Problem A. Кампус

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

В Силиконовой долине очень много компаний. Это неудивительно, ведь в эти компании с радостью берут на работу участников сборов в Ижевске и чемпионатов Урала. Некоторые из компаний настолько суровы, что им нужен огромный кампус для инженеров. Но многие здания уже заняты, и найти что-то, расположенное недалеко друг от друга, практически невозможно. В таком случае нужно искать то, что подходит лучше всего.

Бывшие участники первых сборов в Ижевске решили основать новую компанию в Силиконовой долине. Чтобы вместить весь персонал, состоящий из участников сборов в Ижевске за все годы, компании нужно арендовать три здания.

На данный момент в аренду сдается n зданий. Можно считать, что каждое здание — это точка на плоскости, заданная своими координатами. Никакие два здания не находятся в одной точке.

Так как основатели этой компании — бывшие олимпиадники, по этическим соображениям, их пугает, когда три здания расположены на одной прямой. Более того, для эффективной связи зданий между собой необходимо хорошее сетевое соединение. Каждое здание характеризуется целым числом — качеством соединения. Чтобы получить хорошее соединение, нужно максимизировать суммарное качество соединения выбранных трех зданий. Хотите работу в этой компании? Вот ваша первая задача на интервью.

Input

В первой строке записано единственное целое число n ($3 \le n \le 10^5$) — количество зданий в Силиконовой долине. В каждой из последующих n строк записано по три целых числа x_i, y_i, s_i ($-10^5 \le x_i, y_i \le 10^5, 0 \le s_i \le 10^5$) — координаты и качество соединения для i-го здания. Гарантируется, что существуют три здания, не находящихся на одной прямой.

Output

В первой строке выведите максимальное суммарное качество соединений трех зданий, не расположенных на одной прямой. Во второй строке выведите номера этих зданий в любом порядке. Здания нумеруются от 1 до n в том же порядке, в котором они заданы во входных данных. Гарантируется, что существуют три здания, не лежащих на одной прямой.

Examples

standard input	standard output
3 0 0 1 0 1 2 1 0 3	6 3 2 1
5 0 0 1 0 1 2 0 2 3 0 4 4 1 0 5	12 5 4 3

Problem B. Черная дыра (Division 1 only!)

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

В *3141* проблема черных дыр очень тревожит жителей Сан Франциско. Сан Франциско уже давно расположен не на Земле, а в космосе. Он представляет собой сеть орбитальных станций, соединенных между собой сверхскоростными туннелями, которые невозможно разорвать. Туннели устроены таким образом, что один тоннель может свободно пройти через другой.

В этой задаче можно считать, что Сан Франциско — это замкнутая ломаная в *3D* пространстве без самопересечений и самокасаний.

Черная дыра может поглотить орбитальные станции и туннели. Чтобы этого не произошло, было решено построить специальные устройства в космосе, которые генерируют сверхпрочные лучи в две противоположные стороны, образуя тем самым непрерывную прямую, через которую ничего не может пройти, ни туннель, ни орбитальная станция, ни остальной космический мусор. Устройства были сконструированы и расположены в различных местах космического пространства. Однако, как это обычно бывает, с расчетами произошла ошибка, и ученые не уверены, правильно ли были расположены устройства.

Если устройства расположены верно, то никакая черная дыра не сможет поглотить весь Сан Франциско — туннели не пройдут через прямые. Черная дыра может возникнуть в любой точке пространства, кроме самого Сан Франциско и прямых, образованных устройствами. Сверхпрочные туннели Сан Франциско нельзя разорвать. Если устройства расположены неверно, то Сан Франциско может быть стянут в черную дыру, и прямые не помешают.

Input

В первой строке входного файла вам дано единственное целое число n — число вершин ломаной ($3 \le n \le 1000$). В следующих n строках идет описание вершин ломаной — целые числа x, y, z ($-10^4 \le x, y, z \le 10^4$). В следующей строке задано целое число m ($1 \le m \le 1000$) — число прямых. Все прямые параллельны оси OZ. В следующих m строках идет описание прямых — целые координаты x, y ($-10^4 \le x, y \le 10^4$). Гарантируется, что ни одна из данных прямых не имеет общих точек с многоугольником.

Output

Выведите одно слово "`Yes`", если ломаную можно стянуть в точку, не лежащую на устройствах, не разрывая его. Выведите "`No`" в противном случае.

Examples

standard input	standard output
4 0 0 0 2 0 0 2 2 0 0 2 0 1 1 1	No
4 0 0 0 2 0 0 2 2 0 0 2 0 1 3 3	Yes

Problem C. Кубики (Division 1 only!)

