Problem A. Убийство короля

Input file:	assassination.in
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes
Feedback:	

В процветающей стране Берляндии имеется n прекрасных городов, некоторые из которых соединены односторонними дорогами. У Берляндии есть постоянный и давний враг — Бирляндия, с которой долгие годы идет война. Военачальники Бирляндии поняли, что им не одолеть Берляндию в честной борьбе, поэтому они решили действовать при помощи различных диверсий. И одна из этих диверсий заключается в убийстве короля Берляндии. Король живет в своем дворце в столице страны (город с номером s), в который совершенно невозможно проникнуть. Однако, стало известно, что вскоре король предпримет поездку в город с номером t с целью открытия шоколадной фабрики. В городе t также приняты беспрецедентные меры безопасности, поэтому организовать покушение там также не удастся. Было выяснено, что король собирается совершить остановку в каждом из городов, через которые пройдет его путь, для общения с народом. Бирляндцы пришли к выводу, что именно в один из таких моментов удобнее всего убить вражеского короля. Но маршрут короля держится в секрете, поэтому совершенно непонятно, в каком городе готовить покушение. После долгих раздумий было решено выбрать один из городов, через которые король проедет в любом случае. Найдите все такие города.

Input

В первой строке записано четыре целых числа n, m, s и t ($2 \le n \le 100000, 1 \le m \le 300000, 1 \le s, t \le n, s \ne t$). В следующих m строках записано по 2 целых числа x_i, y_i — номера городов, соединенных дорогами ($1 \le x_i \ne y_i \le n$). Никакая пара городов не соединена более чем одной дорогой в одном направлении.

Output

В первой строке выведите количество городов k, через которые обязательно пройдет маршрут короля. Во второй строке выведите k чисел — номера этих городов, разделенные пробелами. Города выводите в порядке возрастания номера.

Examples

assassination.in	stdout
4 3 1 4 1 2 2 3 3 4	2 2 3
4 4 1 4 1 2 2 3 3 4 1 3	1 3
4 5 1 4 1 2 2 3 3 4 1 3 2 4	0

Problem B. Командный пост (Division 1 Only!)

Input file:	post.in
Output file:	stdout
Time limit:	6 seconds
Memory limit:	512 megabytes

Из-за ошибки резидента покушение на короля провалилось, о замыслах бирляндского командования стало известно руководству Берляндии, после чего было принято решение о проведении «акции возмездия». Однако бирляндские войска оказали упорное сопротивление и на некоторых направлениях перешли в контратаку.

Бирляндский полковник Кругляковский завершил очередную фазу сражения взятием стратегической точки на территории Берляндии. Сейчас задача полковника — закрепиться на удержанном плацдарме. Плацдарм представляет собой круг, командный пункт Кругляковского расположен в центре этого круга. Разведка нашла n точек на ограничивающей плацдарм окружности, пригодных для постройки наблюдательных вышек. Однако ресурсов у группы Кругляковского хватит только на постройку k наблюдательных вышек, соединённых прямыми участками заграждения.

Требуется выбрать вышки так, чтобы защищённая заграждением территория плацдарма имела максимальную площадь. Заграждение представляет собой выпуклый многоугольник с наблюдательными вышками в вершинах. Командный пункт не должен находиться вовне заграждения (но может находиться на его границе). Для целей задачи выпуклостью земной поверхности, линейными размерами наблюдательного поста и командного пункта пренебречь (то есть считать последние точками).

Input

В первой строке входного файла содержатся два целых числа n и k ($3 \le k \le n \le 1000$). Последующие n чисел задают полярные углы точек, в которых можно разместить вышки. Каждый угол задан в отдельной строке и содержит не более четырёх цифр после десятичной точке. Все углы попарно различны и расположены в интервале от 0 до 2π. Командный пункт расположен в начале координат.

Output

В выходной файл выведите k различных целых чисел в интервале от 1 до n — номера точек, в которых требуется построить вышки. Точки должны быть отсортированы по величине их полярного угла. Если решений несколько, выведите любое из них. Если построить заграждение так, чтобы внутри него или на его границе оказался командный пункт, нельзя, выведите 0.

Examples

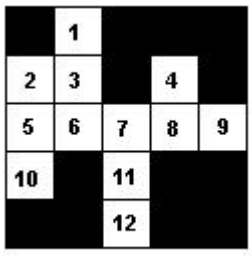
post.in	stdout
4 4 1.57 0 3.14 4.71	2 1 3 4
4 3 1.57 0 3.14 4.71	2 1 4

Problem C. Кроссворд (Division 1 Only!)

