За правильное решение каждой задачи даётся 50 баллов, за неправильное (даже частичное) — 0 баллов.

По каждой задаче можно сделать не более трёх попыток.

Решения проверяются сразу после отсылки; результат сообщается участнику.

Задача A. $A + (-B)$

Имя входного файла:	`aplusminusb.in`
Имя выходного файла:	`aplusminusb.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	16 мегабайт

Формат входного файла

Во входном файле заданы два целых неотрицательных числа A и B (A, $B \leqslant 10^{10\,000}$), каждое на отдельной строке.

Формат выходного файла

В выходной файл выведите одно число, равное разности A и B.

Пример

aplusminusb.in	aplusminusb.out
5 3	2

Задача B. Ботвинник и Фишер

Имя входного файла:	`chess.in`
Имя выходного файла:	`chess.out`
Ограничение по времени:	15 секунд
Ограничение по памяти:	128 мегабайт

В неофициальном матче по шахматам встречаются шестой чемпион мира Михаил Ботвинник и одиннадцатый чемпион Роберт Фишер. Идёт сотая игра матча. Сделано уже больше сотни ходов. После десятого перерыва в партии Михаил и Роберт решили закончить партию максимально быстро, благо их не очень волновал её результат. К счастью, на доске осталось всего 4 фигуры. У каждого из гроссмейстеров осталось по одной тяжёлой фигуре (ладья или ферзь) и, естественно, королю. Игра считается законченной, когда один из королей заматован, или у каждой из сторон остался только король. В первом случае игрок, которого заматовали, проигрывает, а во втором объявляется ничья. **Все другие правила** окончания игры (в том числе правило троекратного повторения позиции и правило пятидесяти ходов) в этой партии **отменены**.

Ваша задача — найти самый быстрый способ закончить игру.

Формат входного файла

На первой строке находится описание позиции белых фигур, на второй — чёрных. Каждая фигура определяется тремя символами. Первый — это тип фигуры: `K` — это король, `Q` — ферзь, а `R` — ладья. Второй и третий символ содержат описание клетки, на которой стоит данная фигура, буква соответствует номеру вертикали, а цифра — горизонтали. Например, `Ka1` обозначает, что король стоит в левой нижней клетке доски. Описания фигур внутри строки разделяются пробелом. Гарантируется, что позиция корректна. Право хода в данной позиции принадлежит чёрным. Первой фигурой в каждой строке является король.

Формат выходного файла

Выведите кратайшую последовательность ходов, приводящую к концу игры. каждый ход записывается шестью символами: первый это название фигуры, следующие два это клетка, на которой фигура стояла до хода, четвёртый это дефис (‘-’), а пятый и шестой — клетка, в которую фигура походила. Если оптимальных решений несколько разрешается вывести любое.

Примеры

chess.in	chess.out
`Ka1 Qb5` `Ka3 Qa8`	`Qa8-a4` `Qb5-b2`

`Ka1 Qf5`
`Ka3 Rh8`
	`Rh8-f8`
	`Qf5-f1`
	`Rf8-f1`

Задача C. Деление уголком

Имя входного файла:	`division.in`
Имя выходного файла:	`division.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

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 входного файла

В двух строках входного файла заданы делимое и делитель, меньшие 10^{100}.

Формат выходного файла

В выходной файл необходимо вывести 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.

Пример

division.in	division.out
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. Мультиплексор

Имя входного файла:	`multiplexor.in`
Имя выходного файла:	`multiplexor.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

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 входного файла

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 выходного файла

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

multiplexor.in

```
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
```

multiplexor.out

```
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
```