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

В Сан Франциско много замечательных магазинов для детей, где можно найти конфеты, книги и, конечно, игрушки! В этих магазинах всегда толпа детей, которые просто мечтают уйти с покупкой. Но родители не в восторге от покупки игрушки только из-за того, что их ребенок плачет и катается по полу. Родители покупают игрушки на Рождество. После очередного Рождества маленький Билли обнаружил потрясающий набор цветных кубиков в своем Рождественском чулке.

Билли играл с кубиками очень долгое время. Его самая любимая игра была строить стену. Он построил очень много разных стен разной длины, используя кубики разного цвета. Мама Билли была в восторге от творений ее сына и не переставала фотографировать все стены, построенные Билли, со всех сторон. Она слала фотографии в "Instakilogram". Но однажды мама Билли получила сообщение о том, что некоторые ее фотографии бесследно исчезли!

Мама Билли нашла только две фотографии стен сфотографированные слева и справа. Более точно, если смотреть прямо на стену, то мы увидим конструкцию с b башнями. Башня с номером i представляет собой параллелепипед размером $1 \times 1 \times h_i$, где h_i — высота i-ой башни. Высота башни может быть равна нулю. Башни стоят рядом друг с другом, формируя стену. Если мы будем смотреть на стену сбоку (слева или справа) мы увидим только одну башню, однако каждый кубик в башне реально может относиться к разным башням.

Мама Билли также нашла свой комментарий к фотографии: «постройка Билли длиной не более b, не более k разных цветов, самая высокая башня имеет ровно h кубиков». Теперь ее интересует, сколько существует различных стен с заданными условиями. Так как ответ может оказаться большим, ее интересует получить его по модулю 1000000007 ($10^9 + 7$).

Input

В первой строке выходных данных даны три целых числа b, k и h ($1 \le b, k, h \le 10^5$) — длина стены, число различных цветов, которые могли быть использованы для постройки (не обязательно использовать все цвета), и высота башни которая получилась на фотографии стены слева и справа.

Во второй строке дано описание фотографии, сделанной слева: h целых чисел c_i ($1 \le c_i \le k$), где c_i обозначает цвет i-ого кубика, считая снизу башни. В третьей строке идет описание фотографии, сделанной справа, в том же формате, что и предыдущее.

Output

Выведите единственное число — количество различных стен, подходящих под данные фотографии, по модулю 1000000007 ($10^9 + 7$).

Examples

standard input	standard output
3 3 3 1 2 3 3 2 3	132
2 3 5 1 2 3 2 1 3 2 1 2 1	5
1 1 1 1 1	1

Note

Разбор второго примера:

```
11    1    1   1    1
22   22    2   22   2
31   31   31   31   31
22   22   22   22   22
13   13   13   13   13
```

Problem D. Мусорная проблема (Division 1 only!)

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Жители Сан Франциско осознали, что на них надвигается новая проблема — проблема мусора. Решение должно быть найдено без промедлений! Но сперва нужно понять в чем дело. Ученые обнаружили, что в городе существуют самые грязные места, которые увеличиваются в размере с каждой секундой. Сейчас перед учеными стоит новая задача: определить, когда мусорные зоны вырастут до такой степени, что город разделится на несколько несвязных частей.

В этой задаче вы можете считать, что город — это многоугольник без внутренних дыр, самопересечений и самокасаний. В начальный момент времени каждый грязный участок — это просто точка. В каждый момент времени каждая точка равномерно растет во всех направлениях со скоростью 1, таким образом в момент времени t каждая грязная область — это круг радиуса t.

Вам необходимо найти первый момент времени, когда грязные области (круги) разобьют многоугольник на несколько несвязных компонент. Вам также надо найти число этих компонент.

Input

В первой строке входного файла вам будет дано одно целое число n ($3 \le n \le 100$) — число вершин многоугольника. В следующих n строках будет записан многоугольник в порядке обхода, по одной точке в строке.

Следующая строка содержит одно целое число k ($1 \le k \le 20$) — число грязных регионов. В следующих k строках перечислены точки — одна в каждой строке. Каждый грязный регион лежит внутри или на границе многоугольника.

Все координаты — это целые числа в диапазоне $[-10^5, 10^5]$.

Output

В первой строке выходного файла выведите "`Never`" если многоугольник никогда не будет разбит на несвязные области.

Иначе выведите "`Yes`" в первой строке. Во второй строке выведите первый момент времени, когда многоугольник разобьется на несвязные области. Абсолютная или относительная погрешность ответа не должна превосходить 10^{-5}. В третьей строке выведите число компонент.