Input file:	`crossword.in`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes
Feedback:	

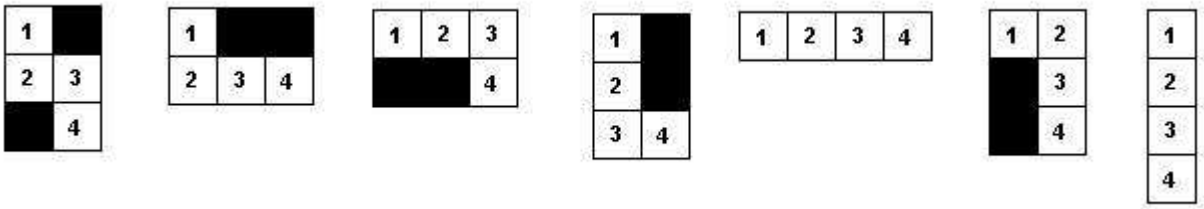
Петя любит кроссворды, но ненавидит их разгадывать. У него огромная коллекция кроссвордов следующего вида. Каждый кроссворд состоит из главного слова по горизонтали и нескольких слов по вертикали, пересекающихся с главным словом. Количество слов по вертикали всегда совпадает с количеством букв в главном слове. Каждое слово содержит не менее одной буквы.

Вы спросите, что Петя делает с кроссвордами, если не разгадывает их. Ответ достаточно неожиданный. Он испытывает на них новый способ шифрования, разработанный им самим. Петя берет кроссворд описанного вида и располагает в пустых клетках числа от 1 до n (где n — общее количество пустых клеток в кроссворде). Он нумерует клетки ряд за рядом сверху вниз. В свою очередь, каждый ряд нумеруется слева направо. При этом числа ставятся только в пустые клетки, закрашенные черным клетки пропускаются.



Затем Петя выписывает последовательность чисел, полученную путем движения столбец за столбцом слева направо и сверху вниз по каждому столбцу. Таким образом, он получает кодовую последовательность. Например, для кроссворда на рисунке он получит последовательность “2 5 10 1 3 6 7 11 12 4 8 9”.

Теперь Петю интересует декодирование. К сожалению, его не всегда можно построить однозначным образом. Например, последовательность “1 2 3 4” соответствует нескольким кроссвордам (см. рисунок ниже).



Как бы то ни было, у Пети есть кодовая последовательность, и он хочет найти по ней любой корректный кроссворд описанного типа. Если найти любую последовательность чисел, которая может располагаться в клетках главного слова, Петя сможет восстановить по ней весь кроссворд. Помогите Пете раскодировать данную последовательность.

Input

В первой строке содержится целое число n ($1 \le n \le 200\,000$) — длина кодовой последовательности. Вторая строка содержит перестановку чисел от 1 до n. Числа в кодовой последовательности разделены пробелами.

Output

В первой строке выведите целое число k — длину главного слова по горизонтали. Вторая строка должна содержать k чисел, которые будут располагаться в клетках главного слова. Выводите числа через пробел, в порядке возрастания. Гарантируется, что хотя бы одно решение существует.

Examples

crossword.in	stdout
12 2 5 10 1 3 6 7 11 12 4 8 9	5 5 6 7 8 9
4 1 2 3 4	1 3

Note

Во втором примере в качестве ответа выбран самый правый кроссворд с рисунка. Очевидно, что главное слово имеет длину 1, но при этом за главное слово можно считать любую из четырех клеток. В таких случаях разрешается выводить любой корректный ответ.

Problem D. Групповой этап

Input file:	`group.in`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Тем временем, пограничные конфликты, называемые в прессе "контркриминальными операциями", далеко не всегда оказываются в центре внимания в Берляндии. Куда больше внимания привлекла реформа футбольного чемпионата. В целях согласования формата с крупными европейскими турнирами чемпионат Берляндии не только переходит на систему "зима-лето", но и меняет схему проведения: сначала играется групповой этап, а затем — игры по олимпийской системе (плей-офф).

Групповой этап устроен следующим образом: участвующие в чемпионате команды первоначально делятся на группы по 4 команды в каждой группе. Каждые две команды в группе играют по одному матчу между собой. Если команда выигрывает матч, она получает 3 очка, если играет вничью — одно очко и если проигрывает, то очков не получает. После завершения всех матчей подсчитываются очки для каждой из команд и команды упорядочиваются по набранным очкам. Две команды, занявшие первое и второе места, попадают в плей-офф. Если две или более команд набрали равное количество очков, то учитываются дополнительные параметры (например, разница забитых и пропущенных мячей, результат матча между командами и так далее), а если и дополнительные параметры не позволяют упорядочить команды, финальную расстановку определяет жеребьёвка.

