Задача 1. Графический интерфейс

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Компания, в которой вы работаете, занимается созданием нового оконного графического интерфейса. Это обусловлено тем, что все существующие реализации такого интерфейса обладают одним фатальным недостатком: их сделал кто-то другой. Окна, в которых отображаются данные, представляют собой прямоугольники, состоящие из пикселей. Стороны окон параллельны сторонам экрана. Вам поручена задача: написать модуль, который определяет взаимное расположение окон.

Согласно техническому заданию, модуль должен получать на вход положение двух окон A и B, и выдавать вердикт о том, как они расположены относительно друг друга. Возможны следующие вердикты взаимного расположения:

- `A in B` — все пиксели окна A являются пикселями окна B;
- `B in A` — все пиксели окна B являются пикселями окна A;
- `Separate` — никакой пиксель не принадлежит окнам A и B одновременно;
- `Intersect` — во всех остальных случаях.

Положение окна задаётся координатами двух пикселей: в левом верхнем его углу и в нижнем правом. Пиксель с координатами (0, 0) находится в левом верхнем углу экрана. Координаты возрастают при движении вправо и вниз.

Гарантируется, что положения окон A и B не совпадают.

Формат входных данных

В первой строке входного файла содержится одно целое число N — количество наборов входных данных для вашего модуля ($1 \leqslant N \leqslant 1000$). Затем следуют N пар строк.

Каждый набор входных данных описывается в двух строках: в первой из них указано положение окна A, а во второй — положение окна B. Для каждого окна это четыре целых числа: координаты X_l, Y_t левого верхнего и координаты X_r, Y_b правого нижнего пикселя ($0 \leqslant X_l \leqslant X_r \leqslant 10^5$, $0 \leqslant Y_t \leqslant Y_b \leqslant 10^5$).

Формат выходных данных

Для каждого набора входных данных в выходной файл нужно вывести в отдельную строку вердикт взаимного расположения окон.

Пример

input.txt	output.txt
3 0 0 3 3 1 1 2 2 0 0 9 9 10 0 19 9 1 0 3 2 0 1 2 3	B in A Separate Intersect

Иллюстрация

Задача 2. Поиск на кубе

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Два друга играют в придуманную настольную игру: поиск клада на кубе. Сначала ведущий загадывает размер куба, а также положение клада и начальное положение робота на этом кубе. Затем игрок пытается найти клад по принципу «теплее / холоднее», вслепую управляя роботом.

Игровое поле представляет собой куб размера $N \times N \times N$, каждая грань которого разбита на $N \times N$ клеток. Согласно правилам игры, ведущий может установить размер куба N в диапазоне от 3 до 300 включительно. В одной из $6N^2$ клеток на поверхности куба ведущий размещает клад, который остаётся в этой клетке в течение всей игры. Также изначально в одну из клеток ведущий ставит робота, повернув его в одном из четырёх направлений так, чтобы он смотрел параллельно одной из сторон клетки.

Игра разбивается на шаги, на каждом из которых игрок даёт роботу одну команду. Есть четыре типа команд:

- `left` — робот поворачивается на 90 градусов влево, оставаясь в той же клетке;
- `right` — робот поворачивается на 90 градусов вправо, оставаясь в той же клетке;
- `forward` — робот проезжает вперёд одну клетку;
- `dig` — робот копает клад в той клетке, в которой он находится.

Направление поворота влево/вправо определяются, если смотреть на робота извне куба.

При выполнении команды `forward` сначала определяется, на какую их четырёх сторон своей клетки сейчас смотрит робот. Затем робот перемещается в соседнюю по этой стороне клетку. Если при этом робот остаётся на той же грани куба, то его направление не изменяется. Если же он переезжает на другую грань куба, то вектор направления естественным образом поворачивается на 90 градусов вокруг ребра куба. При этом всегда действует простое правило возврата: если после выполнения команды `forward` повернуть робота на 180 градусов и снова выполнить команду `forward`, то робот вернётся обратно в предыдущую клетку.

Игрок не видит игрового поля: он вслепую говорит команды ведущему, а ведущий их выполняет. После выполнения каждой команды `forward` ведущий говорит игроку одно из трёх слов:

- `closer` — робот оказался ближе к кладу;
- `farther` — робот оказался дальше от клада;
- `same` — расстояние от робота до клада не изменилось.

Расстояние от робота до клада — это евклидово расстояние между центрами клеток, в которых они находятся, то есть длина соединяющего эти центры напрямую отрезка. Обратите внимание, что расстояние считается насквозь куба, а не вдоль его поверхности. Если эта величина уменьшилась, то ведущий говорит `closer`, а если увеличилась — то `farther`.

Вам предлагается написать программу, которая будет играть в эту игру за игрока. У вас нет права на ошибку: при выполнении команды `dig` робот должен обязательно стоять в клетке с кладом.

Протокол взаимодействия

Это интерактивная задача, и в ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осу-

ществляется через стандартные потоки ввода-вывода.

Ваша программа выводит на поток вывода команды управления роботом. Каждая команда задаётся словом `left`, `right`, `forward` или `dig` (см. описание команд выше). После каждой выведенной вашей программой команды `forward` на поток ввода приходит одно из слов: `closer`, `farther` или `same` (см. описание результатов выше).

