Задача A. Avengers, The

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мебибайта

После нападения читаури на Нью-Йорк команда Мстителей во главе с Капитаном Америкой решила обезопасить город силовыми столбами. Закончив строительство оборонительных сооружений, Тони Старк принялся за программирование столбов. Алгоритм, лежащий в основе новейшей системы безопасности довольно прост и эффективен. При появлении врагов в определённой точке выбирается некоторое множество столбов, образующих выпуклый многоугольник, содержащий эту точку. После чего между ними образуется силовое поле, уничтожающее всё живое внутри и на границе многоугольника. Для эффективной работы такого сооружения, внутри каждого столба требуется поместить источник огромной энергии. Поэтому, если какой-то из столбов окажется под прямым действием силового поля (строго внутри одного из многоугольников), произойдёт колоссальной силы взрыв, который уничтожит большую часть Земли. К аналогичным последствиям приведёт и пересечение двух различных силовых полей, т.е. полей образованных разными наборами столбов. При этом касание силовых полей не считается пересечением. Для обеспечения надёжности системы город должен находиться внутри прямоугольника, в вершинах которого стоят силовые столбы.

Для проверки эффективности системы команда решила потренироваться и провести симуляцию атаки читаури на город. Перед тренировкой выбирается количество атак читаури и точка первого удара. Кроме того, генерируются некоторые смещения $Shift^i$ для каждой из последующих атак. После этого все атаки моделируются следующим образом:

1. Определяется набор столбов, уничтожающих текущую цель.
2. Пусть точка O — среднее арифметическое координат выбранных столбов, округлённое к нулю, тогда очередная точка атаки вычисляется следующим образом:

$$x = R.x + (O.x + Shift^i_x) \mod Width$$
$$y = R.y + (O.y + Shift^i_y) \mod Height$$

   где R — координаты нижнего левого столба, $Width$, $Height$ — ширина и высота прямоугольника, окружающего город.
3. Эти действия повторяются до тех пор, пока все атаки не будут отражены

Формат входных данных

В первой строке задано число N — количество энергетических столбов ($4 \leqslant N \leqslant 10^4$).

В следующих N строках заданы числа X, Y — координаты соответствующего столба. Причем первые четыре точки образуют прямоугольник, ограничивающий зону симуляции атак, стороны которого параллельны осям координат. Точки прямоугольника заданы в порядке обхода по часовой стрелке, начиная с левой верхней. Гарантируется, что никакие три из N точек не лежат на одной прямой.

В следующей строке заданы 3 числа: K, X_0, Y_0 — количество атак читаури и координаты первого удара ($1 \leqslant K \leqslant 10^5$).

В следующих $K - 1$ строках заданы по два числа $Shift^i_x$, $Shift^i_y$ — смещение точки очередной атаки.

Все координаты — целые числа, не превосходящие 10^4 по абсолютному значению.

Формат выходных данных

Для каждой атаки программа должна вывести число $P \leqslant 10$ — количество точек в выпуклом многоугольнике, содержащем точку удара, и P чисел p_i — номера столбов, образующих этот многоугольник в порядке обхода по часовой стрелке.

Если ответов несколько, выведите любой.

Пример

standard input	standard output
5 -1 2 4 2 4 -3 -1 -3 0 0 3 -1 0 2 4 2 3	3 1 5 4 3 1 2 5 3 5 2 3

Задача B. Black Widow

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Наташа Романов, она же Чёрная Вдова, — легендарный агент Щ.И.Т. и непревзойдённый шпион. Она знает с десяток языков, владеет карате, айкидо, джиу-джитсу, самбо, сават, а также кунг-фу. Кроме того, Наташа обладает незаурядным умом и выдающимися актёрскими способностями, которые позволяют ей добывать информацию, даже когда её пытают.

Очередное задание Наташи — украсть описание нового секретного оружия Гидры. Ей не составило труда проникнуть в сейф через вентиляцию, однако выйти из него она может только через коридор, пол которого оснащён датчиками, реагирующими на давление. Конечно, в проекте коридора весь пол устлан датчиками. Но даже в Гидре воруют бюджетные средства! Поэтому на деле датчики представляют собой тонкие полосы от стены до стены, установленные в определённых точках коридора.