При такой схеме могут происходить нетривиальные на первый взгляд ситуации — например, команда X в группе 1 набирает вдвое меньше очков, чем команда Y в группе 2, при этом команда X выходит в плей-офф, а команда Y — нет.

Действительно, рассмотрим следующую ситуацию.

Группа 1 (команды: A, B, C, X):

- Команда A обыгрывает все остальные команды со счётом 1-0 и получает 9 очков.
- Команда B проигрывает командам A и X, но обыгрывает команду C со счётом 1-0.
- Команда C проигрывает командам A и B, но обыгрывает команду X со счётом 1-0.
- Команда X проигрывает командам A и C, но обыгрывает команду B со счётом 4-0.

В результате по окончании группового этапа положение команд будет следующим:

- Команда A: 9 очков.
- Команда X: 3 очка.
- Команда B: 3 очка.
- Команда C: 3 очка.

Команда X, имеющая лучшую разницу забитых и пропущенных мячей, выходит в плей-офф со второго места вместе с командой A.

Группа 2 (команлы: E, F, G, Y):

- Команда E проигрывает все матчи в группе и получает 0 очков.
- Команда F выигрывает у команды E со счётом 2-0, у команды Y со счётом 1-0, и проигрывает команде G.
- team G выигрывает у команды E со счётом 2-0, у команды F со счётом 1-0, и проигрывает команде Y.
- team Y выигрывает у команды E со счётом 1-0, у команды G со счётом 1-0 и проигрывает команде F.

В результате по окончании группового этапа положение команд будет следующим:

- Команда F — 6 очков.
- Команда G — 6 очков.
- Команда Y — 6 очков.

- Команда E — 0 очков.

У трёх команд по 6 очков, однако команда Y имеет худшую разницу забитых и пропущенных мячей и не выходит в плей-офф.

В преддверии дальнейших реформ берляндского футбола рассмотрим более общую ситуацию. Пусть в групповом этапе некоторого турнира участвуют n команд, n из которых выходят в плей-офф. Всего в каждой группе играется $n \cdot (n-1)/2$ матчей. Для упрощения ситуации предположим, что при равенстве набранных очков положение команд сразу же определяется с помощью жребия.

Требуется найти два числа:

- Минимальное количество очков, которое может набрать команда, чтобы всё ещё (при удачной жеребьёвке) иметь шансы на выход в плей-офф.
- Максимальное количество очков, при котором ещё не гарантируется попадание в плей-офф независимо от исхода жеребьёвки.

Input

Входнй файл содержит два целых числа n и m, разделённых пробелом ($2 \le n \le 100$, $1 \le m \le n-1$) — количество команд в группе и количество команд, выходящих в плей-офф.

Output

Выведите два требуемых в условии задачи числа.

Examples

group.in	stdout
4 2	2 6

Problem E. Раскраска схемы (Division 1 Only!)

Input file:	coloring.in
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

Железнодорожная система Берляндии состоит из n станций. Станции соединены $n-1$ перегонами, на каждом из которых организовано двустороннее движение поездов. При этом между любыми двумя станциями можно проехать по железной дороге.

Готовясь к очередной серии актов саботажа на территории Берляндии, бирляндская разведка заказала детальную схему берляндских железных дорог. При этом для того, чтобы диверсантам было легче ориентироваться по карте, каждый перегон должен быть окрашен в свой цвет так, чтобы любые два перегона, выходящие из некоторой станции, были окрашены в разные цвета.

При печати в типографии расценка на печать одной линии цвета t (в типографии цвета пронумерованы в некотором порядке, начиная с единицы) равна t бирляндских марок. Суммарная стоимость схемы складывается из затрат на печать изображения каждого перегона. Учитывая необходимость экономии военного бюджета, военное руководство Бирляндии собирается сделать карту как можно более дешёвой.

Ваша задача — найти вариант раскраски перегонов на схеме, затраты на который будут минимальны.

Input

Первая строка входного файла содержит одно целое число — общее количество станций n ($2 \le n \le 100$). Последующие $n-1$ строк описывают перегоны. Каждый перегон задан как пара целых чисел a_j и b_j — номера станций, соединённых j-й дорогой ($1 \le a_j, b_j \le n$, $a_j \ne b_j$). Станции пронумерованы от 1 до n.