После вывода команды `dig` игра заканчивается. Если при этом робот и клад находятся в разных клетках, то ваше решение получает вердикт Wrong Answer. Количество команд не должно превышать $100N$, иначе решение получит вердикт Wrong Answer.

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждой выведенной команды. Иначе решение может получить вердикт Timeout.

Пример

stdin	stdout
farther closer closer farther closer	forward left left forward forward forward right right forward dig

Задача 3. Зеркала

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

Сбылась мечта идиота! Отныне ему не надо улепетывать от разъяренной вдовушки по коридору. Теперь мадам Грицацуева — главный персонаж игры «Отрази меня».

Согласно правилам этой игры, она движется по прямоугольному клеточному полю размера $N \times M$, постоянно отражаясь то от границ поля, то от двусторонних зеркал, которые либо вращаются, либо неподвижны. Каждое зеркало представляет собой отрезок единичной длины, центр которого находится в центре клетки. Сама мадам намного меньше и зеркала, и единичной клетки, поэтому её можно считать точкой, которая в любой целочисленный момент времени находится в центре какой-нибудь клетки.

За секунду мадам обязательно переходит в соседнюю клетку, если только она не отражается от границ поля. При этом, если она попадает в клетку, где находится зеркало, то она меняет направление движения по правилу «угол падения равен углу отражения». Более точно, если зеркало было наклонено под 45 градусов к её траектории, то мадам отразится на 90 градусов, а если зеркало перегораживало ей путь, то есть было перпендикулярно её движению, то она поменяет направление на 180 градусов. Если же зеркало не мешало её движению, то есть было параллельно направлению движения, то оно никак не помешает дальнейшему движению мадам. Если же вдовушка попадает в граничную клетку поля, то направление ее движения меняется на противоположное. При этом в этой граничной клетке мадам находится два соседних момента времени: можно считать, что полсекунды вдова движется от центра клетки до границы, там моментально отражается и ещё полсекунды возвращается обратно.

Мадам Грицацуева может стартовать в любой неотрицательный момент времени в произвольном направлении в одной из допустимых для старта точек. Её цель — как можно раньше оказаться в заветной точке с координатами (r_0, c_0).

Формат входных данных

В первой строке входного файла записано шесть целых чисел N, M, K, S, r_0 и c_0, где N, M — размеры поля по вертикали и горизонтали ($1 \leqslant N, M \leqslant 10^9$), K — количество возможных мест старта ($1 \leqslant K \leqslant 10^3$), S — количество зеркал ($0 \leqslant S \leqslant 10^3$), r_0, c_0 — номер строки и столбца, в которой находится точка выхода ($1 \leqslant r_0 \leqslant N$, $1 \leqslant c_0 \leqslant M$).

Далее в K строках приводятся координаты допустимых стартовых точек — в каждой строке по два целых числа r_j и c_j ($1 \leqslant r_j \leqslant N$, $1 \leqslant c_j \leqslant M$).

В остальных S строках приводятся описания зеркал: по пять целых чисел r_i, c_i, a_i, t_i, p_i в каждой. Здесь r_i, c_i — номера строки и столбца клетки, в которой расположено i-ое зеркало ($1 \leqslant r_i \leqslant N$, $1 \leqslant c_i \leqslant M$).

Число a_i описывает положение зеркала в начальный момент времени ($0 \leqslant a_i \leqslant 3$). Если $a_i = 0$, то зеркало расположено вертикально, а с каждым увеличением значения a_i на 1 оно поворачивается на 45 градусов по часовой стрелке. Таким образом, если $a_i = 1$, то отрезок зеркала направлен из левого нижнего угла в правый верхний, если $a_i = 2$, то оно горизонтально и т.д.

Значения t_i и p_i задают соответственно время первого поворота зеркала на 45 градусов по часовой стрелке и период, за который оно совершает каждый последующий такой поворот ($1 \leqslant p_i \leqslant 7$, $1 \leqslant t_i \leqslant p_i$). Таким образом, зеркало поворачивается в моменты времени: t_i —

первый поворот, $t_i + p_i$ — второй поворот, $t_i + 2p_i$ — третий поворот и т.д. Заметим, что в моменты времени t_i, $(t_i + p_i)$ зеркало уже находится в новом положении, а в моменты времени $(t_i - 1)$, $(t_i + p_i - 1)$ — ещё в предыдущем. Если же зеркало неподвижно, то два последних числа t_i и p_i в описании равны -1.

Гарантируется, что все позиции стартовых точек, зеркал и точки выхода попарно различны, в том числе, что ни одно зеркало не находится в стартовой точке. Ориентация поля и нумерация строк и столбцов таковы, что при движении вниз увеличивается номер строки, а при движении вправо увеличивается номер столбца.