У Наташи, конечно же, есть план коридора, через который ей необходимо пройти. Поэтому ей известно расстояние до каждого из датчиков. Агент Романов хочет преодолеть этот коридор самым быстрым способом — цепочкой фляков. Однако, недостатком этого способа является тот факт, что расстояние между двумя последовательными касаниями пола (руками или ногами) на протяжении всего пути оказывается одинаковым (и выбирается спортсменом в момент первого прыжка).

Учитывая, что чем длиннее прыжок, тем больше сил на него тратится, вычислите, какое минимальное расстояние между двумя последовательными касаниями пола позволит Чёрной Вдове не попасться.

Формат входных данных

В первой строке задано число N — количество датчиков на полу в коридоре ($1 \leqslant N \leqslant 1000$ — кризис, сотрудникам тоже что-то есть надо, поэтому датчиков так мало). Во второй строке даны N чисел x_i — расстояния от точки, где первоначально стоит Наташа, до соответствующего датчика ($1 \leqslant X \leqslant 10^9$ — коридор, ведущий к суперсекретному оружию, должен быть длинным даже в кризис).

Формат выходных данных

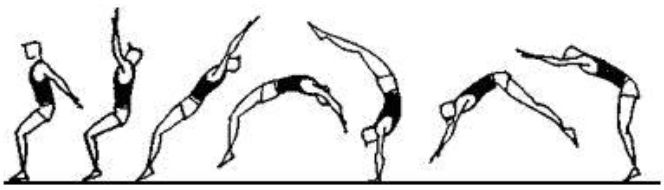
В единственной строке выведите одно целое число — минимальное расстояние между двумя последовательными касаниями пола.

Примеры

standard input	standard output
5 1 3 4 8 10	6
10 2 3 5 10 12 15 20 30 44 63	8

Замечание

Фляк — соединение двух прыжков: назад на руки и с рук на ноги



Задача C. Chitaury

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Как известно, читаури — воинственная раса, находящаяся на службе у Таноса. Поручения одного из величайших суперзлодеев могут забросить их в самые отдалённые уголки галактики. Неблагодарный труд, долгий путь, недоброжелательное местное население... В таких суровых условиях одна радость — мародёрство! Но и тут постоянно возникает проблема — как поделить награбленное? После многих испорченных выходных, потраченных на кровопролитные схватки за свою долю, на одной из планет читаури услышали про «демократию». Переварив это понятие, они пришли к следующей схеме.

1. Все награбленные трофеи собираются в кучу. В ней все предметы считаются равными.
2. Из всех ещё живых читаури выбирается самый старший и опытный, которому и поручается справедливо поделить награбленное.
3. Он предлагает некоторую схему дележа, которая выносится на обсуждение, после чего все читаури голосуют.
4. Если большинство (т.е. больше половины) читаури против данного предложения, то выдвинувшего его убивают, и делёжка начинается с пункта 2.
5. Если же противников этого варианта недостаточно, то делёж осуществляется в соответствии с предложенной схемой.

Заметим, что среди читаури нет двух, равных друг другу по старшинству и опыту.

Каждому читаури нелегко принять решение при голосовании, ведь ему нужно найти компромисс между всеми своими стремлениями. Читаури не дурак, поэтому важнейшим стимулом при принятии решения является желание выжить. Далее, поскольку все читаури являются страстными коллекционерами, они просто обожают собирать разные сувениры с захваченных планет. Поэтому следующим фактором, влияющим на решение, является количество трофеев, доставшихся самому читаури при дележе: чем больше, тем лучше. Кроме того, если от его выбора не зависит размер полученной доли, кровожадный читаури стремится убить как можно больше конкурентов. И напоследок, решая, как поделить добычу, читаури руководствуется ещё одним принципом: лучше не злить понапрасну опытных читаури. Соответственно, он выбирает вариант, при котором больше трофеев отходит старшим читаури.

К сожалению, процесс принятия решения оказался ужасно утомительным. А если читаури думают о чём-то слишком долго, то всё вновь заканчивается кровавой бойней. Помоги читаури принять решение как можно быстрее.