Examples

standard input	standard output
4 0 0 0 5 9 5 8 0 1 2 4	Yes 2.0 2
4 0 0 8 0 8 4 0 4 2 4 1 4 3	Yes 1.0 2
3 0 0 7 2 2 4 1 2 4	Yes 3.2966535347685366 2

Problem E. Интервью

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Интервью — это самая большая забава для инженеров. Ты идешь в офис компании и решаешь задачки. Конечно, можно делать это дома, просто участвуя в соревнованиях, но после интервью можно получить работу в крутой компании и участвовать в соревнованиях из офиса.

Интервьюеры готовят задачи тщательно. Каждый хочет, чтобы его задача была самая лучшая из всех. Одна из самых сложных тем для задач — это деревья. Однажды, один интервьюер придумал задачу на деревья, которая была очень хорошая, но... Он не смог ее решить. А вы сможете?

Задача звучит несложно: посчитайте количество различных помеченных деревьев с n вершинами таких, что для каждой вершины выполняется следующее условие: количество соседей, которые являются листьями, у этой вершины не более k. Выведите ответ по модулю $10^9 + 7$.

Помеченным деревом порядка n называется дерево, вершинам которого взаимно однозначно соответствуют числа от 1 до n.

Input

В единственной строке вам даны два целых числа n, k ($2 \le n \le 100, 1 \le k \le 100$) — число вершин дерева и ограничение на листьев-соседей.

Output

Выведите ответ по модулю 1000000007 ($10^9 + 7$).

Examples

standard input	standard output
3 3	3
3 1	0

Note

Примеры помеченных деревьев с двумя и тремя вершинами:

```
1-2
1-2-3
2-1-3
1-3-2
```

Problem F. Reverse engineering

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Как вы знаете, Сан Франциско — это центр программирования. Самые сложные задачи решаются здесь. Одна из компаний по сей день не может решить очень сложную задачу.

Описание простое — "черный ящик". Что в черном ящике? Это вам и предстоит узнать. Черный ящик представляет собой программу, которая по определенным входным данным производит определенные выходные данные.

Для запросов к чёрному ящику можно использовать специальную задачу **Q**.

Также даны некоторые примеры входных и соответствующих выходных данных. Известно, что черный ящик не принимает строки длиннее 1000 символов. Вам нужно воспроизвести логику черного ящика.

Input

На вход дана одна строка не длиннее 1000 символов с разрешенными ASCII символами от 32 до 127.

Output

Ваш вывод должен быть одинаков с выводом черного ящика на данную строку.

Examples

standard input	standard output
`int foo;`	`Declare foo as int`
`double *arr[23];`	`Declare arr as array 23 of pointer to double`
`double (*a)(int, double (*)()))(int);`	`Incorrect input`

Note

Начните с данных примеров, чтобы понять, что разрешено, а что нет.

Problem G. Покрытие WiFi

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Сан Франциско — это удивительный город. Он расположен возле океана и его жители могут гулять по пляжу. А для удобного передвижения на автомобиле все важные места связаны мостами. Однако в час пик автомобилистов ждут ужасные пробки такие, что люди часами не могут сдвинуться с места.

Правительство штата Калифорния решило, что нужно что-то делать с проблемой продолжительного ожидания в пробках. И оно придумало простое решение — WiFi! Если в пробке люди смогут сидеть в интернете, то они станут счастливее. Однако правительство не хочет тратить очень много денег на этот проект, поэтому было решено установить ровно одну точку передачи — "хот-спот". Наша задача состоит в том, чтобы выбрать его оптимальное месторасположение.

Можно считать что пробки на дороге — это просто точки на плоскости. Правительство хочет установить хот-спот так, чтобы покрыть не менее чем k из этих точек. Для экономии они хотят использовать хот-спот с минимальной силой сигнала которая измеряется как радиус покрытия, т.е. максимальное расстояние до хот-спота, при котором сигнал еще ловится. Ваша задача состоит в том, чтобы найти минимальный радиус покрытия, при котором удастся покрыть не менее k точек, и местоположение хот-спота.

Input

Первая строчка входного файла содержит два целых числа n и k ($2 \le n \le 10000, 2 \le k \le min(50, n)$) — количество проблемных точек с пробками и минимальное количество точек, которое должен покрывать хот-спот. Следующие n строк содержат координаты всех проблемных точек — целые числа x, y ($-10000 \le x, y \le 10000$), все точки различны.

Output

В первой строке выведите минимальный радиус покрытия хот-спота. Абсолютная или относительная погрешность ответа не должна превышать 10^{-5}. На следующей строке выведите координаты хот-спота, с той же допустимой погрешностью.