Формат выходных данных

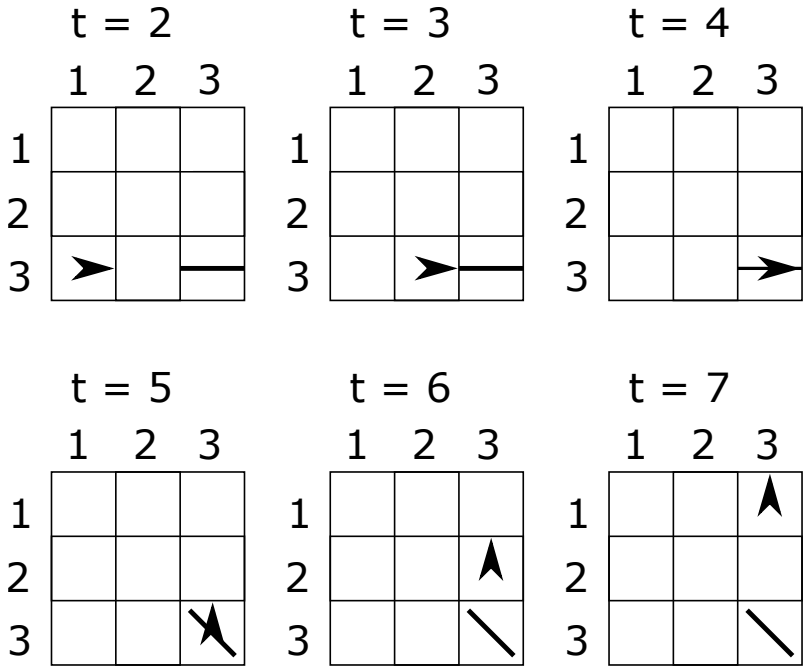
В выходной файл требуется вывести одно целое число: самое раннее время, когда мадам может оказаться в точке r_0, c_0, или -1, если это невозможно.

Пример

input.txt	output.txt
3 3 1 1 1 3 3 1 3 3 2 5 7	7
3 3 1 2 3 2 1 1 1 3 2 5 7 3 3 2 -1 -1	-1

Пояснение к примеру

На рисунке ниже изображён первый тест из условия. Стрелка соответствует положению и направлению движения мадам, отрезок в клетке $(3, 3)$ обозначает зеркало. Мадам стартует в момент времени $t = 2$ вправо. В момент времени $t = 4$ она пролетает мимо зеркала и в момент времени $t = 4.5$ отражается от стены, после чего отражается от повернувшегося зеркала в момент времени $t = 5$. В момент времени $t = 7$ мадам достигает точки выхода.



Задача 4. Дороги к синематографу

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

После фиаско в Санта-Каролине, мистер Фёст и его друзья открыли новый храм синематографа. Новое место популяризации синематографа выбрали неподалёку от большой железной дороги. Чёрный Джек предложил построить дороги до близких станций, на которых останавливаются пассажирские поезда.

Мистер Фёст нарисовал на плане систему координат. Началом системы координат выбрано то место, в котором расположились миссионеры синематографа, а оси X и Y направлены на восток и север соответственно. Станции отмечены на плане точками с координатами (x_i, y_i) при $i = 1 \ldots n$.

Железная дорога идёт в направлении на юго-восток, поэтому выполняются неравенства:

$$\begin{cases} 0 \leqslant x_1 \leqslant x_2 \leqslant x_3 \leqslant \ldots \leqslant x_{n-1} \leqslant x_n \\ 0 \leqslant y_n \leqslant y_{n-1} \leqslant y_{n-2} \leqslant \ldots \leqslant y_2 \leqslant y_1 \end{cases}$$

Диана настаивает на том, чтобы каждая дорога шла в точности в какую-нибудь сторону света, то есть чтобы соответствующий ей отрезок на плане был параллелен какой-нибудь оси координат. Джек же говорит, что дороги нужно построить таким образом, чтобы можно было доехать по дорогам от каждой станции до синематографа, не используя железную дорогу и проехав при этом минимально возможное расстояние. Разумеется, расстояние должно быть минимальным при условии параллельности дорог осям координат.

Дороги стоят денег, поэтому мистер Фёст хочет, чтобы суммарная длина дорог была минимальна при условии выполнения требований Дианы и Джека. А сэкономленные деньги можно потратить на покупку новых кинолент.

Формат входных данных

В первой строке входного файла задано целое число n — количество станций ($1 \leqslant n \leqslant 500$). В каждой из оставшихся n строк записано по два целых числа x_i, y_i — координаты одной из станций ($0 \leqslant x_i, y_i \leqslant 10^6$).

Гарантируется, что последовательность (x_i) нестрого возрастает, а последовательность (y_i) — нестрого убывает. Никакие две станции не находятся в одной точке. Никакая станция не расположена в начале координат.

Формат выходных данных

В первую строку выходного файла нужно вывести два целых числа: k — сколько дорог предлагается построить ($1 \leqslant k \leqslant 2\,000$) и A — суммарную длину всех этих дорог. В каждую из следующих k строк требуется выдать описание одной дороги.

Описание дороги должно содержать четыре целых числа x_1, y_1, x_2, y_2 — координаты концевых точек дороги ($x_1 \geqslant x_2$, $y_1 \geqslant y_2$). При этом должно быть верно ровно одно из двух утверждений: либо $x_1 = x_2$, либо $y_1 = y_2$.