Формат входных данных

На вход подается два числа N и K — количество читаури в команде и количество награбленных трофеев соответственно ($1 \leqslant N \leqslant 1000, 0 \leqslant K \leqslant 1000$).

Формат выходных данных

Для каждого читаури, начиная с самого старшего, выведите одно число — количество трофеев, которое он получит, или -1, если он умрёт.

Примеры

standard input	standard output
2 1	1 0
3 0	-1 0 0

Задача D. Dr. Banner

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Доктор Беннер — выдающийся физик. Никто не знает о гамма-лучах так много, как он. К сожалению, у него бывают... ну эээ... вы знаете... "зелёные" проблемы. Что он только ни пробовал, чтобы справиться с ними! Выход оказался довольно простым. Чтобы погасить приступ нарастающего гнева, ему нужно сосредоточиться на чём-нибудь другом. Например, на построении пирамидок из детских кубиков.

Для построения очередной серии пирамидок Доктор Беннер сначала выбирает размер кубика в основании. Всё-таки он учёный и любит, чтобы все построенные им пирамидки имели одинаковые основания. После выбора основания процесс построения пирамидки предельно прост: нужно ставить очередной кубик на предыдущий. При этом, если предыдущий кубик имеет ребро размера K, то можно либо положить сверху кубик с ребром размера от 1 до $K/2$, либо остановиться и считать пирамидку построенной.

Беннер не любит повторяться, а значит, чтобы случайно не разозлиться, ему необходимо строить различные пирамидки.

Так как один из друзей Доктора Беннера — миллиардер Тони, в распоряжении Доктора есть неограниченное количество кубиков с ребром любой целой положительной длины.

Друзья очень любят Доктора Беннера, особенно, когда он незеленый. Они хотят понять, как долго можно находиться в безопасности рядом с Доктором при условии, что он справляется со строительством одной пирамидки за одну секунду.

Формат входных данных

На вход подается одно число N — размер основания пирамидки, выбранный доктором. ($1 \leqslant N \leqslant 10^5$)

Формат выходных данных

На выход программа должна выдать одно число — количество секунд, которое потребуется Доктору на построение всех возможных пирамидок. Ответ необходимо вывести по модулю числа $10^9 + 7$.