Output

В первой строке выходного файла выведите минимальную стоимость печати схемы. Вторая строка должна содержать $n-1$ целое число, j-е из этих чисел обозначает цвет j-го в порядке перечисления во входном файле перегона. Если существует несколько оптимальных раскрасок, выведите любую из них.

Examples

coloring.in	stdout
8 8 1 5 1 6 5 5 2 3 2 7 8 6 4	11 2 1 2 3 1 1 1
5 3 5 4 3 5 2 4 1	6 2 1 1 2

Problem F. Боулинг в Берляндии (Division 1 Only!)

Input file:	`bowling.in`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

После анализа экономической и военно-политической обстановки руководство Бирляндии приняло предложение Берляндии начать мирные переговоры. Первый, неофициальный, раунд переговоров проходил в боулинг-клубе Министерства Обороны Берляндии. В этом клубе посетителям предлагается новая игра — «стратегический боулинг». Правила "стратегического боулинга" таковы:

Задача игрока — набрать как можно больше очков, сбивая шаром, запущенным по плоской (обычно деревянной или пластиковой) поверхности кегли.

Первая стадия игры — "розыгрыш" — состоит из n раундов. Изначально в каждом раунде выставлено 10 кеглей. Раунд состоит из двух попыток. За каждую попытку игрок получает количество очков, равное количеству сбитых в этой попытке кеглей (без учёта модификаторов). По результатам раунды делятся на три типа:

- Если игрок сбивает все кегли с первой попытки, такой раунд называтся "*strike*" (страйк). В случае страйка вторая попытка не используется, так как все кегли уже сбиты. Набранные в следующем раунде после страйка очки удваиваются.
- Если игрок не делает страйк, но сбивает все кегли в результате двух попыток, такой раунд называется "*spare*" (спэа). Набранные в первой попытке следующего после спэа раунда очки удваиваются.
- Если после раунда остались не сбитые кегли, то раунд считается обычным и очки в следующем раунде начисляются обычным порядком.

Если последний раунд был страйком, то работает правило *The Bonus Round Rule* — игрок получает дополнительный раунд, то есть для него игра состоит из $n+1$ раундов. При этом очки за обе попытки в бонусном раунде удваиваются: предыдущий удар был страйком. При этом *The Bonus Round Rule* применяется только один раз — даже если $n+1$-й раунд тоже был страйком, игра заканчивается.

Вторая стадия игры — "отладка" состоит в следующем. По завершении всех раундов игрок может переупорядочить раунды по своему желанию, но так, чтобы партия оставалась корректной (то есть при $n+1$ сыгранных раундах n-й раунд обязан быть страйком, а при n — не быть страйком).

Вам даны результаты всех раундов "розыгрыша". Ваша задача — переупорядочит раунды по правилам «отладки» так, чтобы полученное количество очков было максимальным.

Input

Первая строка входного файла содержит единственное целое число n ($1 \le n \le 50$) — количество раундов в стадии розыгрыша.

Следующие n строк описывают раунды. i-я из них содержит два неотрицательных целых числа, в сумме не превосходящих 10 — количество кеглей, сбитых в первой и второй попытках i-го раунда. Страйк обозначается как `10 0`.

Если n-й раунд был страйком, входной файл содержит ещё одну строку того же вида — количество кеглей, сбитых в первой и второй попытках $n+1$-го раунда.

Output

Выведите одно целое число — максимальное количество очков, которое игрок может набрать после перестановки раундов.

Examples

bowling.in	stdout
2 5 2 10 0 3 7	44

Problem G. Апельсин (Division 1 Only!)

Input file:	orange.in
Output file:	stdout
Time limit:	5 seconds
Memory limit:	512 megabytes
Feedback:	

Вот и наступил Новый год. Подходят к концу новогодние праздники. Подходят к концу и стратегические запасы в Петином холодильнике. Однажды Петя залез в холодильник в поисках чего-нибудь и нашел там апельсин. Друзья Пети, бывшие у него в гостях, были совсем не прочь полакомиться фруктом. Поэтому Петя решил поделить апельсин на несколько частей. Для этого он ножом делает разрезы, проходящие через центр апельсина (можно считать апельсин идеальным шаром). Каждый разрез делит шар на две одинаковые половинки. Таким образом, каждый разрез можно представить плоскостью, проходящей через центр апельсина.

С разрезанием Петя справился превосходно. Теперь ему предстоит посчитать объем каждого куска, чтобы понять, сможет ли он справедливо распределить апельсин между друзьями.

