Задача A. Логи Apache

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

8 октября в Центр была передана информация от Штирлица — лог сервера Apache, отражающий доступ к ключевому интернет-ресурсу, расположенному в государстве Borderland. В результате успешной спецоперации в Borderland сложилась тупиковая ситуация вокруг референдума о том, какая из букв нового алфавита должна быть первой. После предыдущих итераций выбор сузился до `Y` и `Z`. Очередная итерация процесса выхода из кризиса намечалась уже через пять дней. Для того чтобы восстановить последовательность запросов, сотрудники Центра, не совсем пришедшие в себя после Дня Чекиста, решили перевести события лога к временной зоне секретной базы, на которой разрабатывается новая диверсия.

Формат входных данных

В первой строке записана временная зона секретной базы, к которой нужно преобразовать лог. Начиная со второй строки, задан сам лог. Строки лога Apache всегда имеют вид

$$\langle ip \rangle \ \texttt{-} \ \langle username \rangle \ \langle date \rangle \ \langle extra\ fields \rangle$$

Поле *ip* представляет собой ip-адрес узла, сделавшего запрос, в стандартном формате. Второе поле всегда содержит один символ «`-`». Имя пользователя состоит из произвольного ненулевого количества латинских букв, символов «`-`» и кавычек. Формат даты следующий:

$$[\langle день \rangle / \langle месяц \rangle / \langle год \rangle : \langle часы \rangle : \langle минуты \rangle : \langle секунды \rangle \ \ \langle временная\ зона \rangle]$$

Временная зона всегда записывается ровно пятью символами:

$$\langle знак \rangle \langle часы \rangle \langle минуты \rangle$$

Знак принимает значение «`+`», если часы в данной временной зоне опережают часы в зоне GMT (Greenwich Mean Time) или показывают то же время, в противном случае ставится знак «`-`».

День, часы, минуты и секунды всегда состоят ровно из двух цифр (как и в поле даты, так и во всех записях временных зон), год — из четырех. Месяц записан сокращением на английском языке и может принимать одно из следующих значений: `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, `Dec`.

Допустимые во входных данных номера года лежат в диапазоне от 1980 до 2099. Високосными считаются года, номер которых делится на 4.

Дополнительные поля состоят из произвольного количества символов с кодами от 32 до 126.

В записях временной зоны поле часов всегда содержит значение, не превосходящее 12. Размер входных данных не превосходит одного мегабайта. Все даты корректны и проверка их не требуется.

Формат выходных данных

Результатом работы программы является лог Apache, в котором все времена и даты преобразованы к заданной временной зоне. Все остальные поля в каждой строке лога должны оставаться без изменений. Имейте в виду, что проверка производится автоматически, поэтому все даты и времена должны содержать по две цифры в полях дней, часов, минут и секунд, три символа в поле месяца (case-sensitive), и четыре цифры в поле года.

Пример

стандартный ввод
-0200 195.19.224.104 - abc [02/Dec/2004:18:25:19 +0100] "GET / HTTP/1.1" 304 - 195.19.224.104 - - [02/Dec/2004:18:25:19 +0100] "GET /main.css HTTP/1.1" 304 - 195.19.224.79 - - [02/Dec/2004:20:58:20 +0300] "GET /tts HTTP/1.1" 302 293 195.19.224.79 - - [02/Dec/2004:20:58:20 +0300] "GET /tts/ HTTP/1.1" 302 303 207.46.98.42 - - [03/Dec/2004:04:39:32 +0300] "GET /robots.txt HTTP/1.0"
стандартный вывод
195.19.224.104 - abc [02/Dec/2004:15:25:19 -0200] "GET / HTTP/1.1" 304 - 195.19.224.104 - - [02/Dec/2004:15:25:19 -0200] "GET /main.css HTTP/1.1" 304 - 195.19.224.79 - - [02/Dec/2004:15:58:20 -0200] "GET /tts HTTP/1.1" 302 293 195.19.224.79 - - [02/Dec/2004:15:58:20 -0200] "GET /tts/ HTTP/1.1" 302 303 207.46.98.42 - - [02/Dec/2004:23:39:32 -0200] "GET /robots.txt HTTP/1.0"

Задача B. Статистика звонков

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Андрюша постоянно жалуется папе, что его младший братик Саша часами сидит в интернете и не даёт ему готовиться к олимпиаде. Папа решил всерьёз заняться ситуацией, так как Саша действительно, похоже, превышает отведённый ему на интернет лимит времени. Для выяснения того, сколько же Саша просидел в интернете, папа решил написать анализатор логов модема от Windows'95. Единственная проблема заключается в том, что Windows ведет лог в крайне неудобочитаемом формате и выводит туда слишком много бессмысленной информации...

Лог модема Windows состоит из строчек, начинающихся датой и временем в формате mm-dd-yyyy hh:mm:ss.ss, то есть две цифры месяца, дефис, две цифры дня, дефис, четыре цифры года, пробел, часы, двоеточие, минуты, двоеточие, секунды, точка, сотые доли секунды. Затем следует пробел, дефис, еще один пробел и далее сама строчка.

Например:

```
04-24-2003 14:51:55.48 - Connection established at 24000bps.
```

Кроме этого, в файле лога могут встречаться строчки, не подходящие под данный формат. Их анализатор должен игнорировать.

Когда анализатор видит строчку вида «`Connection established at 24000bps.`», он засчитывает этот момент как момент начала соединения. Скорость соединения также собирается для статистики.

Соединение считается завершённым в момент вывода в лог строчки `Hanging up the modem.`. Кроме того, строчки такого вида могут появляться в местах, не связанных с соединением (например, в момент дозвона).

Кроме этого, через несколько строчек после `Hanging up the modem.`, но до строчки `Standard Modem closed.` идут две строки, в которых написано, сколько байт было прочитано и записано во время этой сессии с интернетом.

Заодно папа решил посчитать, сколько же минут придётся оплачивать. Известно, что любое соединение длительностью менее 60 секунд — бесплатное, остальные же считаются с точностью до секунд и складываются (сотые доли секунды папин анализатор вообще полностью игнорирует).