Examples

standard input	standard output
5 4 0 0 1 1 0 1 1 0 10 10	0.7071067813839326 0.5000000000000001 0.4999999997208554
3 3 0 0 1 1 0 1	0.7071067813839326 0.49998818593030947 0.5000118140696904

Problem H. И снова покер

Input file:	`stdin`
Output file:	`standard input`
Time limit:	*standard output*
Memory limit:	1 second
Feedback:	512 mebibytes

Однажды компания программистов после очередного соревнования собралась поиграть в покер. Через много часов игры количество фишек перед каждым участником стало разным. У кого-то была целая гора, а кто-то вертел в руках последнюю. Все игроки были азартны, то и дело слышалось: "*Raise*", "*Raise*", "*All in*", "*Call*". В конце оставшиеся в игре участники раскрывали свои карты, если считали, что у них есть шанс на выигрыш, и подводились итоги. Эта задача состоит в том, чтобы помочь участникам правильно подвести итоги игры, понять кто сколько выиграл или проиграл.

Для простоты мы будем рассматривать только финальные торги (в покере типа "Texas Holdem" всего проходит четыре раунда торгов перед тем, как игроки вскрывают карты). В общем случае игра выглядит так: игрокам раздаются по две карты из 52 случайным образом. Каждый игрок знает свои карты, но не знает карты оппонентов. Затем проходят несколько раундов торгов. После каждого раунда *дилер* (о нем будет сказано позже) выкладывает на стол одну или несколько карт рубашкой вверх (выложенные карты становятся известны всем участникам). После первого раунда *дилер* выкладывает три карты, после второго добавляет еще одну, после третьего еще одну. Итого к последнему раунду торгов на столе будет лежать **пять** карт.

Колода состоит из 52 карт: 13 достоинств по 4 масти в каждом. Масти будем обозначать буквами латинского алфавита: *C, D, S, H*. Ранги карт (достоинства), начиная с младшей: *2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A*.

Комбинация карт для каждого игрока формируется из его карт и карт лежащих на столе — всего семь карт. *Из семи карт выбираются такие пять, которые формируют самую лучшую комбинацию*. Ниже перечислены комбинации в порядке уменьшения значимости:

- *Straight flush*: Смотри *Straight* и *Flush*. Если выполняются оба этих условия, тогда комбинация называется *Straight flush*. Если у нескольких игроков *Straight flush*, выигрывает тот, у кого старшая карта в *Straight* выше. В случае равенства объявляется ничья. Пример: *5H 6H 7H 8H 9H*.
- *4 of a kind*: Среди пяти карт четыре одного ранга. Если у нескольких игроков *4 of a kind*, выигрывает тот, у кого старше ранг для равных карт. Если ранги одинаковы, смотрится ранг оставшейся карты. Пример: *5H 5C 5D 5S 9H*.
- *Full house*: Среди пяти карт три имеют одинаковый ранг, оставшиеся две тоже имеют одинаковый ранг. При равенстве смотрится сперва ранг для трех одинаковых карт, затем для оставшихся двух. Пример: *5H 5S 5D 6H 7D*.
- *Flush*: Все пять карт одной масти. При равенстве смотрится ранг самой старшей карты из пяти. Если они равны, смотрится вторая старшая карта и т.д. Пример: *5H 6H 7H 8H AH*.
- *Straight*: ранги пяти карт идут подряд. При равенстве смотрится старшая карта из пяти. При этом туз (A) может как начинать *Straight* так и заканчивать. Пример: *5H 6H 7H 8H 9D*, *AH 2D 3C 4S 5C*, *TH JC QS KS AH*. Заметьте, что король не может начинать *Straight*.
- *3 of a kind*: Среди пяти карт три одного ранга. Если у нескольких игроков *3 of a kind*, выигрывает тот, у кого старше ранг для трех одинаковых карт. Если ранги одинаковы, смотрится старший ранг из оставшихся карт. При равенстве — следующая. Пример: *5H 5C 5D KH AD*.
- *2 pairs*: Среди пяти карт две имеют одинаковый ранг, а из оставшихся трех две тоже имеют одинаковый ранг. При равенстве смотрится сперва ранги для пар (по старшинству), затем для оставшейся карты. Пример: *5H 5C 6D 6H AD*.

- *Pair*: Среди пяти карт две имеют одинаковый ранг. При равенстве смотрится сперва ранг для пары, затем для оставшихся карт по старшинству. Пример: *5H 5C 2D KH AD*.
- *High card*: Смотрится ранг самой старшей карты из пяти, при равенстве смотрится вторая старшая карта и т.д. Пример: *5H 4C JD TH AD*.

Торги для последнего раунда проходят следующим образом. Игрок не может сделать ставку ниже некоторой минимальной (она называется *Blind*), если только у игрока недостаточно денег для нее, тогда он обязан идти на все ("*All in*"). Так же есть банк, где уже накопилась сумма за предыдущие раунды. Эта сумма равна количеству игроков, умноженному на *Blind* (т.е. считаем, что за все предыдущие раунды каждый из игроков поставил ровно *Blind*). Один из игроков на столе раздавал карты (его называют *дилер*). Игрок слева (следующий по часовой стрелке) от дилера начинает торги. Во время торгов, игроки по часовой стрелке называют свое действие, в результате которого он может поставить или нет некоторое число фишек на стол.