Input

Во входном файле в первой строке даны два числа r и n ($1 \le r \le 100, 1 \le n \le 300$), радиус апельсина и количество разрезов соответственно. Каждая из следующих n строк содержит по три числа x_i, y_i, z_i — координаты вектора, ортогонального плоскости, в которой лежит разрез. Центр апельсина лежит в начале координат. Координаты по модулю не превосходят 200, $|x| + |y| + |z| > 0$. Все числа во входном файле целые. Никакие два разреза не совпадают.

Output

Выведите в первую строку число m — количество кусков. В следующие m строк выведите объемы кусков в неубывающем порядке. Абсолютная или относительная погрешность для выводимых объемов не должна превышать 10^{-6}.

Examples

orange.in	stdout
1 1 3 4 5	2 2.0943951024 2.0943951024
10 2 0 0 1 1 0 0	4 1047.1975511966 1047.1975511966 1047.1975511966 1047.1975511966

Problem H. Puzzle Quest

Input file:	`puzzle.in`
Output file:	`stdout`
Time limit:	10 seconds
Memory limit:	512 megabytes

В качестве отдыха между решением задач Вам предлагается сыграть в Puzzle Quest.

Игра «Puzzle Quest» проходит на доске $n \times m$. В данный вариант игры играет один участник (чтобы не отвлекать остальную команду от решения задач). Правила игры таковы:

Каждая из клеток доски представлена одним из следующих символов (соответсвующим типам фишек):

- "`.`" — пустое поле
- "`R`" — красный шарик
- "`G`" — зелёный шарик
- "`B`" — синий шарик
- "`Y`" — жёлтый шарик
- "`D`" — звёздочка
- "`M`" — монета
- "`*`" — череп
- "`#`" — огненный череп

Ходы в игре делаются в следующем порядке:

- Вы можете поменять содержимое двух соседних по вертикали или горизонтали полей (в том числе и сдвинуть содержимое клетки на соседнюю пустую клетку), если после этого происходит как минимум один взрыв (определение «взрыва» см. далее).
- После завершения хода производятся «взрывы». Если на поле присутствуют 3 или более фишек одного типа (отличного от пустого поля), расположенных подряд по горизонтали или вертикали, все они отмечаются для взрыва. После того, как всё поле просмотрено и нужные фишки отмечены, одномоментно все помеченные фишки удаляются.
- Если после взрыва снизу некоторой фишки оказалось пустое место, то соответствующая фишка перемещается на это место, и так далее для всех фишек до тех пор, пока процесс не остановится.
- Предыдущие два шага («взрыв» и «падение») повторяются до тех пор, пока при очередном просмотре поля можно «взорвать» какой-то блок клеток.

«Огненный череп» обладает особенным свойством: если огненный череп находится в клетке (r,c) и оказывается задействован во взрыве, он также уничтожает фишки, расположенные в соседних с ним клетках с координатами $(r-1,c-1)$, $(r,c-1)$, $(r+1,c-1)$, $(r-1,c)$, $(r+1,c)$, $(r-1,c+1)$, $(r,c+1)$, $(r+1,c+1)$. При просмотре поля на предмет «взрывающихся» последовательностей «огненный череп» считается идентичным обычному черепу, то есть если 3 или более любых черепов расположены в ряд, то все черепа и все клетки, соседние с «огненными черепами», отмечаются для взрыва; если при этом какой-то «огненный череп», не участвующий в первоначальном взрыве, оказался затронут взрывом другого «огненного черепа», то соседние с ним клетки также отмечаются для взрыва.

По заданному первоначальному положению фишек на поле Вам требуется найти последовательность ходов, которая оставляет на поле только пустые клетки, и, возможно, некоторое количество «огненных черепов».

Input

В первой строке входного файла заданы два целых числа n и m — соответственно количество строк и столбцов игрового поля. Каждая из последующих n строк содержит m символов, описывающих изначальную расстановку фишек на поле и принадлежащих множеству "`.RGBYDM*#`".

Все тесты к этой задаче открыты: гарантируется, что Ваше решение будет тестироваться на 54 тестах, в каждом из которых $m = n = 8$. Вы можете скачать программу, которая сгенерирует для Вас все 54 теста, а далее или решать каждый из примеров самостоятельно с помощью эмулятора игры (о нём далее), или написать программу, которая их решает.