Вам необходимо написать анализатор, который работает так же, как и папин. Интересно же, сколько братик сидит в интернете :-)

Формат входных данных

Входной файл состоит из одной или нескольких строчек, составляющих лог модема Windows. Часть из этих строчек может не подходить под описанный формат. Гарантируется, что файл содержит только символы с кодами от 32 до 255, а также переводы строки. Длина любой строчки не превосходит 250 символов. Длина файла не превышает 128 килобайт, так как большие логи Windows зачем-то очень странным образом обрезает.

Гарантируется, что в логе нет сессий длиной больше 12 часов. Кроме того, входной лог корректен, и не может содержать два начала сессии без промежуточного завершения.

Формат выходных данных

В выходной файл необходимо вывести список всех сессий в интернете, которые имели место. Для каждой сессии нужно вывести момент её начала в формате mm-dd-yyyy hh:mm:ss, затем длительность соединения в секундах, скорость, на которой было установлено соединение, а также количество прочитанных и записанных байт.

После списка сессий нужно вывести общую статистику в формате `Total seconds to pay = xxxxx, total seconds = yyyyy; total bytes zzzzz/ttttt`.

Следуйте формату вывода, указанному в примере и в условии, максимально точно — проверка производится автоматически!

Пример

стандартный ввод

```
04-24-2003 00:02:14.72 - Standard Modem in use.
04-24-2003 00:02:14.72 - Modem type: Standard Modem
04-24-2003 00:02:14.72 - Modem inf path: MICROS~1.INF
04-24-2003 00:02:14.72 - Modem inf section: PNPC107
04-24-2003 00:02:15.24 - Initializing modem.
04-24-2003 00:02:15.24 - Send: AT<cr>
04-24-2003 00:02:15.24 - Recv: AT<cr>
04-24-2003 00:02:15.36 - Recv: <cr><lf>OK<cr><lf>
04-24-2003 00:02:15.36 - Interpreted response: Ok
04-24-2003 00:02:15.36 - Send: ATE0V1<cr>
04-24-2003 00:02:15.37 - Recv: ATE0V1<cr>
04-24-2003 00:02:15.49 - Recv: <cr><lf>OK<cr><lf>
04-24-2003 00:02:15.49 - Interpreted response: Ok
04-24-2003 00:02:16.69 - Dialing.
04-24-2003 00:02:16.69 - Send: ATDT#######<cr>
04-24-2003 00:02:16.69 - Recv: ATDT4284778<cr>
04-24-2003 00:02:24.42 - Recv: <cr><lf>BUSY<cr><lf>
04-24-2003 00:02:24.42 - Interpreted response: Busy
04-24-2003 00:02:24.42 - Hanging up the modem.
04-24-2003 00:02:24.42 - Send: ATH<cr>
04-24-2003 00:02:24.42 - Recv: ATH<cr>
04-24-2003 00:02:24.55 - Recv: <cr><lf>OK<cr><lf>
04-24-2003 00:02:24.55 - Interpreted response: Ok
04-24-2003 00:02:24.61 - Standard Modem closed.
#@$@#$#@$#$#@$$#@%$$#@$%$#%@#$@%#$%#@%# Windows errfhmvcxlj System is unstabl.c,mx
04-24-2003 00:10:29.07 - Dialing.
04-24-2003 00:10:29.07 - Send: ATDT#######<cr>
04-24-2003 00:10:29.08 - Recv: ATDT4284779<cr>
04-24-2003 00:10:49.65 - Recv: <cr>
04-24-2003 00:10:49.65 - Interpreted response: Informative
04-24-2003 00:10:49.65 - Recv: <lf>
04-24-2003 00:10:49.65 - Interpreted response: Informative
04-24-2003 00:10:49.65 - Recv: CONNECT
04-24-2003 00:10:49.65 - Interpreted response: Connect
04-24-2003 00:10:49.65 - Connection established at 57600bps.
04-24-2003 00:10:49.65 - Error-control off or unknown.
04-24-2003 00:10:49.65 - Data compression off or unknown.
04-24-2003 00:10:49.65 - 57600,N,8,1
04-24-2003 00:12:39.70 - Remote modem hung up.
04-24-2003 00:12:39.71 - Recv: <cr><lf>NO CARRIER<cr><lf>
04-24-2003 00:12:39.71 - Interpreted response: No Carrier
04-24-2003 00:12:39.71 - Hanging up the modem.
04-24-2003 00:12:39.71 - Send: ATH<cr>
04-24-2003 00:12:39.71 - Recv: ATH<cr>
04-24-2003 00:12:39.84 - Recv: <cr><lf>OK<cr><lf>
04-24-2003 00:12:39.84 - Interpreted response: Ok
04-24-2003 00:12:39.90 - Session Statistics:
04-24-2003 00:12:39.90 -                Reads : 60870 bytes
04-24-2003 00:12:39.90 -                Writes: 8179 bytes
04-24-2003 00:12:39.90 - Standard Modem closed.
```

стандартный вывод

```
04-24-2003 00:10:49    110 57600 60870/8179
Total seconds to pay = 110, total seconds = 110; total bytes 60870/8179
```

Задача C. Деление уголком

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Вы отправились на машине времени в далёкое прошлое Франции с важной исторической миссией. Для укрепления своего положения в обществе вам необходимо получить учёную степень.

Явившись в ближайший университет, вы с удивлением обнаружили, что для этого необходимо всего лишь продемонстрировать умение делить числа уголком. Для вас это просто, но для стопроцентной гарантии вы решили попросить компьютер проделать аналогичные вычисления.

Напишите программу, которая выводит процесс деления двух десятичных чисел уголком.

Формат входных данных

В двух строках входных данных заданы делимое и делитель, меньшие 10^{100}.

Формат выходных данных

Выведите процесс деления. В точности следуйте формату примера. Никакие числа в процессе вывода не могут иметь ведущие нули. Сравнение будет производиться посимвольно. Делимое должно быть отделено от вертикальной черты ровно одним пробелом.

Примеры

стандартный ввод	*стандартный вывод*

50082003928
491
	50082003928 |491
	491 +---------
	----- |102000008
	  982
	  982
	  --------- 	
	  3928
	  3928
	  ---- 	
	  0

239
717
	239 |717
	  +---
	  |0

239 17	239 |17 17 +-- --- |14 69 68 -- 1
667700 6677	667700 |6677 6677 +---- ------ |100 0
12 7	12 |7 7 +- -- |1 5

Задача D. Яйца

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

У Гоши на дверце холодильника есть лоток для 16 яиц: два ряда углублений, по восемь углублений в каждом ряду, с неподвижным разделителем посередине. Изначально в холодильнике нет яиц.