Гарантируется, что существует оптимальный план, удовлетворяющий ограничениям формата выходных данных. Если возможно несколько вариантов ответа, выведите любой из них.

Пример

input.txt	output.txt
3 0 9 2 4 5 1	4 16 0 9 0 4 2 4 0 4 5 1 0 1 0 4 0 0

Задача 5. Геометрический решатель

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Мальчик Кедас участвует в разработке нового геометрического решателя. Ему поручили реализовать устранение избыточных ограничений на двумерном чертеже. Помогите Кедасу справиться с задачей.

В рамках данной задачи будем полагать, что на чертеже есть только N отрезков $L_1, L_2, \ldots, L_n$ и набор связывающих их ограничений. Каждое *ограничение* — это утверждение об объектах чертежа. Ограничения бывают четырёх типов:

- `hor` i — отрезок L_i горизонтальный (Y-координаты его концов совпадают);
- `vert` i — отрезок L_i вертикальный (X-координаты его концов совпадают);
- `par` i j — отрезки L_i и L_j параллельны;
- `perp` i j — отрезки L_i и L_j перпендикулярны.

Решением чертежа называется размещение всех $2N$ концов всех отрезков чертежа на координатной плоскости, при котором:

1. все ограничения чертежа удовлетворены, т.е. соответствующие утверждения верны;
2. каждый отрезок имеет ненулевую длину, то есть его концевые точки **не** совпадают.

Считается, что направление отрезка значения не имеет.

Избыточные ограничения могут плохо повлиять на работу численных алгоритмов геометрического решателя, поэтому их по возможности стараются удалить. Подмножество ограничений чертежа называется *избыточным*, если при его удалении множество решений чертежа не изменяется. Если множество решений чертежа пусто, то чертёж называется *несовместным*, а в противном случае — *совместным*.

В данной задаче требуется определить, является ли заданный чертёж совместным, и если он является, то также найти на нём избыточное подмножество ограничений максимального размера.

Формат входных данных

В первой строке входного файла записано два целых числа n и m, где n — количество отрезков на чертеже ($1 \leqslant n \leqslant 10^5$), а m — количество ограничений на чертеже ($0 \leqslant m \leqslant 10^5$). В каждой из следующих m строк описывается одно ограничение.

Каждое ограничение описывается так, как показано выше в условии: сначала записано слово (`hor`, `vert`, `par` или `perp`), определяющее тип, потом через пробел записан номер i первого отрезка-аргумента. Если тип равен `par` или `perp`, то ещё через пробел указан номер j второго отрезка-аргумента. При этом $1 \leqslant i \neq j \leqslant n$.

Отрезки пронумерованы числами от 1 до n, ограничения пронумерованы числами от 1 до m в порядке описания. Ограничения могут иметь полностью одинаковые описания.

Формат выходных данных

В первую строку требуется вывести слово `consistent`, если чертёж совместный, и `inconsistent`, если чертёж несовместный. Если чёртеж несовместный, то больше ничего выводить **не** нужно.

Если чертёж совместный, то во вторую строку нужно вывести целое число k — количество ограничений в найденном избыточном множестве ($0 \leqslant k \leqslant m$). В третью строку нужно вывести k различных целых чисел — номера ограничений в этом множестве в любом порядке.

Если искомых ответов несколько, разрешается вывести любой из них.

Примеры

input.txt	output.txt
2 4 vert 1 hor 2 vert 1 perp 2 1	consistent 2 4 1
2 4 vert 1 hor 2 vert 1 par 2 1	inconsistent

Пояснение к примеру

В первом примере на отрезок L_1 наложено две вертикальности, очевидно одну из них можно убрать, и множество решений не изменится. Перпендикулярность отрезков L_1 и L_2 тоже лишняя, так как один отрезок вертикальный, а другой горизонтальный. Если удалить три ограничения, то всегда можно привести решение, на котором один из отрезков **не** параллелен осям координат, то есть множество решений чертежа увеличится.

Во втором примере чертёж несовместный, поскольку ограничение 4 говорит, что отрезки L_1 и L_2 параллельны, тогда как из ограничений 1 и 2 следует, что они перпендикулярны.

Задача 6. Монстры

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Вася обожает играть в компьютерные игры. Ещё в начальной школе он мечтал о том, чтобы связать свою карьеру с разработкой игр. К сожалению, он никогда не умел программировать. И проектировать 3D-модели он тоже не научился. Вася всегда был уверен, что совсем не обязательно иметь какие-то навыки, главное — по-настоящему любить своё дело. Поэтому он решил стать тестировщиком игр — ведь играть-то он страсть как любил!

Его мечта сбылась. Теперь он сидит в кубической комнате два-на-два-на-два метра и в триста семьдесят пятый раз проходит одну и ту же глупейшую браузерную игру. В этой игре Вася управляет персонажем, который бегает по уровням и сражается с N монстрами. Вася успешно завершает игру, только когда он побеждает всех этих монстров. Уровни спроектированы таким образом, что Вася может драться только с одним монстром за раз, причём порядок сражений с монстрами строго зафиксирован разработчиками. Вероятно, так разработчики представляли себе понятие «нелинейное прохождение».