Примеры

standard input	standard output
1	1
2	2

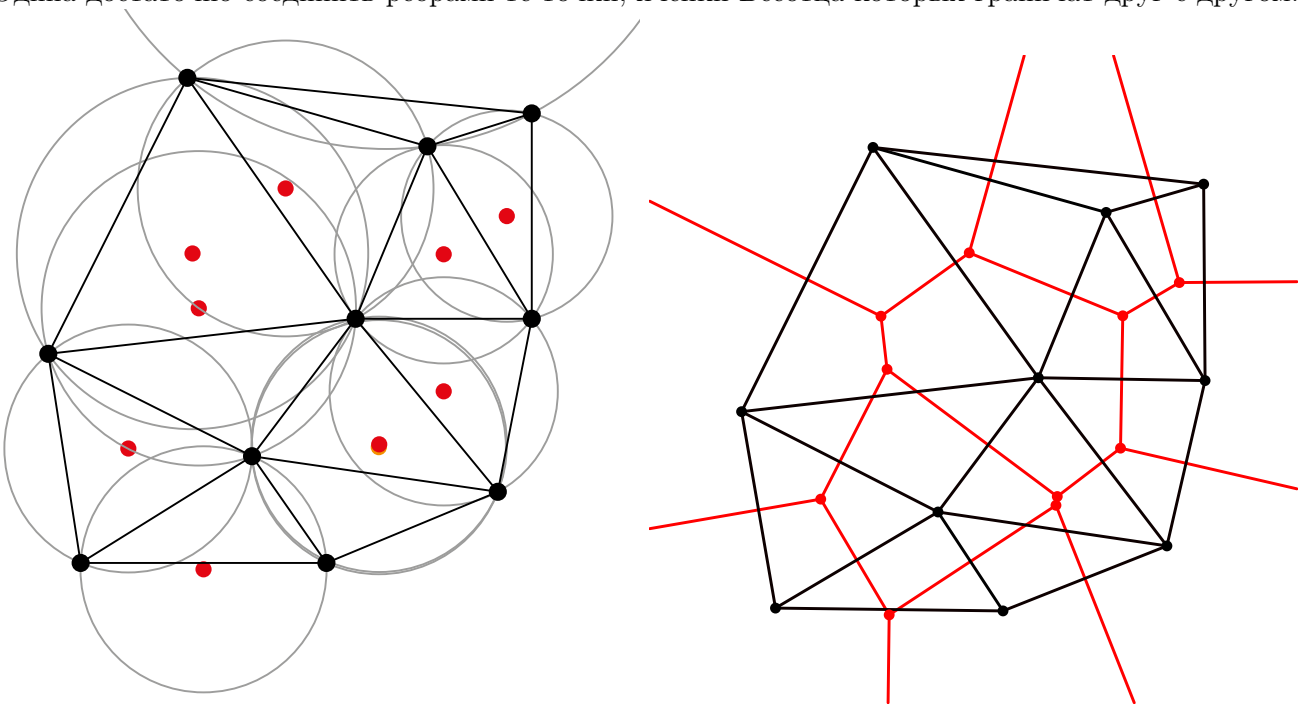
Задача E. Egocentric Loki

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Локи рос щуплым асгардцем. Он не мог сравниться с Тором в физической силе, упорстве и храбрости в сражениях. Недостаток качеств, ценимых в Асгарде в первую очередь, он компенсировал хитростью, ловкостью и умом, отчего вырос довольно самолюбивым богом.

С детства Локи очень любил решать различные геометрические задачи и, в особенности, задачу построения триангуляции. Один заметил это и научил его, как строить триангуляцию с полезными свойствами, которая в честь него была названа триангуляцией Одина.

На случай, если вы не учились в асгардской школе, напоминаем, что триангуляцией Одина для заданного набора точек на плоскости называется разбиение этого множества на треугольники, вершинами которых являются точки этого множества такое, что для каждого треугольника ни одна из точек множества не лежит строго внутри его описанной окружности. Двойственной к триангуляции Одина является диаграмма Всеотца, которая для заданного множества точек представляет собой разбиение плоскости на ячейки Всеотца таким образом, что каждая ячейка соответствует некоторому элементу исходного множества и состоит из всех точек, для которых этот элемент является ближайшим среди всех элементов множества. Известно, что есть однозначное соответствие между триангуляцией Одина и диаграммой Всеотца. Нетрудно заметить, что для построения триангуляции Одина достаточно соединить рёбрами те точки, ячейки Всеотца которых граничат друг с другом.



Многие годы для построения триангуляции множества из N точек Локи использовал следующий итеративный алгоритм со временем работы $O(N \log N)$:

1. Для трёх точек возможна единственная триангуляция: нужно попарно соединить точки отрезками.
2. Для четырёх точек возможны два варианта:
   (a) Если точки образуют невыпуклый четырёхугольник, то нужно соединить все 4 точки отрезками.
   (b) Если точки образуют выпуклый четырёхугольник, то берём любые 3 точки и проверяем положение четвёртой точки относительно окружности, описанной вокруг первых трёх точек. Здесь возможны три варианта:

i. Точка лежит за пределами окружности. Данная триангуляция оптимальна, строим треугольник из первых трёх точек и соединяем с четвёртой ближайшие к ней 2 точки.

ii. Точка лежит внутри окружности. В этом случае нарушается условие триангуляции Одина, и отрезками соединяется четвёртая точка со всеми остальными точками, а также те точки, отрезки между которыми не создадут пересечений с уже проведёнными отрезками.

iii. Точка лежит на окружности. В этом случае любая триангуляция оптимальна.

3. Если число точек больше четырёх, то исходное множество разбивается на два более мелких множества. Для этого нужно провести вертикальные или горизонтальные прямые в середине множества и уже относительно этих прямых разделить точки на две части примерно по $N/2$. После для каждой группы точек рекурсивно запустить процесс деления в зависимости от их количества:

   (a) Если число точек $N \geqslant 12$, то множество делится с помощью прямых.

   (b) Если число точек $N \leqslant 12$, то множество делится на 3 и $N - 3$ точек.