Когда в лотке есть хотя бы одно яйцо, Гоша может взять одно из них и съесть. Когда в лотке не больше пяти яиц, Гоша может пойти в магазин, купить ещё десяток яиц и поставить их в лоток. Обратите внимание: после покупки новых яиц и расстановки их в лотке всегда останется хотя бы одно свободное углубление.

Чтобы яйца не лежали слишком долго и не портились, Гоше нужно каждый раз брать для еды одно из самых старых яиц: тех, что куплены раньше всего. Он хочет расставлять купленные яйца так, чтобы по положению яиц в лотке всегда было понятно, какие яйца самые старые. Помогите Гоше придумать стратегию: куда класть купленные яйца и откуда брать очередное яйцо для еды, чтобы всегда брать одно из самых старых.

Формат входных данных

В этой задаче два теста. В первом тесте в первой строке ввода записано слово «`sample`», а во втором — слово «`test`». В первом тесте проверяется только корректность вывода, а во втором — ещё и правильность стратегии.

Формат выходных данных

Выведите стратегию в виде нескольких инструкций.

Каждая инструкция состоит из начального состояния (слева), действия (посередине) и конечного состояния (справа). Пожалуйста, следуйте формату, указанному в примере, как можно более точно! Формальные правила вывода описаны после примера.

Действие «`buy`» должно добавлять ровно десять яиц, а действие «`eat`» — удалять ровно одно яйцо. Одно и то же начальное состояние может встречаться в инструкциях не более двух раз: не более одного раза с действием «`buy`» и не более одного раза с действием «`eat`».

Оформление и все правила, описанные выше, проверяются на обоих тестах к задаче. А вот проверка самой стратегии, описанная ниже, производится только на втором тесте.

Гоша пользуется стратегией так. Сначала выбирает очередное возможное действие: он может съесть яйцо, если в холодильнике есть хотя бы одно, или расставить десять купленных яиц, если в холодильнике их не больше пяти. Далее Гоша находит инструкцию с таким действием, в которой начальное состояние совпадает с фактическим положением яиц у него в холодильнике. После этого Гоша меняет состояние на конечное состояние этой инструкции.

Если существует последовательность возможных действий, после которой Гоша не может найти нужную инструкцию, или по инструкции выбирает для еды яйцо, не являющееся самым старым, стратегия считается неверной. Если при любой последовательности возможных действий Гоша найдёт нужную инструкцию, а ест всегда только одно из самых старых яиц, стратегия считается верной. Обратите внимание: стратегия должна учитывать все последовательности возможных действий: когда у Гоши два варианта следующего действия (есть или покупать), оба должны быть учтены.

Разрешается выводить инструкции с начальными состояниями, которые недостижимы при любой последовательности действий.

Примеры

стандартный ввод	*стандартный вывод*
sample	****|**** eat ****|**** ...*|**** ....|**** --- ....|.... buy ...*|**** ....|.... ...*|**** ---

Замечание

Правила оформления инструкций таковы. Инструкция занимает три строки: верхние две строки описывают состояния и действие, а за ними следует строка с тремя символами «-» (минус, ASCII-код 45). В каждом состоянии пустые углубления обозначаются символом «.» (точка, ASCII-код 46), яйца — символом «*» (звёздочка, ASCII-код 42), а разделитель — символом «|» (вертикальная черта, ASCII-код 124). Действия обозначаются словами «buy» (добавляется десять новых яиц) и «eat» (удаляется одно яйцо). В первой строке действие отделяется от состояний одинарными пробелами, во второй строке состояния разделены пятью пробелами.

Задача E. Мультиплексор

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Мультиплексор в TestSys — это программа для одновременной организации контестов в разных местах с использованием TestSys. Например, если в одно и то же время проходят тренировки в Петергофе и Санкт-Петербурге, самый простой способ их организовать — с помощью мультиплексора. Тренировки по интернету тоже проводятся при помощи мультиплексора.

В этой задаче вам предстоит реализовать часть функциональности мультиплексора. Программа должна прочитать конфигурационный файл и, далее, определить место назначения для каждого входящего пакета.

Конфигурационный файл мультиплексора состоит из нескольких разделов. Описание каждого раздела соответствует одному порту. Заголовок раздела состоит из одной строки вида:

A·<*port*>

где <*port*> — произвольное целое число между 0 и 65 535, а '·' означает произвольное количество пробелов (ASCII-код 32) и табуляций (ASCII-код 9). Для каждого порта определены некоторые правила. Описание каждого правила находится на отдельной строке, выглядящей так:

F·<*from*>·T·<*to*>[·I·<*id list*>][·P·<*password*>]

Квадратные скобки означают "может быть опущено", как и обычно. Параметр <*from*> может быть либо в форме <*IP*> , либо в форме <*IP*>/<*mask*>. В этой задаче <*IP*> всегда выглядит как <*a*>.<*b*>.<*c*>.<*d*>, где a, b, c и d — любые целые числа в интервале $0..255$. <*mask*> — это целое число между 0 и 32, интерпретируется следующим образом. Каждый IP-адрес можно рассматривать как 32-битное двоичное число: a соответствует битам с 31 по 24 (старшие), b — битам с 23 по 16, c — с 15 по 8, d — с 7 по 0 (младшие). Эти биты записаны в обычном порядке, то есть, например, старший бит a является старшим битом всего числа, а младший бит d является младшим битом всего числа. Например, адрес 195.19.228.2 будет соответствовать двоичному числу $11000011000100111110010000000010_2$=0xC313E402=3272860674 (или −1022106622, если рассматривать знаковый тип). Значение маски определяет количество значимых старших битов адреса. Остальные биты могут быть произвольными. Например, под `195.19.228.2/24` подходит любой адрес, начинающийся на `195.19.228` ("`195.19.228.*`"), а `195.19.228.2/20` означает, что адрес должен быть в интервале `195.19.224.0`–`195.19.239.255`. Если параметр <*mask*> опущен, маска считается равной 32 (все биты значимые). Пакет подходит под правило, если источник пакета находится в указанном диапазоне.