Когда Вася встречается с монстром, он достаёт свой меч и начинает бить им монстра. А в это время монстр начинает есть Васю. Никаких навыков, уклонений, защит и прочего в игре нет, потому что это было бы слишком сложно для целевой аудитории. После минуты ритмичных постукиваний и пожёвываний поединок заканчивается, причём его исход определяется исключительно двумя числовыми параметрами: крутизной Васиного меча и размером зубов монстра. Если размер зубов строго больше крутизны меча, то монстр съедает Васю, и, как следствие, Вася проигрывает. В противном случае Вася побеждает монстра, и тогда из монстра выпадает новый меч, как бы в качестве награды за страдания. Вася может этот меч взять вместо своего текущего, а может просто проигнорировать его. Учитывая то, что для каждого меча известна его крутизна в числовом виде, становится совершенно очевидно, когда нужно брать новый меч, а когда нет . После этого Вася полностью излечивается от своих ран и топает по уровням к следующему монстру.

Причина столь многократного прохождения столь никчёмной игры заключается в том, что коллеги Васи не могут придумать более умного способа настроить баланс игры. Хотя, возможно, им просто нравится мучить Васю — в Васиной голове эта версия становится всё более и более убедительной.

Изначально у Васи есть меч, крутизна которого равна целому числу X, заданному в конфиг-файле игры. Крутизна меча, который выпадает из i-ого монстра, задаётся целым числом S_i в конфиг-файле. Проблема в том, что размер зубов каждого монстра **не** зафиксирован. В конфиг-файле указывается только целое число M_i — максимально возможное значение для размера зубов i-ого монстра. Конкретный размер его зубов генерируется игрой в самом начале случайным образом в момент создания нового персонажа и может принять любое целое значение в диапазоне от 1 до M_i включительно.

Судя по всему, случайную генерацию сделали только для того, чтобы можно было говорить о высокой «реиграбельности» игры в маркетинговых материалах. Не помогает тестированию и тот факт, что периодически кто-то меняет числа в конфиг-файле, пытаясь подкрутить баланс. Хоть Вася и не умеет программировать, он смутно чувствует, что процесс его страданий можно автоматизировать. Он просит Вас помочь ему: написать программу, которая будет оценивать вероятность успешного завершения игры по конфиг-файлу.

В данной задаче нужно определить, сколькими способами можно сгенерировать монстрам

размеры зубов так, чтобы Вася успешно завершил игру. Поскольку количество таких способов может быть очень велико, требуется найти его остаток от деления по модулю $(10^9 + 7)$. Кроме того, в конфиг-файле задаются по очереди K изменений, и Вам нужно определять ответ заново после каждого изменения. Каждое изменение либо увеличивает крутизну S_i одного из мечей, либо увеличивает M_i — ограничение сверху на размер зубов одного из монстров. Разумеется, Вася достаточно умён, и потому он играет оптимально.

Формат входных данных

В первой строке входного файла записано три целых числа N, K и X, где N — количество монстров в игре $(1 \leqslant N \leqslant 10^5)$, K — количество изменений конфиг-файла игры $(0 \leqslant K \leqslant 10^5)$ и X — крутизна начального меча Васи $(1 \leqslant X \leqslant 10^9)$.

Во второй строке дано N целых чисел M_i, каждое из которых определяет максимально возможный размер зубов одного из монстров $(1 \leqslant M_i \leqslant 10^9)$. В третьей строке дано N целых чисел S_i — крутизны мечей, выпадающих из монстров $(1 \leqslant S_i \leqslant 10^9)$.

Далее во входном файле записано K строк, в каждой из которых описано одно изменение конфиг-файла. Для каждого изменения записано три целых числа: t_j — тип изменения $(0 \leqslant t_j \leqslant 1)$, k_j — номер монстра, подвергшегося изменению $(1 \leqslant k_j \leqslant N)$, и V_j — новое значение параметра $(2 \leqslant V_j \leqslant 10^9)$. Если $t_j = 0$, то у k_j-ого монстра максимальный размер зубов M_{k_j} увеличивается до V_j. Если $t_j = 1$, то крутизна S_{k_j} меча, выпадающего из k_j-ого монстра, увеличивается до V_j.

Гарантируется, что при каждом изменении новое значение параметра больше старого.

Формат выходных данных

В выходной файл требуется вывести $K+1$ целое число, каждое число в отдельной строке. Первое число — ответ на задачу с исходным состоянием конфиг-файла игры. $(k+1)$-ое число должно равняться ответу на задачу после применения к конфиг-файлу первых k изменений.

