Problem 2. Primes Pattern

Input file:	patprim.in
Output file:	patprim.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Research Institute of Primes (R.I.P.) gives Vasya another problem — for given K even positive integers a_i, find N smallest primes p_j such that $p_j + a_i$ is also prime for all $i : 1 \le i \le K$.

Input

First line contains two integers K ($1 \le K \le 15$) and N ($1 \le N \le 100$) separated by single space.

Each of next K lines contains single integer a_i ($2 \le a_i \le 30$). Numbers a_i are given in increasing order ($a_{i-1} < a_i$ for all $1 < i \le K$).

Output

Output should contain N lines. The j-th line should contain single integer p_j. You may safely assume that there is always at least one solution such that $p_j < 2^{31} - a_K$.

Examples

patprim.in	patprim.out
1 5 2	3 5 11 17 29
2 1 2 4	3

Problem 3. Гости с Тау Кита

Input file:	tau1.in
Output file:	tau1.out
Time limit:	2 секунды
Memory limit:	256 мегабайт

Организаторы Кубка Векуа были изрядно удивлены, когда среди заявок на участие в соревнованиях оказалась заявка команды с Тау Кита. Однако заявка была оформлена по всем правилам, и таукитяне были включены в списки. В Батуми они собирались прибыть на своём космическом корабле. Для коммуникации с приближающимся кораблём таукитяне предложили установить стационарный лазер и использовать некий аналог азбуки Морзе.

Однако в день прибытия возникла неожиданная сложность: на небе появились плотные облака, которые, проходя над установкой, закрывали луч лазера. Для того, чтобы сделать поправку на вызванные облаками ошибки, гости подали срочный запрос в жюри соревнований: какое наибольшее число раз луч лазера будет закрыт? К сожалению, жюри не знает место установки лазера — это дело оргкомитета, да и из прогноза погоды известно только, что ветер будет дуть с постоянной скоростью. Но положение облаков на небе жюри определить может. Поэтому было принято решение сообщить наибольшее для всех возможных расположений лазера и направлений ветра количество закрытий луча. Вам поручено написать программу, вычисляющую это количество.

Для упрощения задачи поверхность земли считается плоской, проекция каждого облака на землю представляется в виде многоугольника с целыми вершинами, все проекции считаются попарно непересекающимися, а ветер с постоянной скоростью смещает все имеющиеся облака вдоль некоторого вектора параллельно поверхности земли. Лазер представляется точкой на плоскости. Считается, что облако закрывает лазер, если точка, его представляющая, лежит на границе или внутри проекции облака.

Input

В первой строке входного файла записано число n — количество облаков на небе. Далее идёт n строк, описывающих проекции отдельных облаков: в i-й строке сначала идёт количество $n_i > 3$ вершин в многоугольнике, представляющем i-е облако, затем $2n_i$ координат этих вершин $-10^9 \le x_j,\ y_j \le 10^9$. При этом суммарное количество вершин многоугольников для всех облаков, заданных во входном файле не превышает 2000.

Output

Одно число k - максимальное количество перебоев в видимости лазера, вызванных облаками.

В случае, показанном в примере, максимальный ответ 3 достигается, например, при размещении лазера в точке (0,4) и ветре, направленном вдоль вектора [-1,0];

Example

tau1.in	tau1.out
2 5 1 1 5 1 5 4 3 2 1 4 3 2 4 3 5 4 4	3

Problem 5. Таблицы Руматы Эсторского

Input file:	`table.in`
Output file:	`table.out`
Time limit:	2 секунды
Memory limit:	256 мегабайт

Часто финалы TopCoder проходят в Лас-Вегасе. В качестве ознакомления с местной спецификой на этот раз участникам было предложено сыграть в покер на игровом автомате. Один из участников заявил, что у него с собой есть таблицы, которые по пришедшему раскладу определяют, какое количество карт необходимо поменять для максимизации ожидаемого выигрыша (то есть для максимизации суммы по всем исходам произведений выигрыша в каждом исходе на вероятность данного исхода). Таблицы эти приписываются легендарному Румате Эсторскому, который с их помощью регулярно обыгрывал других благородных донов. Проверить огромные таблицы «на глазок» трудно даже участнику онсайт-раунда TopCoder, а вот подсчитать количество раскладов, в которых необходимо менять определённое число карт, вполне возможно. Именно этим Вам и предстоит заняться.