Есть ещё три условия для того, чтобы пакет подошёл под правило. Во-первых, конечно, пакет должен быть получен через порт <*port*>, который указан в заголовке раздела. У всех разделов указаны разные порты. Ещё два правила — это правила соответствия идентификатора и пароля. Правило соответствия пароля очень простое: пароль, содержащийся в правиле, должен совпадать (с учётом регистра) с паролем к правилу (если правило не содержит пароля, любой принимается, если в пакете нету пароля, он принимается только в случае отсутствия пароля у правила). Пароль всегда состоит из не более, чем 64 букв и цифр.

Правило соответствия идентификатора несколько сложнее. <*id list*> может иметь вид <*id*> или <*firstid*>-<*lastid*>. В первом случае необходимо точное совпадение <*id*> и id, указанного в пакете (с теми же оговорками, что и в правиле соответствия пароля, аналогично, идентификатор состоит из не более, чем 64 букв и цифр). Во втором случае <*firstid*> и <*lastid*> должны быть целыми числами одинаковой длины (ведущие нули разрешены). Пакет удовлетворяет правилу, если идентификатор, указанный в пакете имеет ту же длину, что и <*firstid*> и <*lastid*>, является числом и лежит между <*firstid*> и <*lastid*> включительно.

Длины этих чисел во входном файле никогда не будут больше пяти.

В случае, когда пакет удовлетворяет всем четырём условиям, он должен быть направлен хосту <*to*>, где <*to*> — IP-адрес (<*IP*>). Правила проверяются в порядке их следования. Как только найдено удовлетворяющееся правило, пакет немедленно перенаправляется и система переходит к обработке следующего пакета. Если пакет не подходит ни под одно правило, он перенаправляется в *NULL device* (то есть игнорируется).

Формат входных данных

Входной файл начинается с нескольких строк, описывающих конфигурацию мультиплексора. После последнего раздела идёт разделяющая строка, состоящая из трёх дефисов ("`---`"). После этого начинаются пакеты. Пакеты описываются в том же виде, что и правила, с тремя исключениями: во-первых, в адресе источника не разрешается маска (<*mask*>), во-вторых, адрес назначения отсутствует (поскольку мы знаем, что этот пакет для нас), параметр <*to*> содержит номер порта, на который пришёл пакет, в-третьих, идентификатор не может быть указан в виде диапазона (<*firstid*>-<*lastid*>).

В файле нет пустых строк. Количество разделов не менее одного и не более 32, и для каждого порта определено не менее одного и не более 32 правил. Количество пакетов не превышает 3000.

Формат выходных данных

Для каждого пакета выведите на отдельной строке адрес, на который его надо перенаправить. Если пакет перенаправлен в *NULL device*, выведите `/dev/null` вместо адреса.

Пример

стандартный ввод

```
A       12345
F       192.168.64.90/24    T       195.19.228.2
F       195.19.228.2        T       192.168.64.90   I       00-29
A       13456
F       195.19.228.2/28     T       195.19.228.213  P       passwd
F       0.0.0.0/0           T       195.19.228.2    I       ttt     P       secret
A       23917
F       1.2.3.4             T       195.19.228.2    I       1
---
F       192.168.64.239      T       12345           I       abc
F       192.168.65.0        T       12345
F       195.19.228.2        T       12345           I       1
F       195.19.228.2        T       12345           I       23      P       super
F       1.2.3.4             T       23917           I       01
F       1.2.3.4             T       23917           I       1
F       1.2.3.4             T       23917
F       195.19.228.16       T       13456           P       passwd
F       195.19.228.16       T       13456           I       ttt     P       passwd
F       195.19.228.16       T       13456           I       ttt     P       secret
F       195.19.228.15       T       13456           P       passwd
F       195.19.228.15       T       13456           I       ttt     P       passwd
```

стандартный вывод

```
195.19.228.2
/dev/null
/dev/null
192.168.64.90
/dev/null
195.19.228.2
/dev/null
/dev/null
/dev/null
195.19.228.2
195.19.228.213
195.19.228.213
```

Задача F. Умножение полиномов

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Одному математику при работе над диссертацией понадобилось перемножить два полинома от нескольких переменных. Для этого он решил применить программу "MathCAD", но оказалось, что при большой размерности полиномов вычислить их с помощью этой программы не удается. Тогда он обратился к вам за помощью.

Вычислите произведение двух полиномов, заданных аналитически.

Формат входных данных

В первой строке записан первый полином, во второй, соответственно, второй. Полиномы могут содержать переменные, обозначенные символами от 'a' до 'z', целые числа, знаки операций '+', '-', '*', '(', ')' и произвольное количество пробелов. Все промежуточные результаты вычислений удовлетворяют ограничениям 64-битного целого знакового типа.

Формат выходных данных

Выведите результат перемножения двух полиномов с приведёнными подобными членами. Полученное выражение не должно содержать знаков '(' и ')'. Символ '^' означает возведение переменной в числовую степень. Унарная операция '+' недопустима, а унарный '-' может быть выведен только перед первым слагаемым.

Пример

стандартный ввод	*стандартный вывод*
2*(a + 4*b) b-c	2*a*b-2*a*c+8*b^2-8*b*c

Задача G. Стресс-тестирование

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Витю пригласили в жюри межпланетной олимпиады по программированию. Это большая честь для него, и он постарался не ударить в грязь лицом. Он реализовал технологию, которая позволяет находить интересные тесты для задач по имеющимся решениям жюри. Эта технология называется *стресс-тестирование*.

Поиск теста осуществляется по следующей схеме. Сначала пишется генератор общего случая теста к задаче, то есть такой генератор, который с достаточно большими вероятностями выдает тесты на все случаи, которые необходимо разобрать. Он не обязательно является генератором полностью случайного теста, так как зачастую на таких тестах реализуются далеко не все случаи решения.

Генератор, естественно, использует случайные числа для создания теста. Он инициализируется с помощью функции randomize, которая берет случайное значение для инициализации генератора случайных чисел из внешнего источника (как правило, в качестве такого источника используется системное время). Для того чтобы можно было потом восстановить тест, значение, которым был проинициализирован генератор случайных чисел, выдается в файл протокола. Всегда гарантируется, что при одной и той же инициализации генератор случайных чисел будет выдавать одну и ту же последовательность.