   (c) Если число точек $N = 8$, то множество делится на 2 группы по 4 точки. Деление продолжается до тех пор, пока не останется 3 или 4 точки.

4. Объединение оптимальных триангуляций. Сначала находятся две пары точек, отрезки которых образуют в совокупности с построенными триангуляциями выпуклую фигуру. Они соединяются отрезками, и один из полученных отрезков выбирается как начало для последующего обхода. Обход заключается в следующем: на этом отрезке мы как будто «надуваем пузырь» внутрь до первой точки, которую достигнет раздувающаяся окружность «пузыря». С найденной точкой соединяется та точка отрезка, которая не была с ней соединена. Полученный отрезок проверяется на пересечение с уже существующими отрезками триангуляции, и в случае пересечения они удаляются из триангуляции. После этого новый отрезок принимается за начало для нового «пузыря». Цикл повторяется до тех пор, пока начало не совпадёт со вторым отрезком выпуклой оболочки.

Эго Локи росло и со временем ему стала ненавистна мысль о том, что он до сих пор пользуется старым отцовским алгоритмом. И недавно он всё-таки смог придумать алгоритм, который строит триангуляцию Одина за время $O(N)$. Он страшно гордится этим и демонстрирует его работу при любой возможности, втайне надеясь, что триангуляцию Одина переименуют в триангуляцию Локи.

Вам предоставляется уникальный шанс уязвить самолюбие Локи.

Формат входных данных

В первой строке задано число N — количество треугольников в триангуляции Локи.

В следующих N строках заданы 6 чисел x_1, y_1, x_2, y_2, x_3, y_3 — координаты вершин очередного треугольника. Гарантируется, что данные треугольники образуют триангуляцию некоторого множества точек, размер которого не превосходит 1000, и что координаты всех точек — целые числа, по модулю не превосходящие 10^4.

Формат выходных данных

Выведите "`YES`", если триангуляция является триангуляцией Одина, иначе выведите "`NO`".

Примеры

standard input	standard output
1 0 0 1 0 0 1	YES
2 1 1 3 0 0 3 3 3 3 0 0 3	NO

Задача F. Fury

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Щ.И.Т — огромное международное агентство по борьбе с преступностью, причём не только в Солнечной системе, но и за её пределами. Компания, которая противопоставляет себя посланникам Таноса, имеет очень сложную псевдоиерархическую структуру. В рамках этой структуры сотрудник может ставить задачи своим непосредственным подчинённым (при этом отношение подчинения весьма произвольно — допустимы даже случаи, когда a является непосредственным подчинённым b и в то же время b является непосредственным подчинённым a). Кроме того, чтобы избежать излишней бюрократии, когда задача должна быть много раз перепоручена на пути к исполнителю, в Щ.И.Т. допускается ставить задачу напрямую.

К сожалению, после очередного крушения Летающего Авианосца$^{\text{TM}}$, база данных сотрудников организации была утеряна, а о бэкапах никто не позаботился. Таким образом вся информация о структуре организации была потеряна. Однако сохранились журналы, куда Ник Фьюри заставлял всех сотрудников записывать информацию обо всех поручениях. Таким образом для каждого агента удалось определить список тех, кто выполнял его поручения.

Теперь он просит вас помочь воссоздать структуру организации, которая бы наиболее соответствовала исходной. А именно, в построенной структуре должны быть возможны те и только те назначения задач, которые присутствуют в журнале, а также выводятся из них с учётом правил передачи задач. Кроме того, поскольку организация Щ.И.Т. очень большая, вас просят построить схему с минимальным количеством прямых связей подчинения между агентами.

Формат входных данных

В первой строке заданы два числа N, M — количество агентов в организации и количество записей в журнале соответственно ($1 \leqslant N \leqslant 300, 0 \leqslant M < N^2$).

Следующие M строк задают записи журнала. Каждая запись состоит из номеров двух агентов J и K, и обозначае постановку агентом J задачи для агента K. ($1 \leqslant J, K \leqslant N$, $J \neq K$)

Формат выходных данных

В первой строке выведите числа $P(P = N), Q(Q \leqslant M)$, — количество агентов и количество отношений в построенной схеме.

В следующих Q строках выведите номера агентов J и K, обозначающих, что агент K является непосредственным подчинённым агента J.