Примеры

input.txt	output.txt
5 3 2 4 2 3 5 7 3 2 4 3 2 1 3 5 0 2 3 0 2 10	192 300 450 450
6 0 1 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000	993000007

Пояснение к примеру

В первом примере до изменений игра проходит следующим образом. С первым монстром Вася встречается, имея начальный меч крутизны $X = 2$. Размер зубов монстра должен быть 1 или 2, чтобы его можно было победить. После победы Васе выпадает меч крутизны 3, который он берёт взамен начального. Второй монстр имеет размер зубов 1 или 2, в любом случае Вася его побеждает. При этом выпадает меч крутизны 2, который Вася **не** берёт. Размер зубов третьего монстра не превышает 3, и в любом случае Вася его побеждает, забирая выпавший из него меч крутизны 4. Последние два монстра побеждаемы, только если размер их зубов лежит в диапазоне от 1 до 4, и с них падают слабые мечи, брать которые нет смысла. Получается $2 \times 2 \times 3 \times 4 \times 4 = 192$ варианта генерации проходимой игры.

Задача 7. Регулярные выражения

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	10 секунд
Ограничение по памяти:	256 мегабайт

Павел не учился в университете, он — программист от природы. Больше всего на свете он любит лаконичные куски кода, преимущественно однострочные. В его стремлении сделать код короче часто помогают регулярные выражения. Нет ничего более творческого и увлекательного, чем составлять очередное регулярное выражение глубокой ночью!

Всё бы ничего, но на код Павла часто жалуется его коллега Семён. Сам Семён никогда не использует регулярные выражения в своём коде. Да Семён просто завидует настоящему мастеру! Если бы он умел составлять регулярные выражения так же виртуозно, как Павел, он точно не стал бы жаловаться.

Однажды Павел неосторожно заметил, что Семён недостаточно творчески развит для составления сложных регулярных выражений. На что Семён кощунственно ответил, что в составлении регулярных выражений нет ничего творческого. Жаркий спор перерос в соревновение: коллеги поспорили, кто быстрее сможет составить регулярное выражение, под которое подходит заданный набор строк. Семён хочет разбить оппонента в пух и прах, составив программу, которая будет решать эту задачу автоматически, тем самым доказав отсутствие творческой составляющей в процессе. Судить спор будет общий знакомый спорщиков — биоинформатик Петя.

Семён посмотрел синтаксис перловых регулярных выражений и ужаснулся: во что программисты превратили простое и красивое понятие? Семён закончил мехмат, и для него регулярное выражение всегда определялось так:

1. `0` (символ нуля) — регулярное выражение, под которое не подходит никакая строка.
2. `a`, `g`, `t`, `c` (буква) — регулярное выражение, под которое подходит ровно одна строка: однобуквенная строка с той же буквой, что и в выражении.
3. Если `R` и `P` — регулярные выражения, то `(R|P)` — тоже регулярное выражение. Под него подходят все строки, которые подходят хотя бы под одно из выражений `R` и `P`.
4. Если `R` и `P` — регулярные выражения, то `(RP)` — тоже регулярное выражение. Под него подходят все строки, которые которые можно разбить на две части так, что первая часть подходит выражению `R`, а вторая — выражению `P`.
5. Если `R` — регулярное выражение, то `(R*)` — тоже регулярное выражение. Под него подходят все строки, которые можно разбить на $k \geqslant 0$ частей так, что каждая часть подходит выражению `R`.

Например, под регулярное выражение `(a*)` подходит любая строка, состоящая только из букв `a`, в том числе и пустая. Под регулярное выражение `(0*)` подходит только пустая строка (её можно разбить на ноль подходящих под выражение `0` частей). Под регулярное выражение `(a|(g(tc)))` подходит две строки: `a` и `gtc`. Обратите внимание, что в регулярном выражении запрещается опускать скобки или добавлять лишние: должны стоять ровно те скобки, которые получаются при построении выражения по указанным выше пунктам.

Семён хочет выиграть соревнование идеально, построив самое короткое подходящее регулярное выражение. Помогите Семёну составить программу, которая по набору строк находит любое регулярное выражение минимальной длины, под которое все эти строки подходят.

Формат входных данных

В первой строке записано одно целое число T — количество тестов. Далее описаны тесты.

В первой строке описания теста записано целое положительное число N — количество строк в наборе. Каждая из последующих строк начинается с целого неотрицательного числа L — длины строки из набора, после которого записана сама строка через пробел. Строка состоит только из маленьких латинских букв `a`, `g`, `t`, `c`. Обратите внимание, что строка в наборе может быть пустой, в таком случае в строке файла записаны только цифра 0 и пробел.

Суммарное количество строк во всех наборах не превышает 2 000. Сумма длин всех строк в наборах не превышает 2 000.

Формат выходных данных

Требуется вывести ровно T регулярных выражений, по одному в строке. Каждое регулярное выражение должно быть ответом для соответствующего теста. Если для какого-то теста не существует регулярного выражения, под которое подходят все строки из набора, требуется вывести слово `Impossible` вместо ответа.

Пример

input.txt	output.txt
3 2 1 a 3 gtc 1 0 3 1 g 2 gg 3 ggg	(a|(g(tc))) (0*) (g*)

Комментарий

В шестой строке входных данных примера после нуля стоит один символ пробела.