После генерации теста на нем запускаются два различных решения жюри. Специальная программа run замеряет время работы каждого из решений. Естественно, вся информация о времени работы, параметрах запуска и используемой памяти также выводится в файл протокола работы системы.

Затем происходит сравнение выходных файлов обеих программ. К сожалению, в связи с особенностями используемой суперсовременной операционной системы Doors, информацию о сравнении не так-то просто вывести в файл протокола, поэтому Витя ограничился лишь прерыванием тестирования в случае, если одно из решений выдало неверный ответ.

Весь этот процесс запускается в цикле, пока не будет найден тест, на котором одна из программ выдает неверный ответ, или же член жюри, занимающийся тестированием, не сочтет, что уже было проверено достаточное количество различных тестов.

В первом случае, очевидно, интересным является тест, на котором завалилось решение жюри. Во втором же случае Витя считает интересными тесты, на которых каждое из двух решений работало максимальное время. К сожалению, исходный файл протокола не очень удобен для получения этой информации.

Необходимо написать программу, которая бы преобразовывала файл протокола в более удобный формат, а также находила бы тесты, на которых каждое из решений работало максимальное время.

Формат входных данных

Во входном файле содержится исходный файл протокола. Его размер не превосходит двух мегабайт. Он состоит из одного или нескольких блоков, каждый из которых соответствует запуску обоих решений на одном тесте. Каждый из этих блоков начинается и заканчивается непустой строкой, состоящей из одних символов '`-`'.

Внутри блока содержится информация об инициализации генератора случайных чисел, а также о времени работы каждого из решений. Инициализация генератора задается строкой в формате `randseed =` x, где x — целое неотрицательное число, не превосходящее 10^9. За ней следуют протоколы запуска решений. Время работы каждого из решений выведено в формате `Work time:` x_i `ms`, где x_i — вещественное число, в котором в качестве разделителя целой и дробной части используется запятая — так уж вывела операционная система Doors. Целая часть этого числа не превосходит 10^6. Витя знает, что операционная система Doors все равно производит измерения таким образом, что дробные части совпадают при совпадении целых частей, поэтому его интересует только целая часть этого числа.

Кроме указанных строк, в файле протокола может содержаться также другая информация, которую Витя пока не хочет анализировать.

Всегда сначала запускается первое решение жюри, а затем второе. Кроме того, запуск решений происходит только после генерации теста, а, следовательно, и вывода в файл протокола информации об инициализации генератора случайных чисел.

Формат выходных данных

Для каждого из блоков в порядке их следования должны быть выведены три строки информации. В первой строке выводится значение, которым был проинициализирован генератор случайных чисел, в формате `At randseed =` x. Во второй и третьей строках выводится время работы первого и второго решений соответственно, в виде `First:` x_1 `ms` и `Second:` x_2 `ms`, где x_i — время работы каждого из решений с отброшенной дробной частью.

Кроме этого, необходимо собрать информацию о том, на каких тестах каждое из решений работало максимальное время. Эта информация выводится после всей информации о блоках. Если какое-то решение работало одинаково долго на нескольких тестах, то выбирается самый первый из блоков, на котором было достигнуто такое время.

Проверка производится автоматически, поэтому вывод вашей программы должен по возможности максимально совпадать с эталонным выводом!

Пример

стандартный ввод

```
------------------------------------------------
randseed = 43794135
Testing ..\b_mb, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (80 ms)
Work time: 80,11520 ms
Maximal memory usage: 10336 Kb
Testing ..\b_al, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (391 ms)
Work time: 390,56160 ms
Maximal memory usage: 2680 Kb
------------------------------------------------
------------------------------------------------
randseed = 43903502
Testing ..\b_mb, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (110 ms)
Work time: 110,15840 ms
Maximal memory usage: 10328 Kb
Testing ..\b_al, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (310 ms)
Work time: 310,44640 ms
Maximal memory usage: 2680 Kb
------------------------------------------------
```

стандартный вывод

```
At randseed = 43794135
First: 80 ms
Second: 390 ms
At randseed = 43903502
First: 110 ms
Second: 310 ms
Maximal work time for first: 110 at randseed = 43903502
Maximal work time for second: 390 at randseed = 43794135
```

Задача H. Таймер

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Рассмотрим следующую конструкцию таймера. Таймер состоит из двух цилиндров: внутреннего и внешнего, который может вращаться вокруг внутреннего. На поверхность внутреннего цилиндра нанесены отметки и числа. Отметки соответствуют минутам. Внешний же цилиндр покрывает весь внутренний, оставляя видимым только маленькое окошко. Время, которое показывает таймер, можно узнать, посмотрев в центр этого окошка.

Всего на внутреннем цилиндре 60 отметок. Каждая пятая из них отмечена соответствующим числом: 0, 5, 10, 15, ..., 55. Расстояния между любыми двумя соседними отметками, включая расстояние между последней и первой отметками, равны. Ширина окошка ровно в пять раз больше расстояния между соседними отметками.

Исходно в центре окошка во внешнем цилиндре находится отметка 0 на внутреннем цилиндре. Чтобы выставить время на таймере, внешний цилиндр поворачивают вокруг внутреннего по часовой стрелке. После этого он медленно поворачивается против часовой стрелки, пока опять не достигнет отметки 0, после чего останавливается. На таймере можно выставить любое время, строго меньшее, чем 60 минут.

Анна учит своего домашнего робота Берту узнавать время, которое показывает таймер. Берта может фотографировать окошко таймера. Помогите Анне написать программу для Берты, чтобы определять время по фотографии.

Фотография окошка — это растровое изображение 60 пикселов в ширину и 12 пикселов в высоту. Будем для простоты считать, что фотография — это идеальное изображение видимой части внутреннего цилиндра, уложенной на плоскость. В таком случае расстояние между соседними отметками будет ровно 12 пикселов. Также для простоты предположим, что фотография сделана в момент, когда время, которое показывает таймер, кратно 5 секундам. Поскольку окошко не совсем прямоугольной формы, небольшие уголки на фотографии всегда закрыты внешним цилиндром; точную форму уголков можно увидеть в примере. Все остальные пикселы фотографии либо полностью белые, либо полностью чёрные.