Пример

standard input	standard output
7 11 1 2 2 1 3 6 6 3 4 5 5 4 1 3 1 4 2 5 2 6 3 7	7 9 1 2 2 1 3 6 6 3 4 5 5 4 2 3 1 5 6 7

Задача G. Groot

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Я есть Грут.

Формат входных данных

Я есть Грут.

Формат выходных данных

Я есть Грут.

Примеры

standard input	standard output
I am Groot	Pfff
I am Groot!	Wow
I am Groot!!!!!!	Wooooooow

Задача H. Heimdall

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мебибайта

Хеймдалль — один из величайших сынов Асгарда, страж богов и мирового древа. С древнейших времён его важнейшая задача — охранять вход в Асгард, а также Биврёст — мост между мирами.

Биврёст — мистический механизм, позволяющий перемещать материю между мирами. Самое удивительное его свойство состоит в том, что все перемещения чудесным образом сохраняют баланс во вселенной. Если в какой-то момент часть материи перетекает из одного мира в другой, то возникающая в первом мире пустота тут же заполняется материей из некоторого другого уголка вселенной. И наоборот, если в какой-то из миров переносится материя, этот мир тут же отдаёт часть своей материи другому миру.

К сожалению, технологии создания и настройки подобных мостов давно утеряны, поэтому для каждого мира строго фиксирован тот мир, куда мост перемещает материю. Единственная уцелевшая на сегодняшний день древняя технология позволяет создавать мосты, проходящие сквозь миры, объединяя несколько мостов в один. Например для двух мостов объединение работает следующим образом: если первый мост позволяет переместить материю из мира в мир B, а второй — из мира B в мир C, то мост, полученный путём их объединения, позволяет перемещать материю напрямую из мира A в мир C. Более того, эта технология настолько удивительна, что даже позволяет объединять мост с самим собой.

За века верной службы миру асов и безмятежного созерцания просторов вселенной Хеймдалль смог проникнуть своим острым взором в самые дальние её уголки. Он обнаружил, что помимо Биврёста, во вселенной сохранилось ещё несколько подобных мостов. И теперь ему интересно, сколько различных мостов можно получить с помощью их объединения.

Формат входных данных

В первой строке даны целые числа R, N — количество мостов обнаруженных Хеймдаллем и число миров во вселенной соответственно. ($1 \leqslant N \leqslant 15, 1 \leqslant R \leqslant 1000$)

Следующие R строк содержат описания мостов. Описание каждого моста состоит из N целых чисел $a_1, a_2, \ldots a_n$, где a_i соответствует номеру мира, в который этот мост перемещает материю из мира с номером i. Если же мост не влияет на этот мир, то $a_i = i$.

Формат выходных данных

Выведите одно число — количество различных мостов, которые можно построить, используя древнюю технологию.

Пример

standard input	standard output
2 5 2 1 3 4 5 2 3 4 5 1	120

Задача I. Iron Man

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1.5 секунды
Ограничение по памяти:	64 мебибайта

Даже у гениев, миллиардеров и филантропов бывают плохие дни. Только наступили выходные и Тони собрался наконец доказать, что $P = NP$, как созданный им искусственный интеллект Альтрон захватил Джарвиса. Но не останавливать же работу Stark Industries! Придётся Тони арендовать несколько облачных серверов.

Для ускорения вычислений Тони разделил все имеющиеся задачи на K типов и выбрал N серверов. Для каждого сервера известно подмножество типов задач, с которыми он в принципе может справиться, и стоимости выполнения задачи каждого допустимого типа. Поскольку задачи разных типов очень сильно отличаются, каждый сервер можно настроить на выполнение только одного типа задач. Чтобы изменить конфигурацию сервера, нужно обратиться к провайдеру. За одно обращение можно изменить конфигурацию любого количества серверов. Каждый такой запрос стоит денег и занимает некоторое время, поэтому перенастраивать сервера можно не чаще одного раза в день.

По техническим причинам, конфигурацию серверов можно осуществить только перед выполнением задач на текущий день — во время утреннего обслуживания серверов.