Для определения того, какие действия может совершить игрок введем понятие уровня ставок (он учитывает только ставки текущего раунда). Уровень ставок равен максимальному количеству фишек, которое поставил какой-либо из игроков в текущем раунде. В начале раунда уровень ставок равен 0, и далее он может только повышается. Чтобы оставаться в игре игрок во время своего хода должен добавить фишек, чтобы его ставка стала не меньше чем уровень ставок, либо поставить все свои оставшиеся фишки. Список возможных действий игрока:

- *Fold*. Ничего не ставить и выйти из игры, при этом все уже поставленные игроком фишки остаются на столе (т.е. он их теряет).
- *Check* или *Call*. Поставить минимальное количество фишек, чтобы ставка игрока сравнялась с текущим уровнем ставок. При этом *Check* означает, что игрок ставит 0 фишек, а *Call* — что его ставка не нулевая.
- *Raise X*. Игрок ставит столько фишек, сколько ему нужно для того чтобы повысить текущий уровень ставок на X. X должен быть не меньше, чем *Blind*.
- *All in*. Поставить все оставшиеся фишки. Это действие может как повысить, так и не изменить текущий уровень ставок. Это действие можно совершить независимо от того, сколько осталось фишек и какой текущий уровень ставок (конечно, если игрок еще не выбыл из игры). При этом любое действие, при котором игрок ставит все свои оставшиеся фишки, считается *All in* (даже если оно одновременно является *Call* или *Raise*).

После того как игрок сказал *All in* или выбыл из игры, он не участвует в дальнейших торгах (его пропускают, если ход остановился на нем). После окончания торгов, игроки раскрывают карты и определяется кто сколько выиграл денег. Торги заканчиваются, если все, кто остался в игре согласились со ставкой. То есть если очередь говорить дошла до того, кто повышал последним, он не говорит ничего, а торги считаются законченными.

Рассмотрим как происходит раскрытие карт. Если в конце торгов остался только один участник в игре, то он не откроет своих карт. Если в игре осталось несколько человек, то карты начинают открываться с последнего человека повысившего ставку (то есть последнего сказавшего "*Raise*" или "*All in*" с повышением; ситуация когда у игрока было недостаточно фишек, чтобы сказать "*Call*", и он сказал "*All in*" не считается повышением). Если никто не повышал ставку, то первым открывает карты игрок второй по часовой стрелке от дилера. Игрок откроет карты, если он еще в игре (не говорил "*Fold*"), и если у игрока есть шанс выиграть хоть что-то (то есть если игрок точно знает, что он проиграл, он не откроет карты). Если игрок отказался открыть карты это приравнивается к "*Fold*".

После раскрытия карт, вычисляется старшинство комбинации и определяется кто сколько выиграл. Игрок не может выиграть больше денег, чем он поставил. В случае если несколько человек поставили разное число фишек, то игра разбивается на несколько. Для примера рассмотрим ситуацию, когда у

трех игроков соответственно 10, 100 и 1000 фишек. Если первый скажет "*All in*", а второй и третий его поддержат (т.е. каждый поставит по 10), то игра будет идти на 30 фишек. Однако, если второй игрок скажет "*Raise* 50", а третий его поддержит (т.е. второй и третий поставят каждый по 60) то тогда игра разобьется на несколько. Первая игра будет вестись на 30 фишек, и в ней принимают участие все игроки. Во второй игре (допустим второй и третий игрок играют по 50 фишек сверху 10 для первой игры) принимают участие только второй и третий игроки, и они играют на 100 фишек. В каждой игре участник с лучшей комбинацией забирает все. Если у нескольких человек лучшая комбинация, то выигрыш делится поровну между ними. Остаток, который не делится на число участников отходит *дилеру* за хорошую раздачу (даже если *дилер* сказал "*Fold*").

В этой задаче вам дана полная ситуация за столом: карты участников, карты лежащие на столе и все действия участников в последнем раунде торгов. Вам нужно проверить правильно ли совершены действия игроков в последнем раунде (т.е. соответствуют ли они описанным выше правилам покера), определить сколько фишек получит каждый игрок и кто вскроет карты, а кто нет.