При указанных предположениях каждая отметка выглядит как чёрный квадрат 2×2 в первых двух строках фотографии; настоящий центр отметки находится между двумя столбцами этого квадрата. Каждое число печатается чёрным шрифтом размера 8×8 в строках с 4-й по 11-ю. Каждое число (из одной или двух цифр) горизонтально выровнено так, что центр соответствующего блока размера 8×8 или 16×8 расположен по центру отметки, при которой написано это число. Все пикселы, не покрытые уголками, отметками или цифрами, отображаются как белые.

Ниже показаны изображения отдельных цифр от 0 до 9 в используемом шрифте 8×8. Сам шрифт идентичен шрифту, использовавшемуся в видеокартах *Rendition Vérité 1000* в 1990-е годы. Изображения цифр в шрифте можно также скачать по следующему адресу: `https://tsweb.ru/trains/timer/`.

```
.XXXXX.. ...XX... .XXXXX.. .XXXXX.. ...XXX.. XXXXXXX. .XXXXX.. XXXXXXX. .XXXXX.. .XXXXX..
XX..XXX. ..XXX... XX...XX. XX...XX. ..XXXX.. XX...... XX...XX. XX...XX. XX...XX. XX...XX.
XX.XXXX. .XXXX... .....XX. .....XX. .XX.XX.. XXXXXX.. XX...... X....XX. XX...XX. XX...XX.
XXXX.XX. ...XX... ...XXX.. ..XXXX.. XX..XX.. .....XX. XXXXXX.. ....XX.. .XXXXX.. .XXXXXX.
XXX..XX. ...XX... .XXX.... .....XX. XXXXXXX. .....XX. XX...XX. ...XX... XX...XX. .....XX.
XXX..XX. ...XX... XX...... XX...XX. ....XX.. XX...XX. XX...XX. ..XX.... XX...XX. XX...XX.
.XXXXX.. .XXXXXX. XXXXXXX. .XXXXX.. ...XXXX. .XXXXX.. .XXXXX.. ..XX.... .XXXXX.. .XXXXX..
........ ........ ........ ........ ........ ........ ........ ........ ........ ........
```

Формат входных данных

Входные данные состоят ровно из 12 строк, каждая из которых содержит ровно 60 символов: фотография окошка таймера. Поскольку окошко не совсем прямоугольной формы, небольшие уголки на фотографии всегда закрыты символами «-» (знак минус); точную форму уголков можно увидеть в примере. Все остальные символы соответствуют изображению внутреннего цилиндра и равны либо «.» (точка) для белых пикселов, либо «X» (большая буква икс) для чёрных пикселов.

Гарантируется, что таймер корректно показывает некоторое время, строго меньшее 60 минут и кратное 5 секундам.

Формат выходных данных

Выведите время, которое показывает таймер, на отдельной строке в формате «MM:SS». Здесь «MM» — две цифры, соответствующие минутам, а «SS» — две цифры, соответствующие секундам.

Пример

стандартный ввод
-------X.........XX.........XX.........XX.........------- ----..XX.........XX.........XX.........XX.........XX---- -...- X...XXXXXXX.. XX...XX....... XX...XXXXXX... XX...XX.. XX...XX.. XX...XX...XX.. -...XXXXX..- ----...---- -------...-------
стандартный вывод
07:05

Пояснение к примеру

В этом примере центр окошка находится между отметками 10 и 5. Отметка, обозначенная числом 5, хорошо видна справа. Часть нуля из числа 10 можно увидеть слева. Более тщательное изучение фотографии позволяет установить, что время, которое показывает таймер, в точности равно 7 минутам и 5 секундам.

Задача I. Ticket To Ride

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Олег и его друзья любят играть в настольную игру Ticket To Ride. В этой игре каждый игрок строит свою сеть железных дорог.

Внимание! Правила игры в этой задаче немного отличаются от стандартных правил. Рекомендуем прочитать условие внимательно, даже если вы знаете стандартные правила игры Ticket To Ride.

В качестве ресурсов для строительства используются цветные карты с вагончиками. В колоде 12 карт с вагончиками каждого из восьми цветов (пурпурный, белый, синий, жёлтый, оранжевый, чёрный, красный, зелёный) и еще 14 карт с локомотивами, которые можно использовать как карты произвольного цвета. Таким образом, всего в колоде 110 карт.

В начале игры колода перемешивается и кладётся на стол рубашкой вверх, а верхние пять карт открываются и выкладываются на стол в ряд рубашкой вниз.

В начале игры у каждого игрока в руке нет ни одной карты ресурсов. Игроки делают ходы по очереди. Каждым ходом игрок должен сделать одно из двух действий:

- взять несколько карт ресурсов, или
- построить новую дорогу, используя карты ресурсов из своей руки.

За один ход игрок может взять в руку поочерёдно **ровно две** карты ресурсов, или **только одну**, если он берёт открытую карту с локомотивом. Каждая из взятых карт может быть одной из пяти открытых карт или верхней картой колоды (в последнем случае другие игроки не увидят, какая именно карта была взята). Если была взята одна из пяти открытых карт, она немедленно заменяется следующей картой из колоды. При этом, если игрок берёт **открытую** карту **с локомотивом**, то это должна быть **единственная** карта ресурсов, взятая на этом ходу.

Как только колода заканчивается, все карты из сброса перемешиваются и становятся новой колодой. Если в какой-то момент среди пяти открытых карт на столе лежит хотя бы три карты с локомотивом, открытые карты немедленно сбрасываются, после чего открываются следующие пять верхних карт колоды.

За один ход игрок может построить ровно одну дорогу, используя некоторые или все карты из своей руки. Карты, использованные для постройки, немедленно отправляются в сброс и становятся известными всем игрокам. Каждая дорога в зависимости от типа и своих параметров требует определённого количества карт ресурсов определённых цветов. Дороги бывают трёх типов:

- обычная дорога одного из восьми цветов — требует для постройки $length$ карт цвета $color$;
- обычная дорога произвольного цвета — требует для постройки $length$ карт одного цвета (игрок сам выбирает цвет);
- паромная переправа — требует для постройки $length$ карт одного цвета (игрок сам выбирает цвет), при этом хотя бы L из них обязательно должны быть картами с локомотивом.