Файлы, которые Вы можете скачать к этой задаче: генератор тестов в файле "gentest.jar". Файл может быть скачан по ссылке "`http://acm.sgu.ru/download/puzzle/gentest.jar`". Вы можете запустить его без параметров для получения справочной информации, но обычное использование генератора выглядит так:

```
java -jar gentest.jar --test <test-id> --output <output-file>
```

К примеру, следующая команда выведет входной файл к 11-му тесту в файл 11.in, расположенный в текущей директории:

```
java -jar gentest.jar --test 11 --output 11.in
```

Также Вы можете скачать эмулятор игры (интерактивный визуализатор). Он содержится в файле "visualizer.jar", который доступен по адресу "`http://acm.sgu.ru/download/puzzle/visualizer.jar`".

Для запуска визуализатора в режиме демонстрации Вашего решения используется следующая команда, читающая первоначальное расположение фишек из `<input-file>`, последовательность ходов из `<output-file>`, проверяющая корректность каждого предложенного хода и отображающая результаты последовательных ходов на экране:

```
java -jar visualizer.jar --mode playback --input <input-file> --output <output-file>
```

Следующая команда читает первоначальное расположение фишек из `<input-file>` и запускает интерфейс для ручного решения тестового примера.

```
java -jar visualizer.jar --mode manual --input <input-file> [--output <output-file>]
```

Для совершения ходов вы можете использовать мышь. Если задан `<output-file>` и в процессе ручного решения Вы достигли требуемого расположения фишек (то есть поля, на котором присутствуют только пустые клетки и «огненные черепа»), сделанная Вами последовательность ходов будет записана в заданный `<output-file>`:

Гарантируется, что для каждого тестового примера существует решение. Если Вы не уверены, как работает то или иное правило, экспериментируйте с предоставленными Вами программами для лучшего понимания механизмов игры. При этом работа визуализатора при значениях n и m, отличных от 8, не гарантируется.

Output

В единственной строке выходного файла выведите последовательность ходов, которая решает данный тест.

Каждый ход представляется в виде тройки d,r,c, где r — номер строки (начиная с 0), c — номер столбца (начиная с 0) той фишки, которую Вы двигаете этим ходом, а d — направление движения/обмена (один из символов "`LRUD`"). Остальной формат вывода понятен из примера к задаче.

Если данный тест имеет несколько решений, выведите любое из них.

Examples

puzzle.in	stdout
8 8 ....#. ......YG .G....Y* GB....G* YB....*Y BY....R* YB.G.GR#	R60L73R56D66

Note

Комментарий к примеру:

Ход R60 меняет местами жёлтый и синий шарик, приводя к двум взрывам (3 жёлтых шарика взрываются в столбце 0, 4 синих шаника взрываются в столбце 1). После этого два зелёных шарика падают вниз, приводя к:

```
........
......#.
......YG
......Y*
......G*
......*Y
......R*
GG.G.GR#
```

В этой позиции L73 перемещает зелёный шарик на пустое место, что приводит к взрыву трёх зелёных шариков по горизонтали (при этом запись R72 не может быть использована, так как ход делается фишкой, а не пустым местом). Получаем следующую позицию:

```
........
......#.
......YG
......Y*
......G*
......*Y
......R*
.....GR#
```

Хол R56 меняет местами обычный череп и жёлтый шарик, после чего 5 черепов взрываются. Один из этих черепов — «огненный», так что заодно взрываются 2 соседних с ним красных шарика. После падения оставшихся шариков вниз поле выглядит так:

```
........
........
........
......#.
......Y.
......Y.
......G.
.....GYG
```

И наконец, ход D66 меняет местами зелёный и жёлтый шарики, взрывая последние три зелёных и последние три жёлтых шарика на поле. Остаётся только один «огненный череп» и пустые клетки — то есть задача решена.

Problem I. Ложь, наглая ложь и статистика

Input file:	statistics.in
Output file:	stdout
Time limit:	5 seconds
Memory limit:	512 megabytes
Feedback:	

Как известно, существуют три вида лжи: ложь, наглая ложь и статистика. Журналисту Васе срочно надо отнести в редакцию газеты «Берляндская правда» материал для первой страницы, подойдет какое-нибудь громкое открытие, скандал звезд или что-то в этом роде. К сожалению, в Берляндии все спокойно, если не считать серии таинственных ограблений киосков по продаже мороженного.

Журналист Вася решил поразить общественность неожиданным открытием — статистически ограбления лежат на одной прямой! Он отметил на карте точки, где были совершены ограбления. Теперь ему надо провести такую прямую, чтобы вблизи от нее оказалось максимальное число точек. Вася решил, что вблизи — это значит на расстоянии не более d. Количество таких точек будет убийственным аргументом в пользу его теории!