Input

В первой строке входного файла вам дано три целых числа: число игроков — n ($3 \le n \le 8$), размер *Blind* — b ($20 \le b \le 100$) и номер игрока, который является *дилером* — id ($1 \le id \le n$). В следующей строке даны n целых чисел разделенных пробелом — количество фишек, которое есть на текущий момент у соответствующего игрока — c_i ($1 \le c_i \le 20000$). Затем идет n строк, где перечислены карты на руках у соответствующего игрока через пробел. Описание одной карты состоит из двух символов, первый символ — достоинство карты, второй — масть. Масти обозначены заглавными буквами латинского алфавита: *C, D, S, H*. Достоинства, начиная с младшего: *2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A*. Все буквы — заглавные.

Следующая строчка содержит в том же формате описание пяти карт, которые были открыты на столе. Гарантируется, что вся информация о картах верная.

Далее следует описание игры в виде высказываний (смотрите пример для уточнения формата). В первой строке следует единственное целое число — m число высказываний ($1 \le m \le 50$). Далее идут m строк с высказываниями, по одному в строке. Высказывания могут быть следующими: "Fold" (игрок сбросил), "Check", "Call" (Игрок принял), "Raise x", где x — положительная сумма, на которую игрок повысил ставку $x \le 20000$, "All in". Гарантируется, что высказывания не содержат опечаток и имеют корректный формат.

Output

Если описание игры некорректное, выведите "Incorrect game log" без кавычек. Если описание было правильным, то выведите в первой строке одно слово — "Correct" без кавычек, а следующей строке выведите n целых чисел, разделенных пробелом, где i-ое число — это число фишек, которые будет иметь игрок после завершения игры. В следующей строке выведите одну строку, состоящую из нулей и единиц длины n, где i-ый символ равен единице, если i-ый игрок открыл свои карты, иначе — ноль.

Examples

stdin
3 20 2 50 100 120 AH AC KH KD JD JS JH KS KC AD AS 6 Player 3: Raise 20 Player 1: All in Player 2: Call Player 3: Raise 20 Player 2: Raise 25 Player 3: Call
standard input
Correct 210 95 25 110

stdin
3 20 2 50 100 120 AH AC KH KD JD JS JH KS KC AD AS 6 Player 3: Raise 20 Player 1: All in Player 2: Call Player 3: Raise 20 Player 2: Raise 25 Player 3: All in
standard input
Incorrect game log

stdin
3 20 2 50 100 120 AH AC KH KD JD JS JH KS KC AD AS 7 Player 3: Raise 20 Player 1: All in Player 2: Call Player 3: Raise 20 Player 1: All in Player 2: Raise 25 Player 3: Call
standard input
Incorrect game log

stdin
3 20 2 50 100 120 AH AC KH KD JD JS JH KS KC AD AS 6 Player 3: Raise 20 Player 1: All in Player 2: Call Player 3: Raise 20 Player 2: Raise 30 Player 3: Call
standard input
Incorrect game log

stdin
3 20 1 50 80 100 KH QD 2D 3D 7H 2H AS TH JS 4S 5H 8 Player 2: Check Player 3: Check Player 1: Raise 20 Player 2: Call Player 3: Raise 50 Player 1: All in Player 2: All in Player 3: Fold
standard input
Correct 210 50 30 110

Note

Рассмотрим последний пример: Играют три игрока. У первого 50 фишек, у второго — 80, у третьего — 100. Пусть первый игрок был дилером. Тогда торги начинает второй игрок. Размер *Blind* равен 20 фишек. Далее перечислены высказывания игроков начиная со второго:

1. Player 2: *Check*
2. Player 3: *Check*
3. Player 1: *Raise* 20
4. Player 2: *Call*
5. Player 3: *Raise* 50 (Игрок 3 поставил 70)
6. Player 1: *All in* (У игрока 1 недостаточно денег, он мог либо сбросить, либо сыграть "*All in*")
7. Player 2: *All in* (Игрок 2 уже добавлял 20. Теперь он согласен на еще 50 и хочет поднять еще. Но у него осталось денег меньше чем *Blind*, так что ему приходится идти "*All in*")
8. Player 3: *Fold*

Мы видим, что на этом торг закончится, так как игрок 1 уже поставил все свои 50 фишек. В игре игрока 2 и игрока 3 (все, что сверху 50) игрок 3 сбросил, то есть игрок 2 выиграл и забрал деньги третьего игрока сверху 50. При этом игрок 3 выбыл и из игры с первым игроком. Однако его деньги остались на кону (второй игрок выиграл только то, чтобыло сверху — 20 фишек). Итак на этом этапе банк равен 150, а второй игрок выиграл 20. Теперь если у первого игрока комбинация сильнее, чем у второго, то он выиграет 150, в противном случае второй игрок заберет весь банк.