Установленные в данном казино автоматы устроены следующим образом.

Используется стандартная колода из 52 карт (4 масти по 13 карт в каждой масти, внутри каждой масти карты по достоинству расположены в порядке возрастания следующим образом: 2,3,4,5,6,7,8,9,10,валет,дама,король,туз).

Сначала на руки игроку случайным образом сдаётся расклад из 5 карт. Игрок может поменять от 0 до 5 карт на соответствующее количество карт, которые выбираются случайным образом из оставшейся колоды. Полученный в результате расклад оценивается так:

- Если в раскладе есть 2 валета, 2 дамы, 2 короля или 2 туза, то выплачивается выигрыш за «пару».
- Если в раскладе есть 2 пары карт одного достоинства (например, две двойки и две тройки), то выплачивается выигрыш за «две пары».
- Если в раскладе есть 3 карты одного достоинства (например, три девятки), то выплачивается выигрыш за «тройку».
- Если в раскладе 5 карт, идущих подряд, например: (2 червей, 3 треф, 4 червей, 5 пик, 6 бубен) или (8 пик, 9 пик, 10 пик, валет червей, дама треф), то выплачивается выигрыш за «стрит». При этом тузы могут считаться либо как карта, идущая за королём (как обычно), либо как карта, идущая перед двойкой, но не одновременно (то есть «циклического» стрита не бывает).
- Если в раскладе все 5 карт принадлежат к одной масти (например, тройка, семёрка, туз, валет и дама пик), то выплачивается выигрыш за «флеш».
- Если в раскладе 3 карты одного достоинства и 2 карты другого (например, три валета и две тройки), то выплачивается выигрыш за «фулл хаус».
- Если в раскладе 4 карты одного достоинства (например, все четыре туза), то выплачивается выигрыш за «каре».

- Если в раскладе выполняются условия для «стрита» и «флеша» (например, 7 пик, 8 пик, 9 пик, 10 пик, валет пик), то выплачивается выигрыш за «флеш-стрит».
- Если расклад состоит из туза, короля, дамы, валета и десятки одной масти, то выплачивается выигрыш за «флеш-рояль».

При этом всегда выплачивается только один выигрыш — за наиболее дальнюю в списке комбинацию (например, при трёх двойках и двух тузах выплачивается только выигрыш за «фулл хаус», но не выигрыш за «тройку», «пару» или «две пары»).

Если в раскладе нет ни одной из присутствующих в списке комбинаций, то выигрыш не выплачивается.

Автоматы различаются системами выплат — векторами из 9 целых чисел, задающими коэффициент выигрыша за «пару», «две пары», «тройку», «стрит», «флеш», «фулл хаус», «каре», «флеш-стрит» и «флеш-рояль». Всего бывает 11 типов автоматов: с векторами (1,2,3,4,6,9,25,50,800), (1,2,3,4,6,9,25,50,940), (1,2,3,4,5,9,25,50,800), (1,2,3,4,6,8,25,50,800), (1,2,3,4,5,8,25,50,800), (1,2,3,4,5,7,25,50,800), (1,2,3,4,5,6,25,50,800), (1,2,2,4,6,9,30,125,1000), (1,2,2,4,6,9,30,100,1000), (1,2,3,4,5,6,25,50,1000), (1,2,2,4,6,9,30,100,500).

Ваша задача — по заданной системе выплат (одной из перечисленных 11) вывести количество раскладов, в которых наибольший ожидаемый выигрыш достигается без обмена карт, с обменом одной карты, с обменом двух карт, с обменом трёх карт, с обменом четырёх карт, с обменом пяти карт. Если в некотором раскладе для двух случаев с разным числом меняемых карт наибольший ожидаемый выигрыш одинаков, то выбирается вариант с меньшим числом оставляемых себе карт.

Input

Во входном файле заданы 9 целых чисел, задающих коэффициенты за выигрыш по комбинациям в соответствии с условием задачи. Гарантируется, что наборы чисел будут соответствовать одному из описанных в условии типов автоматов.

Output

Шесть целых чисел, обозначающих количество раскладов, в которых для достижения наибольшего ожидаемого выигрыша необходимо менять 0 карт, менять одну карту, менять 2 карты, менять 3 карты, менять 4 карты и менять все 5 карт соответственно.