При постройке дороги игрок может использовать карты с локомотивом как карты с вагончиком произвольного цвета. Таким образом, для постройки обычной дороги синего цвета длины 3 игрок может использовать три синие карты с вагончиком, или одну синюю карту с вагончиком и две карты с локомотивом, или три карты с локомотивом. А для постройки обычной дороги произвольного цвета длины 5 игрок может использовать четыре красных карты с вагончиком и одну карту с локомотивом, но не может использовать три синих и две оранжевых карты.

В процессе игры бывает полезным понимать, может ли другой игрок в данный момент построить дорогу с конкретными параметрами. Естественно, никто из игроков не знает порядок карт в колоде и, как следствие, может не знать всех карт в руке другого игрока. Но, если внимательно следить за выходящими картами, то зачастую можно быть точно

уверенным, что у игрока попросту не хватит ресурсов для постройки такой дороги.

Олег решил написать приложение для игры в Ticket To Ride, в котором игрок может получать ответы на такие вопросы. Конечно, при ответе нужно учитывать, что знает игрок, задающий вопрос, а что от него скрыто. Вы поможете Олегу?

Формат входных данных

В этой задаче есть восемь основных цветов, для их описания используются слова `purple`, `blue`, `orange`, `white`, `green`, `yellow`, `black` и `red`. Для описания каждой карты, доставаемой из колоды, используется название одного из восьми цветов или слово `locomotive`.

В первой строке задано число n — число игроков и число m — количество следующих строк в файле с описанием игры ($2 \leqslant n \leqslant 5$, $2 \leqslant m \leqslant 10\,000$).

Далее следует строка, описывающая карты, выложенные на стол в начале игры. Эта строка имеет вид `open` <*five_cards*>$\{1,6\}$, где *five_cards* — это пять слов через пробел, описывающих пятёрку карт, выложенных на стол. Здесь и далее числа в фигурных скобках обозначают диапазон количества возможных повторений предыдущего элемента, в данном случае *five_cards*. В случае, если на столе оказывается не менее трёх карт с локомотивом, на стол выкладываются новые пять карт, и так далее. Гарантируется, что в тестах к этой задаче такая ситуация не может возникнуть более пяти раз подряд. Именно поэтому в описании команды `open` может быть 5, 10, 15, 20, 25 или 30 слов.

Далее по одному в строке идут описания действий и вопросов игроков.

- `take blind` <*card_color*> — игрок взял верхнюю карту из колоды взакрытую, и это была карта *card_color*;
- `take` <*i*> <*card_color*> <*five_cards*>$\{0,5\}$ — игрок взял i-ю открытую карту, ей на замену была открыта карта *card_color*; также в этой команде перечисляются не более пяти пятёрок карт на случай обновления открытых карт из-за появления среди них трёх карт с локомотивом;
- `build` <*color*> <*length*> <*card_numbers_list*> — игрок строит дорогу цвета *color* длины *length*, используя карты *card_numbers_list*;
- `build any` <*length*> <*card_numbers_list*> — игрок строит дорогу произвольного цвета длины *length*, используя карты *card_numbers_list*;
- `build ferry` <*length*> <*L*> <*card_numbers_list*> — игрок строит паромную переправу длины *length*, которая требует L карт с локомотивом, используя карты *card_numbers_list*;
- `?` <*player*> <*color*> <*length*> — может ли игрок с номером *player* построить дорогу цвета *color* длины *length*;
- `?` <*player*> `any` <*length*> — может ли игрок с номером *player* построить дорогу произвольного цвета длины *length*;
- `?` <*player*> `ferry` <*length*> <*L*> — может ли игрок с номером *player* построить паромную переправу длины *length*, которая требует L карт локомотивов.

Во всех командах параметры удовлетворяют следующим ограничениям: $1 \leqslant i \leqslant 5$, $1 \leqslant player \leqslant n$, $1 \leqslant length \leqslant 26$, $1 \leqslant L \leqslant 14$, $L \leqslant length$. Значение *color* совпадает с названием одного из восьми цветов. Список *card_numbers_list* представляет собой *length* чисел, записанных через пробел — номера использованных для постройки дороги карт в руке игрока. Номера карт перечислены в возрастающем порядке. Каждый игрок всегда держит карты в том порядке, в котором он брал их в руку. Карты в руке игрока нумеруются с единицы, начиная с карты, которая была взята раньше остальных.

Обратите внимание, что игроки ходят по очереди, поэтому в записи очередного действия не указан номер игрока, выполняющего это действие: эта информация была бы избыточной. Каждый вопрос задаётся игроком, который должен делать следующее действие. Очередь хода от вопросов не меняется.

Гарантируется, что описанная последовательность действий корректна, а игроки соблюдали все правила, описанные в этой задаче. Гарантируется, что каждая команда `build` содержит корректные данные для успешной постройки дороги.

Формат выходных данных

На каждый вопрос «?» выведите один из двух ответов: «No», если игрок точно не может построить такую дорогу, или «Maybe» в противном случае.

Пример

стандартный ввод
3 16 open locomotive blue locomotive locomotive blue blue red purple locomotive blue take blind blue take blind blue take 4 red take blind blue take 2 white ? 3 any 2 ? 3 blue 2 build blue 1 1 ? 1 blue 4 ? 1 purple 1 ? 2 ferry 1 1 ? 2 red 1 take blind yellow take blind orange ? 2 ferry 3 1
стандартный вывод
Maybe No No Maybe Maybe Maybe Maybe

Задача J. TTS Rights

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

TTS (Trivial Testing System) используется в СПбГУ для того, чтобы студенты могли решать контесты из дома. TTS использует следующую систему прав. Каждый объект (команда, задача, контест, условие, и т.д.) должен состоять ровно в одной группе. Группы организованы во взвешенный ориентированный граф с целыми весами рёбер. Права объекта a на объект b — это права группы, в которой состоит a, на группу, в которой состоит b. Права группы A на группу B могут быть получены следующим способом. Система рассматривает все непустые пути из A в B. Каждому пути соответствует число, которое можно вычислить как логическое `И` весов всех рёбер пути. Затем система вычисляет число R — логическое `ИЛИ` чисел, соответствующих путям из A в B. Это число определяет права A на B. Четыре младших бита — это права приоритета 0. Они располагаются так: бит 0 это `X` (права на запуск), бит 1 — `W` (права на запись), бит 2 — `R` (права на чтение), бит 3 — `A` (права на администрирование). Тем же образом биты 4–7 определяют права приоритета 1, 8–11 — приоритета 2, и так далее. Максимальный приоритет — седьмой. Окончательные же права определяются так: система делает цикл по всем приоритетам от 0-го до 7-го. При этом для чётных приоритетов система устанавливает включённые биты, а для нечётных — обнуляет включённые биты. Полученные 4 бита и есть окончательные права.