Тони составил план вычислений на несколько дней. Для каждого дня известно, сколько задач каждого типа нужно выполнить. Только Тони хотел заключить договор с провайдером, как в дело вмешалась Пеппер и потребовала, чтобы он выбрал такой план, при котором нужно заплатить как можно меньше. Ну не стой же столбом! Помоги Железному человеку!

Формат входных данных

В первой строке заданы три целых числа N, K, C — количество серверов, количество типов задач и стоимость запроса о перенастройке серверов ($1 \leqslant K \leqslant N \leqslant 15, 0 \leqslant C \leqslant 10^5$).

В следующей строке дано целое число M ($1 \leqslant M \leqslant N \cdot K$). Далее в M строках заданы по 3 целых числа S, T, W, где W определяет стоимость выполнения одной задачи типа T на сервере S ($1 \leqslant S \leqslant N$, $1 \leqslant T \leqslant K$, $1 \leqslant W \leqslant 1000$, для любых двух из M строк или S, или T различаются).

Далее дано целое число Q — количество дней, на которые Тони распланировал вычисления ($1 \leqslant Q \leqslant 100$).

Каждая из следующих Q строк состоит из K целых чисел $a_1, a_2, \ldots a_k$, где a_i — это количество задач типа i, которые нужно выполнить в этот день. $\left(0 \leqslant a_i \leqslant 100, \sum_{i=1}^{K} a_i > 0\right)$

Гарантируется, что решение всегда существует.

Формат выходных данных

Выведите единственное число — минимальную возможную стоимость выполнения всех задач, с учётом стоимости настройки серверов.

Пример

standard input	standard output
2 2 6 4 1 1 2 1 2 1 2 1 4 2 2 2 2 1 1 1 10	25

Задача J. Jarvis

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Наладив работу кластера Stark Industries, Тони приступил к починке Джарвиса. Сначала Старк думал, что атака Альтрона была нацелена на код его помощника. Просидев за дебагом всю ночь, Тони пришёл к выводу, что программа осталось нетронутой. Ведь Джарвис, хоть и говорит человеческим голосом, всё-таки полностью зависит от серверов, на которых запущен.

Проверяя сеть серверов, Тони обнаружил, что Джарвис в попытках самосовершенствования усложнил алгоритмы обмена и синхронизации данных между узлами сети. Но он не учёл такую немаловажную характеристику, как загруженность сервера.

В конце концов, Тони понял, что для балансировки нагрузки ему необходимо менять задержку передачи данных между некоторыми парами узлов. Но так как Джарвис использует очень сложный протокол маршрутизации, Тони не может предугадать путь, по которому будут передаваться данные. Поэтому Тони пришёл к простому решению этой проблемы — внести изменения в сеть требуется таким образом, чтобы изменить задержку на нужную величину для *любого* маршрута из бесконечного количества возможных (не обязательно простых!).

Гений, миллиардер, плейбой и филантроп выделил ключевые пары узлов, между которыми надо скорректировать задержку. Помогите ему посчитать новую карту сети, удовлетворяющую всем требованиям. Заметим, что так как Тони использует тахионную сеть, то в сети возможны и *отрицательные* задержки.

Формат входных данных

В первой строке задано целое число $N(1 \leqslant N \leqslant 150)$ — количество серверов в кластере Джарвиса.

Далее следуют N строк по N целых чисел $a_{ij}(0 \leqslant a_{ij} \leqslant 1000)$ — исходная матрица задержек между серверами. При этом a_{ij} не обязательно равно a_{ji}. Гарантируется, что $a_{ii} = 0$.

В следующей строке задано целое число $M(1 \leqslant M \leqslant 150)$ — количество пар ключевых узлов. Далее в M строках заданы целые числа S, T, D, $(1 \leqslant S, T \leqslant N, -1000 \leqslant W \leqslant 1000)$ описывающие запрос на изменение задержки от сервера S до сервера T на величину D.

Никакая пара S, T не встречается дважды. И все запросы должны быть выполнены одновременно.

Формат выходных данных

Выведите N строк по N целых чисел b_{ij} — матрицу задержек в итоговой сети. Значения b_{ij} не должны превышать 10^9 по абсолютной величине. Если же матрицу, удовлетворяющую всем требованиям составить нельзя, выведите "`Impossible`".