Example

table.in	table.out
1 2 3 4 6 9 25 50 800	18864 292800 147528 1651440 403968 84360

Problem 7. Sokoban

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 megabytes

For the period of his vacation, programmer Stas found a job with the Japanese computer company *Thinking Rabbit*. At first glance, the idea seemed marvelous: he would go abroad, earn some money, and learn from his Japanese colleagues. However, it turned out that the company did not want programmers without good knowledge of Japanese. Therefore, Stas was sent to work as a storekeeper (in Japanese, this profession was called *soko-ban*).

Stas had to put the storehouse to order. Every morning he was given a sheet of paper with a scheme of the room in the storehouse where he had to work that day. The scheme showed the places where he had to put containers. For some reason, the management of the company did not bother about which container would be put to which place; they only wanted all containers to be put to the places marked on the scheme.

The task was not easy. The containers were large and heavy; it was only possible to move them by pushing along the floor, and they were too heavy to push more than one of them at a time. In addition, the containers were so smooth that Stas could not pull or turn them; all he could do was to push them forward in front of him. The dimensions of the room corresponded to the size of containers exactly, so Stas could not clime over a container, squeeze himself between containers, or wriggle himself between a container and a wall. He could only move through unoccupied space. Thus, putting containers in order was a tricky puzzle. And if Stas could not solve it or put incidentally one of the containers into some corner from which it could not be extracted, then Stas was in real trouble. The point was that the walls of the room were solid, with no exits. In the morning, Stas got to the room through one of the hatches in the ceiling. He could not leave the room until the task was completed. When all containers were on their places, the smart control system opened a hatch with a rope-ladder for Stas right over him.

Help Stas to make a plan of moving the containers.

Input

You are given a scheme of the storeroom. This is a table of size $n \times m$ ($3 \le n, m \le 8$). An empty cell is shown by a space, and objects are denoted as follows:

- `#` is a piece of wall
- `.` is an empty cell where a container must be put (an aim cell)
- `@` is the cell from which Stas starts his work if it is not an aim cell
- `+` is the cell from which Stas starts his work if it is an aim cell
- `$` is a container on a cell which is not an aim cell
- `*` is a container on an aim cell

It is guaranteed that the scheme of the room is correct, that is, Stas cannot go out of the room. The number of containers is equal to the number of aim cells.

Output

Output a plan of work for Stas. In a single line, you should specify his movements by letters `r`, `l`, `u`, and `d`, which correspond to the four possible directions of moves. If during a move a container is pushed, then the letters should be capital (`R`, `L`, `U`, and `D`, respectively). The string should be no longer than 10000 symbols. You may assume that there is a solution.

Examples

input.txt	output.txt
######## #@ $.# ########	rrRR
###### ## .# #@ ### # * # # $ # # # #######	dddrrrruLdlUUUluRR

Problem 11. Инцидент с грузовым кораблём

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	6 секунд
Memory limit:	64 мегабайт

«Тахмасиб» принял сигнал SOS. Быков тут же изменил курс корабля, и через несколько часов планетолёт оказался на месте бедствия. Оказалось, что один из кораблей, перевозивший на Луну потребовавшиеся для каких-то научных экспериментов брёвна из специально выведенных на Земле сортов дерева, столкнулся с проблемами, связанными с нарушением центровки.

После того, как ситуацию удалось стабилизировать, Генеральный Инспектор Юрковский решил разобраться в причине инцидента и вместе с бортинженером Жилиным проследовал на борт корабля. Оказалось, что технология загрузки брёвен в бункер была нарушена на Земле — брёвна просто сбрасывались в бункер. Так как длина бревна соответствует длине бункера, то оси брёвен расположены параллельно дну бункера и его боковой стенке, и при моделирования процесса на экране монитора можно обойтись проекциями распилов брёвен на переднюю стенку бункера в виде кругов. Выяснилось, что в процессе загрузки, зафиксированном бортовым компьютером, бревна падали строго по вертикали (считаем, что брёвна имеют бесконечно большой коэффициент трения). После того, как бревно коснется дна бункера или ранее упавших бревен, или бревна и стенки бункера, оно перекатывается и падает до тех пор, пока не займет устойчивое положение, то есть, пока его центр не будет проецироваться между точками его касания. При этом расположение ранее упавших бревен не меняется. Если диаметр падающего бревна меньше либо равен расстоянию между двумя соседними неподвижными бревнами, то это бревно не застрянет между ними, а будет продолжать падать.