Помогите Васе узнать наибольшее количество точек на карте, которые лежат на расстоянии не более d относительно некоторой прямой.

Input

Первая строка содержит пару целых чисел n и d ($1 \le n \le 1000, 0 \le d \le 1000$). Следующие n строк содержат координаты точек. Каждая строка содержит пару целых чисел x_i, y_i ($-1000 \le x_i, y_i \le 1000$) — координаты i-го ограбления. Никакие два ограбления не произошли в одной точке.

Output

Выведите искомое максимальное количество точек.

Examples

statistics.in	stdout
4 3 0 0 5 5 5 0 0 5	4
4 2 0 0 5 5 5 0 0 5	3

Problem J. Такси

Input file:	`taxi.in`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes
Feedback:	

Только что в одном небезызвестном клубе закончилась вечеринка, на которой была компания из n юношей и m девушек. Все приятно провели время, и теперь настала пора расходиться по домам. Все изрядно устали, да и время позднее, поэтому решили ехать домой на такси. Оказалось, что все члены веселой компании живут на одной улице, да еще и на той самой, на которой находится клуб! Дело в том, что в городе есть одна центральная улица, в одной части которой расположены все самые модные заведения города, а в другой живут его самые модные обитатели. Поэтому оказалось, что всем надо ехать в одну сторону от клуба по этой улице, и за счет этого можно здорово сэкономить деньги (посещение модных клубов и так недешевое удовольствие).

В одно такси помещается от одного до четырех человек. Таксист едет до дома того из его пассажиров, кто живет дальше остальных от клуба, попутно высаживая остальных пассажиров у их домов. Тарифы местных такси таковы, что они берут один бурль за каждый километр пути. Местные таксисты обладают весьма сомнительной репутацией, поэтому наши друзья решили, что будут садиться в такси таким образом, чтобы в каждое такси у клуба сел хотя бы один юноша (но ему не обязательно ехать весь путь, он может сойти раньше девушек).

Определите, каким образом членам компании надо рассаживаться по такси, чтобы суммарное количество потраченных ими на проезд денег было минимальным.

Input

В первой строке записано целое число n ($1 \le n \le 2011$) — количество юношей. Каждая из последующих n строк содержит имя юноши и расстояние от клуба до его дома в километрах. Далее дано количество девушек m в отдельной строке и описания девушек в таком же формате ($0 \le m \le 2011, m \le 3n$). Имена юношей и девушек состоят из латинских букв, первая буква имени заглавная, остальные строчные. Имя имеет длину от 1 до 15 символов. Все имена юношей и девушек попарно различны. Расстояние до дома это целое неотрицательное число не более 10^4.

Output

В первой строке выведите наименьшее возможное количество потраченных членами компании денег.

Во второй строке выведите k — количество такси, на которых им следует уезжать. В следующих k строк выведите, кто в каком такси поедет. Изучите примеры для уточнения формата вывода. Если решений несколько, выведите любое.

Examples

taxi.in	stdout
2 Anton 5 Maxim 10 5 Anna 1 Maria 12 Tanya 10 Elena 8 Marina 6	18 2 Taxi 1: Maxim, Maria, Tanya and Elena. Taxi 2: Anton, Marina and Anna.
1 Romeo 100 1 Juliet 200	200 1 Taxi 1: Romeo and Juliet.
1 Jack 17 0	17 1 Taxi 1: Jack.

Problem K. Точки (Division 2 Only!)

Input file:	`points.in`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Feedback:	

На плоскости заданы две точки (x_f, y_f) и (x_t, y_t). Выясните, можно ли добраться от (x_t, y_t) до (x_f, y_f) с помощью некоторого (возможно, нулевого) количества разрешённых преобразований.

Для заданной точки (a, b) разрешены следующие преобразования:

- Перевести в точку (b, a).
- Перевести в точку $(a, -b)$.
- Перевести в точку $(a + b, b)$.
- Перевести в точку $(2a, b)$.

Input

Первая и единственная строка входного файла содержит 4 целых числа x_f, y_f, x_t, y_t ($-10^{10} \le x_f, y_f, x_t, y_t \le 10^{10}$).

Output

Выведите '`YES`', если (x_t, y_t) достижима из (x_f, y_f) с помощью разрешённых преобразований и '`NO`' в противном случае.