Problem I. Цензура

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

HisSql — один из топовых стартапов в Сан Франциско. Как и любому другому стартапу, *HisSql* нужны хорошие инженеры. Поэтому был организован *His[C]up* — соревнование по программированию, где лучшие из лучших соревновались за главный приз.

Конечно, ни у кого в *HisSql* нет времени готовить *His[C]up*. Но программисты в *HisSql* очень находчивы. Вместо того, чтобы придумывать задачу, они решили использовать готовую нерешенную проблему по работе.

Проблема звучит следующим образом: *HisSql* поддерживает базу данных для различных слов, которая хранится в виде "бора". Для любого запроса, который приходит в *HisSql*, мы хотим узнать сколько раз этом запрос встречается в качестве подстроки в базе данных.

Например, строки “ladybug”, “ballad”, и “ladderlad” хранятся в базе данных. Ответом на запрос “lad” будет 4 раза (один раз в “ladybug”, один раз в “ballad” и дважды в “ladderlad”).

Для каждого запроса нужно вывести число раз, которое он встречается в качестве подстроки в базе данных.

Бор или префиксное дерево — абстрактный тип данных (АТД), структура данных, позволяющая хранить ассоциативный массив, ключами которого являются строки. В отличие от бинарных деревьев, в листьях дерева не хранится ключ. Значение ключа можно получить просмотром всех родительских узлов, каждый из которых хранит один или несколько символов алфавита. Корень дерева связан с пустой строкой. Таким образом, потомки узла имеют общий префикс, откуда и произошло название данного АТД. Значения, связанные с ключом, обычно не связаны с каждым узлом, а только с листьями и, возможно, некоторыми внутренними узлами.

Input

В первой строке записано целое число n ($1 \le n \le 10^6$) — число вершин в боре, не считая корня. Корень имеет индекс 0. Далее следует n строк. В i-ой строке содержится описание i-ой вершины:

- p ($0 \le p \le i - 1$) — родитель вершины;
- c (строчная буква латинского алфавита) — символ по которому мы пришли в текущую вершину из родителя;
- $t - 1$, если данная вершина является концом какого-либо слова, 0 в противном случае.

В следующей строке записано целое число q — число запрещенных слов. Общая сумма длин всех запрещенных слов не превосходит 10^6. Следующие q строк содержат запрещенные слова (по одному слову в строке). Все слова содержат только строчные буквы латинского алфавита, и являются непустыми. Все числа во входных данных целые.

Output

Для каждого запрещенного слова выведите в соответствующей строке число раз, которое это слово встречается в боре как подстрока.

Examples

standard input	standard output
9 0 a 0 1 c 0 2 m 1 1 l 0 4 e 0 5 x 1 0 b 0 7 r 0 8 o 1 3 lex flex ro	1 0 1
5 0 a 0 1 b 0 2 a 1 0 b 0 4 a 1 3 ba a ab	2 3 1

Note

Для первого примера “lex” является подстрокой “alex”; и “ro” — подстрока “bro”.

Problem J. Дождь

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Сан Франциско не очень солнечное место. Если кто-то сказал вам, что в Калифорнии все ходят в плавках и купальниках весь год, то он никогда не бывал в Сан Франциско. Но погода же не самое главное! В Сан Франциско мягкий климат, но зимой очень дождливо. Забавно, что в десяти милях от Сан Франциско солнечно и тепло! Все таки Сан Франциско — волшебное место. Волшебное, дождливое место.

В этой задаче будем считать, что Сан Франциско — это прямоугольный город размером $n \times m$, разделенный на участки 1×1. Каждый участок имеет какую-то определенную высоту (Сан Франциско — город на холмах). Океан находится сверху и снизу прямоугольной карты. Дождь заполняет ячейки последовательно, начиная с самых низких по высоте. Дождь заполняет высоту 1 за 1 минуту. Можно считать, что высота ячейки в любой момент времени равна собственной высоте ячейки плюс высоте уровня воды в этой ячейке. Дождь заполняет ячейки итеративно. Более формально, в каждую минуту водой заполняются все ячейки имеющие в данный момент времени наименьшую высоту.

Вам нужно найти момент времени, когда верх и низ карты соединятся водой. Каждая ячейка имеет не более 4 соседей: для ячейки (x, y) соседи $(x + 1, y)$, $(x - 1, y)$, $(x, y - 1)$, $(x, y + 1)$.

Input

В первой строке записаны два целых числа n, m $(1 \le n, m \le 1000)$ — число рядов и столбцов карты. В следующих n строках вам даны по m целых чисел — высоты соответствующих ячеек h_ij $(1 \le h_ij \le 10^9)$.

Output

Выведите единственное число — время в минутах когда океаны соединятся, то есть будет существовать путь сверху карты до низу только по воде.