В случае, если бревно имеет только одну точку касания, его положение будет устойчивым, если оно касается дна бункера, и неустойчивым, если оно касается другого бревна. В последнем случае гарантируется, что ситуация, когда центр падающего бревна располагается строго над центром лежащего, является невозможной.

Требуется по записи процедуры погрузки восстановить окончательное расположение брёвен в бункере.

Input

В первой строке входного файла записаны через пробел два целых числа — S и K, где S — ширина бункера, K — количество брёвен ($1 \leqslant S < 1000, 1 \leqslant K \leqslant 200$). В последующих K строках приведено по два вещественных числа: R_i — радиус i-го падающего бревна ($10^{-1} \leqslant R_i < 500$) , и расстояние L_i от его центра до левого края бункера. При этом $R_i \leqslant L_i \leqslant S - R_i$. Высота бункера считается бесконечной.

Output

В выходной файл нужно выдать K строк, в каждой строке должно быть записано через пробел по два вещественных числа, округленных до 7 знаков после десятичной точки — соответственно координаты X и Y всех бревен в том же порядке, в каком они перечислены во входном файле, начиная с первого. Начало координат совмещено с левым нижним краем бункера, ось OX направлена вправо.

Example

input.txt	output.txt
12 3 3 5 2 4 1 6	5.0000000 3.0000000 2.0000000 7.0000000 9.0000000 1.0000000

Problem 13. Выражение

Input file:	`expr.in`
Output file:	`expr.out`
Time limit:	2 секунды
Memory limit:	256 мегабайт

Петя — большой любитель математических головоломок. Недавно он прочитал в одном популярном журнале о новой головоломке. Он пытался её решить несколько дней, но это ему так и не удалось. Помогите Пете справиться с неподдающейся задачей

В ряд выписаны n чисел. Требуется поставить между каждой парой соседних чисел один из знаков «+» и «×» таким образом, чтобы значение получившегося выражения было как можно больше. Использовать скобки не разрешается.

Например, для последовательности чисел $1, 2, 3, 1, 2, 3$ оптимально расставить знаки следующим образом: $1 + 2 \times 3 \times 1 \times 2 \times 3$. Значение выражения в этом случае равно 37.

Input

Первая строка входного файла содержит число $n (2 \leqslant n \leqslant 200000)$. Вторая строка содержит n целых чисел — числа, между которыми следует расставить знаки. Все числа находятся в диапазоне от 0 до 10^9.

Output

Выведите в выходной файл оптимальное выражение. В качестве знака умножения выводите символ «*» (звёздочку). Если оптимальных выражений несколько, выведите любое из них.

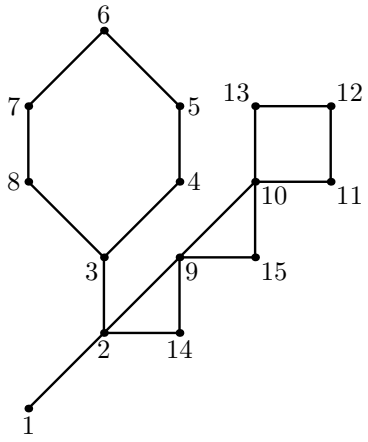
Example

expr.in	expr.out
6 1 2 3 1 2 3	1+2*3*1*2*3

Problem 17. Cactus Reloaded

Input file:	`cactus.in`
Output file:	`cactus.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Cactus is a connected undirected graph in which every edge lies on at most one simple cycle. Intuitively cactus is a generalization of a tree where some cycles are allowed. Your task is to find a *diameter* of the given cactus. Diameter is the maximal length of the shortest path between pairs of vertices.



For example, on the picture above the shortest path between vertices 6 and 12 goes through 8 edges and it is the maximal shortest path in this graph, thus its diameter is 8.

Input

The first line of the input file contains two integer numbers n and m ($1 \le n \le 50\,000$, $0 \le m \le 10\,000$). Here n is the number of vertices in the graph. Vertices are numbered from 1 to n. Edges of the graph are represented by a set of edge-distinct paths, where m is the number of such paths.