В графе есть две вершины, `initial` и `final`. Из вершины `initial` идут рёбра веса -1 во все остальные вершины. В вершину `final` входят рёбра веса -1 изо всех остальных вершин. Невозможно поменять рёбра **из** `initial` и **в** `final`.

Ваша задача — написать программу, которая поддерживает эффективное вычисление прав.

Формат входных данных

Все данные в TTS содержатся в так называемом `логе`. Лог — это файл из текстовых строк, каждая из которых содержит 4 или больше полей, разделённых символами табуляции (код ASCII 9). Первое поле — это всегда UNIX-время создания соответствующей записи. Второе поле определяет собственно запись. В третьем содержится описание компьютера, с которого сделана запись. Четвёртое — это идентификатор записи. Остальные поля, если они есть — это параметры записи.

В этой задаче лог будет содержать только записи о присвоении прав и о вычислении прав. Синтаксис записей следующий:

⟨`utime`⟩ `(+|-|=)GR` ⟨`ip`⟩ ⟨`src-node`⟩ ⟨`rights-mask`⟩ {⟨`dest-nodes`⟩}
⟨`utime`⟩ `(+|-|=)RG` ⟨`ip`⟩ ⟨`dest-node`⟩ ⟨`rights-mask`⟩ {⟨`src-nodes`⟩}
⟨`utime`⟩ `RIGHTS` ⟨`ip`⟩ ⟨`src-node`⟩ ⟨`dest-node`⟩

Эти команды действуют на все рёбра из ⟨`src-node`⟩ в ⟨`dest-nodes`⟩ или на все рёбра из ⟨`src-nodes`⟩ в ⟨`dest-node`⟩. Операции с префиксом '`+`' добавляют указанные права к ребру (то есть выполняет операцию логического `ИЛИ`), с префиксом '`-`' убирают права (то есть обнуляют все биты, которые содержит ⟨`rights-mask`⟩), а с префиксом '`=`' заменяют числа на соответствующих рёбрах.

Если маска указана как '`-`', то соответствующее число считается равным -1.

⟨`rights-mask`⟩ — это десятичное или шестнадцатеричное число. В последнем случае число начинается с префикса `0x`. Числа рассматриваются по модулю 2^{32} и не могут по абсолютной величине превосходит $2^{32} - 1$. Таким образом, каждое число, отличное от 0, имеет два различных представления, отличающихся на 2^{32}. Например, -1 = `0xffffffff` = 4294967295. Знак минус может находиться только перед числом, например `-0x25`.

Формат выходных данных

Для каждой команды RIGHTS выведите сообщения о правах ⟨src-node⟩ на ⟨dest-node⟩. Следуйте формату вывода в примере. Права должны выводиться от старших битов к младшим. Размер лога будет не больше 64 килобайт.

Примеры

стандартный ввод

```
1096000000      =GR     195.19.228.2    admin   0xefffffff      initial
1096000000      RIGHTS  195.19.228.2    admin   initial
1096000000      =GR     195.19.228.2    teams   -       users
1096000000      RIGHTS  195.19.228.2    teams   users
1096000000      +GR     195.19.228.2    users   -       nobody
1096000000      RIGHTS  195.19.228.2    nobody  teams
1096000000      RIGHTS  195.19.228.2    admin   nobody
1096000000      +RG     195.19.228.2    univ    -       univ1   univ2   univ3
1096000000      =RG     195.19.228.2    auniv   -       univ    iuniv
1096000000      +GR     195.19.228.2    auniv   -       teams
1096000000      +GR     195.19.228.2    contests        -       ucontests       pcontests
1096000000      +GR     195.19.228.2    ucontests       -       trains  olymps
1096000000      =GR     195.19.228.2    jury    0xeff   jmonitors
1096000000      =GR     195.19.228.2    final   0x40    jmonitors
1096000000      =GR     195.19.228.2    univ    5       ucontests
1096000000      RIGHTS  195.19.228.2    univ1   trains
1096000000      =GR     195.19.228.2    iuniv   5       trains
1096000000      =GR     195.19.228.2    auniv   5       uproblems
1096000000      =GR     195.19.228.2    nobody  5       pproblems       public
1096000000      =GR     195.19.228.2    teams   4       statements
1096000000      RIGHTS  195.19.228.2    univ3   statements
1096000000      =GR     195.19.228.2    final   0x70    hidden
```

стандартный вывод

```
admin has ARW rights on initial.
teams has no rights on users.
nobody has no rights on teams.
admin has ARW rights on nobody.
univ1 has RX rights on trains.
univ3 has R rights on statements.
```

стандартный ввод

```
1096000000      =GR     195.19.228.2    admin   0xefffffff      initial
1096000000      =GR     195.19.228.2    admin   0xAFFFFFFF      disableadmin
1096000000      RIGHTS  195.19.228.2    admin   disableadmin
1096000000      =GR     195.19.228.2    kittens -       animals
1096000000      =RG     195.19.228.2    animals -       dogs
1096000000      =RG     195.19.228.2    kittens 4       animals
1096000000      =GR     195.19.228.2    animals 4       dogs
1096000000      =GR     195.19.228.2    animals 4       mice
1096000000      =GR     195.19.228.2    kittens 0x3000000       mice
1096000000      RIGHTS  195.19.228.2    kittens mice
1096000000      =GR     195.19.228.2    dogs    0x40    mice
1096000000      RIGHTS  195.19.228.2    dogs    mice
1096000000      -GR     195.19.228.2    animals 0x3f    kittens
1096000000      RIGHTS  195.19.228.2    animals kittens
```

стандартный вывод

```
admin has RX rights on disableadmin.
kittens has RWX rights on mice.
dogs has no rights on mice.
animals has no rights on kittens.
```