Вас помочь с его проблемой, восстановив утерянную функциональнось.

## Формат ввода

На вход даются два числа N*N* и K*K*, разделенные пробельными символами, где 1≤N≤1061≤*N*≤106, 1≤K≤1061≤*K*≤106.

Далее следуют K*K* строк, каждая из которых содержит 2 поля: расширенный Unix-timestamp с **миллисекундной точностью** (unsigned 64-bit integer), и строку - токен, идентифицирующий пользователя. Токен имеет размер в точности равный 20 символов из [a-zA-Z0-9] алфавита.

Гарантируется, что ввод отсортирован по первому полю (timestamp) и что каждая строка всегда оканчивается символом \n. Суммарное количество пользователей не превышает 1000.

## Формат вывода

В ответе нужно вывести тот же самый лог запросов, оставив из него только те запросы, которые будут приняты (не будут отфильтрованы).

Обратите внимание, что каждую строку следует выводить без изменений, включая символ \n в конце строки.

### Пример 1

Ввод

1 1

1679981968781 0fdc847ecc495d6dd584

Вывод

1679981968781 0fdc847ecc495d6dd584

### Пример 2

Ввод

2 4

1679982033000 441079aa62dc3cd57df3

1679982033998 441079aa62dc3cd57df3

1679982033999 441079aa62dc3cd57df3

1679982034000 441079aa62dc3cd57df3

Вывод

1679982033000 441079aa62dc3cd57df3

1679982033998 441079aa62dc3cd57df3

1679982034000 441079aa62dc3cd57df3

1 3

0 AAAABBBBCCCCDDDDEEEE

999 AAAABBBBCCCCDDDDEEEE

1000 AAAABBBBCCCCDDDDEEEE

corr = 2

2 4

0 AAAABBBBCCCCDDDDEEEE

998 AAAABBBBCCCCDDDDEEEE

999 AAAABBBBCCCCDDDDEEEE

1000 AAAABBBBCCCCDDDDEEEE

corr = 3

1 4

0 AAAABBBBCCCCDDDDEEE1

0 AAAABBBBCCCCDDDDEEE2

0 AAAABBBBCCCCDDDDEEE3

0 AAAABBBBCCCCDDDDEEE4

corr = 4

1 8

0 AAAABBBBCCCCDDDDEEE1

0 AAAABBBBCCCCDDDDEEE2

0 AAAABBBBCCCCDDDDEEE3

0 AAAABBBBCCCCDDDDEEE4

990 AAAABBBBCCCCDDDDEEE1

995 AAAABBBBCCCCDDDDEEE2

999 AAAABBBBCCCCDDDDEEE3

1000 AAAABBBBCCCCDDDDEEE4

corr = 5

# Музыкальный лейбл

Музыкальный лейбл аккумулировал вокруг себя тысячи артистов и собрал в своем архиве бессчетное количество треков, музыкальных партий и битов. Накопленной музыкой лейбл активно делится со своими нынешними артистами.

Архив с музыкой – это дерево ресурсов, в котором есть два типа узлов: директории и файлы. Файлы – это узлы у которых нет “детей”. То есть листовые узлы, которые содержат конкретное музыкальное произведение. Директории – это узлы у которых могут быть “дети”, по сути это папка с музыкальными произведениями.

У каждого узла есть имя. У “детей” одной директории уникальные имена. Полный путь определяется названиями узлов разделенные знаками /, например:

/ – это корневая директория.

/Release – это директория с именем Release внутри корневой директории.

/Release/Echoes\_of\_Eternit – это директория Echoes\_of\_Eternit внутри директории Release.

/Release/Echoes\_of\_Eternit/Supernova.flac – это файл Supernova.flac внутри директории Echoes\_of\_Eternit.

Недавно лейбл столкнулся с проблемой: сломался сервис выдачи прав и работа многих артистов остановилась. Если артисты не работают, лейбл теряет деньги, а слушатели не получают новые релизы. Вас позвали на помощь. Ваша задача: написать сервис, который выдает пользователям (артистам и администраторам) права на узлы дерева и эти права проверяет.

По умолчанию, у пользователей отсутствуют какие-либо права. Есть два режима прав, read и write – права на чтение и на запись файлов и директорий.

Чаще всего артисты получают права на чтение (они нужны, чтобы прослушивать и семпилировать музыкальные треки), а администраторы лейбла права на запись (админы чаще просто добавляют новые релизы). Права на чтение и на запись выдаются отдельно. У пользователя могут быть права как на чтение, так и на запись. При этом последняя операция выдачи права является более приоритетной.

Есть три операции:

grant user mode path – выдает доступ с режимом mode пользователю user на узел с путем равным path. Ответ – это строка done

block user mode path – запрещает доступ с режимом mode пользователю user на узел c путем равным path. Ответ – это строка done

check user mode path – проверяет, есть ли у пользователя user права с режимом mode до узла с путем равным path. Ответ – это строка allowed или forbidden.

Все выданные права и запреты наследуются вниз по дереву. При этом, если возникает противоречие, правила, определенные ниже по дереву, имеют более высокий приоритет.

## Формат ввода

В каждой строке входного файла записана одна из команд grant, block или check, с параметрами user*user*, mode*mode* и path*path*.

* user*user* &ndash; имя пользователя, непустая строка, длина не превышает 2020.
* mode*mode* &ndash; режим доступа, строка со значением read или write.
* path*path* &ndash; путь до узла, непустая строка, всегда начинается с символа /, длина не превышает 5000050000.
* Количество команд не превышает 5000050000.
* Количество пользователей не превышает 5000050000.
* Количество узлов в дереве ресурсов не превышает 100000100000.
* Имя узла &ndash; непустая строка, длина не превышает 2020.
* Разрешенные символы в названиях узлов и именах пользователей: a-z, A-Z, 0-9, ., \_.
* Размер входного файла не превышает 1MiB1*MiB*.

## Формат вывода

Для каждой команды выведите результат ее выполнения.

* Для команд grant и block, это всегда строка со значением done.
* Для команды check, это строка со значением allowed, если доступ разрешен, или forbidden, если в доступе отказано.

## Пример

Ввод

grant bob read /home/bob

block bob read /

check bob read /home/bob/Pictures/gary.jpg

check bob read /etc/shadow

check bob write /home/bob/Pictures/gary.jpg

grant bob write /home/bob

check bob write /home/bob/Pictures/gary.jpg

block bob write /home/bob/Pictures

check bob write /home/bob/Pictures/gary.jpg

check bob read /home/bob/Pictures/gary.jpg

grant root read /

check root read /home/bob/Pictures/gary.jpg

check root read /etc/shadow

Вывод

done

done

allowed

forbidden

forbidden

done

allowed

done

forbidden

allowed

done

allowed

allowed