Each of the following m lines contains a path in the graph. A path starts with an integer number k_i ($2 \le k_i \le 1000$) followed by k_i integers from 1 to n. These k_i integers represent vertices of a path. Adjacent vertices in a path are distinct. Path can go to the same vertex multiple times, but every edge is traversed exactly once in the whole input file. There are no multiedges in the graph (there is at most one edge between any two vertices).

The graph in the input file is a cactus.

Output

Write to the output file a single integer number — the diameter of the given cactus.

Example

cactus.in	cactus.out
15 3 9 1 2 3 4 5 6 7 8 3 7 2 9 10 11 12 13 10 5 2 14 9 15 10	8

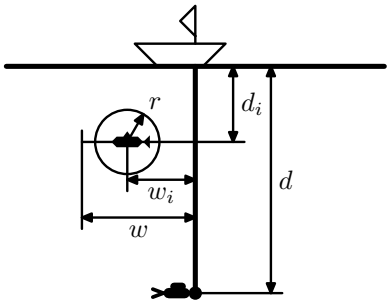
Problem 19. Diver

Input file:	`diver.in`
Output file:	`diver.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Diver had just completed her mission in the depths of the ocean and needs to resurface. To get to the surface she must use the rope that goes straight down from her boat on the surface to her location d feet under the water. However, while she was working, several sharks gathered near the rope. They do not consider her a danger or a prey yet, but if she gets closer than r feet to a shark, then it immediately attacks her.

To avoid decompression sickness diver can descend (go down) or ascend (go up) at most v_d feet per second. She also cannot go deeper than d feet under the water.

Each shark swims at its own constant depth of d_i feet near the rope. Speed and the pattern of movement for all sharks is the same. They cannot just stay in the water near the rope. They have to swim to avoid sinking, so they swim in a back-and-forth motion with a constant speed of v_s — swimming away from the rope on distance of w feet and swimming back to the rope again. Sharks change the direction of their movement so fast, that we consider it being instantaneous. When a shark attacks the diver it also moves so fast, that we consider it to happen instantaneously as soon as the diver is inside a circle of r feet in radius around a shark.



Your task is to figure out if the diver can get to the surface without being attacked by a shark, and if yes, then how fast she can do it.

Input

The first line of the input file contains 6 integer numbers:

- d $(10 \le d \le 100)$ — initial depth of the diver.
- v_d $(1 \le v_d \le 10)$ — maximal speed of the diver.
- n $(1 \le n \le 20)$ — number of sharks.
- r $(1 \le r \le 10)$ — minimal safe distance between a shark and the diver.
- w $(10 \le w \le 100)$ — maximal distance that a shark swims away from the rope.
- v_s $(1 \le v_s \le 50)$ — speed of a shark.

Then follow n lines describing sharks with 3 integer numbers per line for each shark:

- d_i $(1 \le d_i < d)$ — depth of i-th shark.
- w_i $(0 \le w_i \le w)$ — initial distance from i-th shark to the rope.

- f_i (f_i is 1 or -1) — initial direction of i-th shark's movement in relation to the rope (1 if it swims away from the rope, or -1 if it swims to the rope).

Initially the diver is more than r feet from any shark.

Output

Write to the output file `IMPOSSIBLE` if the diver cannot get to the surface or write the minimal time that it will take the diver to resurface with precision of at least 10^{-5}.

Example

diver.in	diver.out
10 1 2 1 10 1 6 4 -1 1 1 1	11.414213562373096

Problem 23. Лотерея с именами задач

Input file:	`lottery.in`
Output file:	`lottery.out`
Time limit:	2 секунды
Memory limit:	256 мегабайт

А упала, В пропала...

Из воспоминаний участника контеста

При распределении задач по порядку в жюри возникли жаркие споры. Каждый отстаивал свою точку зрения на порядок задач. Дело почти дошло до драки, когда Ааз предложил свой вариант решения проблемы.

— Давайте решим вопрос жеребьёвкой. Вот круглый стол, разделим его на N одинаковых секторов, по числу задач — как при игре в рулетку — и разложим по секторам условия задач. Затем запускается рулетка, шарик которой указывает на один из секторов. Если в секторе лежит условие задачи, то данная задача получает очередной номер. Если сектор пуст, выбирается первая встретившаяся при обходе по часовой стрелке, начиная с «указанного» шариком сектора, задача. Считается, что выпадение всех секторов равновероятно.