Задача 8. Команда на ВСО 2017

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	6 секунд
	12 секунд (для Java)
Ограничение по памяти:	256 мегабайт

Вы слышали что-нибудь о самом важном спортивном событии года в Сибири — «Всесибирской Спортивной Олимпиаде (ВСО)»? В этому году Паша планирует собрать футбольную команду для участия в этой олимпиаде. У Паши есть N знакомых, пронумерованных целыми числами от 1 до N, из которых он планирует набрать игроков в команду. Для i-го из них Паша без труда определил два значения: a_i — оценка его умения играть в атаке и d_i — оценка его умения играть в защите.

Чтобы определиться с итоговым составом команды на «ВСО 2017», Паша планирует провести M тренировочных сборов. Каждые сборы описываются парой целых чисел l_j и r_j ($1 \leqslant l_j < r_j \leqslant N$) — левой и правой границами интервала номеров игроков, которых планируется привлечь к этим сборам. Другими словами, к сборам под номером j будут привлечены все игроки с номерами не меньше l_j и не больше r_j. Для проведения сборов необходимо разделить всех участвующих в них игроков **поровну** на защитников, полузащитников и нападающих. Значения l_j и r_j для всех сборов Паша выбрал так, чтобы количество участников сборов было кратно трём.

Паша знает, что если i-й знакомый играет в защите, то качество его игры равняется d_i, если в нападении — равняется a_i, если же он играет в полузащите, то качество игры для него равно $\frac{1}{2}(a_i + d_i)$. Теперь для каждых сборов он хочет знать максимальное возможное суммарное значение качества, которое можно получить путём оптимального разделения участников на нападающих, защитников и полузащитников.

Ваша задача — написать программу, которая по известным значениям a_i и d_i для всех игроков, а также известным значениям l_j и r_j для всех сборов, определит для каждых сборов максимальное возможное значение суммарного качества.

Формат входных данных

Первая строка входного файла содержит единственное целое число N — количество знакомых Паши ($3 \leqslant N \leqslant 128\,000$).

Вторая строка содержит N целых чисел a_i — значения оценок умения играть в атаке ($0 \leqslant a_i \leqslant 10^9$). Третья строка содержит N целых чисел d_i — значения оценок умения играть в защите ($0 \leqslant d_i \leqslant 10^9$).

Четвёртая строка входного файла содержит единственное целое число M — количество запланированных сборов ($1 \leqslant M \leqslant 100\,000$).

Далее следует M строк, каждая из которых описывает очередные сборы двумя целыми числами l_j и r_j — левой и правой границей номеров привлекаемых игроков соответственно ($1 \leqslant l_j < r_j \leqslant N$). Гарантируется, что количество привлекаемых игроков для любых сборов кратно трём.

Формат выходных данных

Для каждых сборов в выходной файл требуется вывести одно вещественное число — максимальное возможное суммарное качество игры. Выведенные значения должны отличаться от правильных не более чем на 10^{-2}.

Пример

input.txt	output.txt
11 1 4 3 2 4 6 7 2 3 5 1 9 3 5 2 5 9 2 4 5 8 3 5 1 6 2 10 5 7 6 11 3 8	32.5 46.5 20.500 32 30.5

Задача 9. Примитивные делители

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Топ-кодер Витя очень любит интернет. Каждый день он регистрируется на большом количестве новых сайтов, и каждый раз его просят придумать очередной пароль для входа. Витя написал генератор паролей, который выдает новый пароль по данному простому числу. Осталось только научиться генерировать различные простые числа.

У Вити есть любимое число q. В качестве простых чисел он решил брать простые делители числа $q^n - 1$, каждый новый день увеличивая показатель n. Однако, чтобы простые числа не повторялись, в n-й день он выбирает лишь те простые числа, которые не встретились ранее, т.е. не были простыми делителями числа $q^m - 1$ для некоторого $m < n$.

Помогите топ-кодеру написать программу, которая будет находить подходящие простые делители для заданных q и n. Поскольку генератор паролей Вити работает лишь с небольшими числами, то необходимо искать лишь простые числа, не превосходящие 10^7.

Формат входных данных

В единственной строке входного файла даны два натуральных числа q и n ($2 \leqslant q \leqslant 10^9$, $1 \leqslant n \leqslant 10^9$).

Формат выходных данных

В первую строку выходного файла нужно вывести количество искомых простых делителей числа $q^n - 1$. Во вторую строку нужно вывести эти делители в порядке возрастания через пробел. Все они должны быть простыми, не должны превышать 10^7, и не должны встречаться при меньшем n.

Пример

input.txt	output.txt
3 4	1 5
2 6	0

Пояснение к примеру

В первом примере $q^n - 1 = 80 = 2^4 \cdot 5$. Таким образом, простыми делителями будут числа 2 и 5. Однако 2 делит $3^1 - 1$, а 5 не делит числа $3^1 - 1$, $3^2 - 1$ и $3^3 - 1$, поэтому Витя может взять только один простой делитель – число 5. Во втором примере $2^6 - 1 = 63 = 3^2 \cdot 7$. Ни один простой делитель не подходит, поскольку 3 делит $2^2 - 1$, 7 делит $2^3 - 1$.

Задача 10. Билеты

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Киса и Ося уселись на добытых непосильным трудом стульях и играют в следующую занимательную игру. Каждый держит в руках трамвайный билет с шестизначным номером от 000000 до 999999. Они по очереди называют цифры слева направо на своем билете, и после каждой пары цифр проигравший, тот кто назвал меньшую цифру, выполняет желание победителя.