Example

points.in	standard output
5 7 4 2	YES

Problem L. Минимальное значение (Division 2 Only!)

Input file:	`minimum.in`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Feedback:	

Даны n целых чисел $k_1, k_2, \ldots, k_n$ и целое число x, удовлетворяющее тождеству ${x_1}^{k_1} \cdot {x_2}^{k_2} \cdot \ldots \cdot {x_n}^{k_n} = x$.

Для заданных $a_1, a_2, \ldots, a_n$ и $y_1, y_2, \ldots, y_n$ найдите минимальное положительное значение v для следующего выражения: $a_1 \cdot {x_1}^{y_1} + a_2 \cdot {x_2}^{y_2} + \ldots + a_n \cdot {x_n}^{y_n}$.

При этом $x_1, x_2, x_3, \ldots, x_n$ обязаны быть положительными, но не обязаны быть целыми.

Input

В первой строке входного файла заданы два целых числа n и x. Вторая строка содержит n целых чисел n $k_1, k_2, \ldots, k_n$, третья — n целых чисел $a_1, a_2, \ldots, a_n$ и четвёртая — n целых чисел $y_1, y_2, \ldots, y_n$ ($1 \le n \le 20,\ 1 \le x \le 10^6,\ 1 \le k_i, a_i, y_i \le 20,\ x_i > 0$).

Output

Выведите одно число — минимальное положительное значение заданного выражения v, округлённое до ближайшего целого. Например, 13.7 округляется до 14, а 11.3 — до 11. Гарантируется, что среди тестовых примеров не будет случая, при котором дробная часть ответа равна 0.5.

Example

minimum.in	standard output
1 9 2 4 4	324
2 6 1 1 1 1 1 1	5

Problem M. Неравенства (Division 2 Only!)

Input file:	inequality.in
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Feedback:	

Заданы n неравенств. Каждое из неравенств может относиться к одному из следующих 4 типов:

- Тип 1: $a > v$
- Тип 2: $a < v$
- Тип 3: $a = v$
- Тип 4: $a \neq v$.

a — переменная, которая принимает только целые неотрицательные значения, v — целочисленная константа. Требуется определить, какое наибольшее количество неравенств могут быть справедливы одновременно, и найти минимальное значение переменной a, при котором этот максимум достигается.

Input

В первой строке входного файла содержится целое число n — общее количество неравенств. Каждая из последующих n строк содержть два целых числа t_i и v_i. t_i обозначает тип неравенства, а v_i — константу в правой части соответствующего неравенства ($1 \le n \le 10^5,\ 1 \le t_i \le 4,\ 1 \le v_i \le 10^{18}$).

Output

Выведите два целых числа: максимальное количество неравенств, которые могут выполняться одновременно, и минимальное значение a, при котором это возможно.

Example

inequality.in	standard output
4 1 16 2 8 3 7 4 6	3 7

Problem N. БайтДональдс (Division 2 Only!)

Input file:	btdonalds.in
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Feedback:	

Дональд МакБайт является владельцем сети из M ресторанов медленного обслуживания "Байтдональдс". В связи с экономическим кризисом график работы ресторанов изменяется следующим образом. Сначала Дональд разбил день на N временных интервалов. Для каждого интервала времени j в каждом ресторане i вычислены количество посетителей B_{ij}, количество официантов A_{ij} и сумма C_{ij}, которую тратит один клиент в данном ресторане при посещении его в данный временной интервал. При этом концепция "Байтдональдс" — ни один официант не должен обслужиать более одного клиента за временной интервал — остаёся в силею.

Изучив данные по ресторанам, Дональд хочет изменить график их работы так, чтобы каждый ресторан за сутки работал только в течение одного временного интервала.

Ваша задача — найти максимальную выручку, которую МакБайт получит с сети ресторанов за сутки после изменения графика работы.

Input

Первая строка входного файла содержит два целых числа M и N. Последующие M строк содержат по N целых чисел, j-е число в i-й строке задаёт значение A_{ij}. Далее в аналогичном формате следуют M строк, описывающих B_{ij}, а за ними — M строк, описывающих C_{ij}. ($1 \le M, N \le 100,\ 1 \le A_{ij}, B_{ij} \le 5000,\ 0 \le C_{ij} \le 10^9$).

Output

Выведите одно целое число — максимальную суточную выручку сети ресторанов "БайтДональдс" после изменения графика работы.

Example

btdonalds.in	standard output
2 3 2 3 1 4 3 2 4 3 2 2 3 3 6 7 2 2 1 7	35