Примеры

standard input	standard output
2 0 1 1 0 1 1 2 2	0 3 -1 0
2 0 1 1 0 2 1 2 1 2 1 1	Impossible

Замечание

Если в первом примере Джарвис использует путь $1 \to 2$, то до коррекции задержек передача

данных занимает 1 секунду, а после неё займёт 3. Если Джарвис выбрал путь $1 \to 2 \to 1 \to 2$, то первоначально путь занимал $1 + 1 + 1 = 3$ секунды, а после коррекции займёт $3 - 1 + 3 = 5$ секунд, и так далее.

Задача K. KSON

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Агентство Щ.И.Т. — огромная организация с базами по всему миру. Как и любая другая организация такой величины, она начала вязнуть в бюрократии. Зарплата, командировочные шпионам, покупка полония, печенье сотрудникам — просто голова идёт кругом!

Чтобы избавиться от кипы бумаг, было решено перейти на электронные таблицы. Для их генерации используется Killing Spree Object Notation. KSON-объект представляет собой набор пар ключ-значение, в котором ключом является строка, а значением — строка или другой KSON-объект.

По техническим причинам все строки в KSON заключены в двойные кавычки. Строки, являющиеся ключами, не могут иметь ведущих и конечных пробелов и ограничены длиной в десять символов. Все пары ключ-значение разделены запятыми, а между ключом и значением ставится двоеточие. Более того, для удобства чтения между любыми значимыми элементами языка может находится любое количество пробелов и знаков перевода строки. Для возможности быстрого доступа все ключи любого KSON-объекта уникальны.

Данные, хранящиеся в KSON, можно представить в виде дерева, в котором промежуточными вершинами являются объекты, а листьями — строки-значения или пустые объекты. Ребра в этом дереве ведут из промежуточных вершин в значения ключей, хранящихся в соответствующем объекте. Таким образом, корнем дерева является весь KSON.

Таблица генерируется по таблице-шаблону, в котором каждая из ячеек представляет собой либо текст, либо ссылку на данные KSON. Ссылка выглядит как строка, начинающаяся с "`Value:`", после которой через точку указаны ключи. Этот список ключей указывает путь от корня KSON к некоторому листу, хранящему строку-значение. Перед ссылкой допускаются ведущие пробелы. Если ссылка является корректной, то она заменяется на значение листа, иначе текст остаётся неизменным.

После генерации таблицы её нужно вывести в том же формате, что и шаблон, не забыв, при необходимости, изменить размеры столбцов, чтобы вмещать новое содержимое. Итоговая ширина столбца должна соответствовать максимальной длине значения, среди всех значений этого столбца.

А теперь проверим, как вы поняли устройство электронных таблиц Агенства Щ.И.Т. Сгенерируйте пару-тройку таблиц по данным KSON и шаблону.

Формат входных данных

В первых строках дан KSON с данными, за ним, начиная с новой строки — таблица-шаблон. Все строки и все ячейки таблицы содержат только строчные и заглавные латинские буквы, цифры, пробелы, точки и двоеточия. Кроме того, строки-ключи KSON не содержат двоеточий и точек. Размер входных данных не превосходит 100 000 символов. Формат таблицы-шаблона смотрите в примере.

Формат выходных данных

Выведите заполненную таблицу в формате, аналогичном таблице-шаблону из входных данных.

Пример

standard input

```
{
    "Ural" : {
        "State" : " University ",
        "is" : "forever",
        "in" : "our"
    },
    "hearts" : " "
}
```

```
+-------------+------------------------+----------------------+--+
| Static      |Value:Ural.State        |Value:hearts          |  |
+-------------+------------------------+----------------------+--+
|   text      |Value:type              | Value:Ural.is        |  |
+-------------+------------------------+----------------------+--+
|Value:Ural.in|                        |Value:Ural.in         |  |
+-------------+------------------------+----------------------+--+
```

standard output

```
+-------------+------------------------+-------+--+
| Static      | University             |       |  |
+-------------+------------------------+-------+--+
|   text      |Value:type              |forever|  |
+-------------+------------------------+-------+--+
|our          |                        |our    |  |
+-------------+------------------------+-------+--+
```