Определите, пожалуйста, кто сколько желаний исполнит.

Формат входных данных

В первой строке входного файла записан номер билета Оси, а во второй — номер билета Кисы. Каждый номер состоит из шести цифр в диапазоне от `0` до `9`. В номере могут быть ведущие нули.

Формат выходных данных

В первую строку выходного файла нужно вывести число желаний, которые исполнит Ося, а во вторую — сколько желаний исполнит Киса.

Пример

input.txt	output.txt
013936 132946	3 1

Пояснение к примеру

Ося проигрывает на первой, второй и пятой цифрах, а Киса — на третьей цифре.

Задача 11. Сглаживание логарифма

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Жизненная траектория Васисуалия Лоханкина вся прямо-таки изогнутая, ну вылитый логарифм (правда, умноженный на некоторое число). На досуге, размышляя о трагической судьбе русской интеллигенции и о великой сермяжной правде, он решил эту траекторию исследовать повнимательнее. А именно, он решил ее заменить на более понятную широким массам ломаную линию. При этом он, естественно, хочет, чтобы, с одной стороны, выбранная им ломаная как можно меньше отличалась от исходного логарифма, а с другой, количество звеньев этой ломаной, не превышало заданного числа n.

Рассмотрим проблему более формально. Будем говорить, что определённая на отрезке $[a, b]$ функция является ломаной с не более чем n звеньями, если её график можно разбить на n частей, каждая из которых является отрезком прямой. Обозначим класс непрерывных ломаных, определённых на заданном отрезке $[a, b]$ и имеющих не более n звеньев, через $B_n[a, b]$. Расстоянием между двумя функциями $f(x)$ и $g(x)$ на отрезке $[a, b]$ назовём число:

$$D(f, g) = \max_{x \in [a,b]} |f(x) - g(x)|$$

Требуется найти ломаную из класса $B_n[a, b]$, расстояние от которой до функции $f(x) = c \ln x$ минимально.

Формат входных данных

В первой строке входного файла задано одно целое число m — количество тестовых примеров ($1 \leqslant m \leqslant 20$).

Далее следует m строк, в каждой из которых записано по четыре числа n, c, a и b, где n — количество звеньев ломаной ($1 \leqslant n \leqslant 20$), c — множитель при логарифме ($1 \leqslant c \leqslant 10^4$), a, b — левая и правая границы рассматриваемого отрезка соответственно ($10^{-2} \leqslant a < b \leqslant 10^2$). Числа n и c — целые, а числа a, b — вещественные, заданные не более чем с двумя знаками после десятичной точки.

Формат выходных данных

В выходной файл необходимо вывести m строк, в каждой из которых должно быть записано одно вещественное число — минимально возможное максимальное отклонение ломаной с не более чем n звеньями от логарифма, ответ на соответствующий тестовый пример. Абсолютная или относительная погрешность вашего ответа не должна превышать 10^{-8}.

Пример

input.txt	output.txt
2 1 1 1 10 1 1 0.5 7	0.309517460 0.398765993

Задача 12. Космические сигналы

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Группа космических исследований по-прежнему занимается поисками внеземного разума. Некоторое время назад из космоса было принято два сигнала, предположительно отличающиеся от простого шума. Этим сигналам присвоили имена X и Y. Оба сигнала закодировали в виде строк, состоящих из маленьких латинских букв.

Недавно космических исследователей вновь постигла удача — был принят ещё один сигнал. Этому сигналу дали имя Z и также закодировали в виде строки из маленьких латинских букв. Теперь стоит следующая задача: требуется выяснить, является ли Z принципиально новым сигналом или он — всего лишь комбинация уже известных сигналов X и Y.

Считается, что сигнал Z является комбинацией сигналов X и Y в том и только в том случае, если он содержит непересекающиеся вхождения сигналов X и Y. Иными словами, если можно указать в Z две неперекрывающихся подстроки, одна из который равна X, а другая равна Y (если X является подстрокой Y или Y является подстрокой X, подстроки считаются перекрывающимися).

Формат входных данных

В первой строке входного файла содержится количество тестовых наборов данных (от 1 до 1000). Далее следуют тестовые наборы в указанном количестве, по три строки каждый. В первой строке тестового набора содержится представление сигнала Z, в следующих двух строках записаны представления сигналов X и Y. Каждая из этих строк не пуста.

Суммарная длина всех строк во всех тестовых наборах не превосходит 10^6.

Формат выходных данных

Для каждого тестового набора в выходной файл нужно вывести ответ в отдельную строку, «`YES`» (без кавычек), если сигнал Z является комбинацией сигналов X и Y, и «`NO`» (без кавычек) — в противном случае.

Примеры

input.txt	output.txt
3 abacada aba ada abacada aba aca abacada aba aaa	YES NO NO

Замечание

Во втором тестовом наборе обе строки X и Y присутствуют в строке Z как подстроки. Однако эти подстроки перекрываются: третий символ строки Z принадлежит им обеим.