Examples

standard input	standard output
3 5 102 101 102 10 102 102 100 102 1 102 102 101 102 10 102	10
3 4 1 9 9 9 1 1 5 1 9 9 9 1	5

Problem K. Мерцающие звезды

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

Однажды, прогуливаясь по одной из улиц Сан Франциско, профессор понял, что на небе нет ни одного облака. Для этого города такое явления было необычно. Звезды мигали и были великолепно видны. Профессор не мог упустить шанс доказать свою теорию "мигающих звезд". Согласно этой теории одна звезда может мигать несколько раз за свой период жизни. В такие моменты звезда производит огромное количество энергии. Профессор побежал в свою квартиру, чтобы установить необходимое оборудование и начать снимать важнейшие кадры в его жизни. Но когда он пришел в свою квартиру и проверил фотоаппарат, оказалось, что он израсходовал почти всю память на фотографии своих подруг. Ему совершенно не хотелось удалять эти снимки. У профессора остался только один кадр, и он должен стать лучшим.

Чтобы сделать лучший кадр необходимо установить три параметра: ширину кадра, высоту кадра и выдержку (продолжительность времени, когда линза будет открыта). Стороны кадра должны быть параллельны осям координат. Профессор хочет, чтобы на его снимке было как минимум k миганий. Все мигания, которые произойдут в период от начала съемки до ее конца включительно (длительность выдержки), а также мигания, происходящие на границе кадра, попадут на снимок. Если звезда мигнет несколько раз, все эти мигания будут считаться как независимые.

Помогите профессору найти минимально возможное значение площади кадра умноженной на величину выдержки, чтобы запечатлеть как минимум k миганий.

Input

В первой строке входного файла вам даны два целых числа n и k ($1 \le n, k \le 50$) — число звезд и минимальное число миганий, которое профессор хочет запечатлеть. В следующих $2n$ строках следует описание каждой звезды (две строки для каждой звезды). На первой из двух строк даны координаты звезды — целые числа x и y ($-10^5 \le x, y \le 10^5$), и число миганий этой звезды — целое число m ($1 \le m \le 50$). В следующей строке следуют времена миганий в миллисекундах — целые числа t ($1 \le t \le 10^5$). Суммарное количество миганий не превосходит 50. Никакие две звезды не расположены в одной точке.

Output

Выведите одно число: минимальное значение произведения ширины кадра на его высоту и на величину выдержки (в миллисекундах), чтобы в кадре оказалось не менее k миганий. Заметим, что каждый из множителей может быть равен нулю. Если не существует способа заснять k миганий, выведите -1.

Examples

standard input	standard output
4 5 0 0 2 1 2 1 1 1 1 1 2 1 1 5 100 1 1	500
3 5 10 10 2 1 2 0 10 2 1 2 10 0 2 2 1	100

Problem O. Запрос к чёрному ящику (supplementary for F)

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Перед вами вспомогательная задача для задачи F. Сдать её невозможно — чекер (являющийся чёрным ящиком) выводит «Presentation Error».

Далее следует описание Черного ящика, которое поможет вам понять формат и требования.

Во-первых, мы постарались сделать Черный ящик разумным. Это значит, что в нем не будет совершенно нелепых вещей.

Для того, чтобы упростить вашу жизнь, раз уж вы связались с этой задачей, чекер к задаче Q принимает набор запросов, произведенных программой, которую вы должны послать как решение к задаче Q. Число запросов не должно превышать 10. После отправки запроса Вам будет доступен отчёт по задаче (раздел «посылки», ссылка «посмотреть протокол»). В конце отчёта после текста "`Black box says:`" и пустой строки идёт вывод на Ваши запросы — ответы будут перечислены в том же порядке, в каком Ваша программа их отправляла.

Общее число попыток, которое Вы можете сделать по данной задаче, не должно превосходить 100.

Чтобы решить задачу F, вам нужно повторить логику Черного ящика согласно условий задачи F, то есть Ваше решение к задаче F должно принимать одну строку-запрос (а не код, производящий этот запрос!) и выдавать один ответ.

Input

У этой задачи нет входных данных.

Output

В первой строке выведите одно целое число n ($1 \le n \le 10$) — число запросов. В следующих n строках выведите по одному запросу к Черному ящику. Длина запросов не должна превышать 1000.

Examples

standard input
No input for the problem
standard output

7
bool Alex_Artem_Slava(int, long, short, float, double, void, char);
bool wish_you_good_luck(long long, long double, long int);
bool and_have_fun(unsigned int, unsigned long, unsigned short);
bool and_dont_solve_this_problem(unsigned float, unsigned double);
unsigned float read_other_problems_first(double, long, int, short);
bool if_you_solve_it_we_will_give_you_some_prize(int);
int glhf(unsigned void, unsigned char);