Предложение Ааза было принято, и началось распределение задач. В какой-то момент часть задач уже получила номера, и соответствующие секторы опустели. Автор одной очень сложной задачи не уверен в своём решении и хочет, чтобы его задача была последней — возможно, тогда многие участники «не дойдут» до этой задачи. Дождавшись момента, когда все члены жюри отвернутся, он хочет поменять условия своей задачи и ещё какой-то задачи, ещё оставшейся на столе, так, чтобы шанс на то, что его задача окажется последней, был максимален. Но для этого ему необходимо знать для каждого из оставшихся непустыми секторов вероятность того, что задача из этого сектора будет выбрана последней.

Напишите программу, вычисляющую эти вероятности.

Input

В первой строке задано число $1 \le N \le 200$. Во второй — N чисел, каждое из которых равно 0 или 1 — соответственно, осталась ли на данный момент задача в соответствующем секторе, или нет. Сектора даны по часовой стрелке.

Output

Выведите ровно N вещественных чисел не менее чем с тремя точными знаками после запятой — вероятности того, что последней окажется задача с соответствующего сектора (сектора выводятся в том же порядке, что и во входном файле). Вероятность, что последней окажется задача с уже выбранного сектора, равна нулю.

Examples

lottery.in	lottery.out
3 1 1 1	0.333 0.333 0.333
3 1 0 1	0.667 0 0.333
3 0 1 0	0 1 0

Problem 29. Отражение

Input file:	`mirror.in`
Output file:	`mirror.out`
Time limit:	2 секунды
Memory limit:	256 мегабайт

Здесь была моя ладья!

Садко после встречи с пиратами

Пока представители технической группы изучали жалобу Скива и разбирались с системой, Ааз зашёл в компьютерные классы. Участники развлекались как могли. Кто играл в крестики-нолики, кто пытался запустить свежеустановленный Turbo Delphi...

Двое участников склонились над шахматной доской, позиция на которой была крайне странной. Присмотревшись, Ааз понял, что в этой позиции необычного: на поле не было королей. Впрочем, и других фигур там тоже не было, кроме четырёх ладей — двух чёрных и двух белых.

— Странные правила, — подумал Ааз. — Если игра является нетривиальной, то имеет смысл её изучить и потом предложить в казино Базара-на-Деве. Всё-таки дополнительный источник авторитета (да и прибыли) корпорации «МИФ» не помешает.

Полученное Аазом описание игры — она называлась «Отражение» — было следующим:

> В игре участвуют два игрока. Изначально каждый игрок обладает двумя ладьями, которые, как и в шахматах, ходят на произвольное количество клеток по вертикали или горизонтали. Если ладья переместилась на край доски, то она может «отскочить» по диагонали, при этом если на пути «отскока» встретится хотя бы одна ладья противника, то та сбивается с доски.
>
> Запрещается отскакивать от той клетки, в которой ладья находится в данный момент, то есть, если ладья находится на некотором краю доски, то она не может отскакивать от этого края, а может отскакивать только от угловых клеток и от противоположного края. Если ладья находится в угловой клетке, то она может отскакивать только от двух угловых клеток.
>
> Например, ладья, находящаяся на поле d6, переместившись на поле h6, может сбить ладью противника на полях g5, f4, e3, d2, c1, g7, f8. Если на пути ладьи при отскоке встретилась ладья своего цвета, то через неё нельзя перепрыгивать, однако можно сбить ладью противника на любой из клеток, находящихся на пути отскока до ладьи своего цвета.
>
> Дополнительные ограничения таковы: в процессе обычного хода ладья не может сбить или даже перепрыгнуть через другую ладью, как своего, так и чужого цвета. Также запрещается ставить ладью на одну вертикаль или горизонталь с ладьёй противника, при этом отскок от клетки, находящейся на одной вертикали или горизонтали с ладьёй противника, разрешён.
>
> Кроме того, запрещается «отскакивать» без сбития ладьи противника.
>
> Цель игры — сбить ладью противника.

Игра показалась Аазу достаточно нетривиальной, а значит — достойной изучения. Но вот какое положение выигрышное, какое проигрышное, а какое — ничейное, он с ходу понять не смог. Возможно, сказывался сдвиг по времени, а возможно, что какое-то количество опасных строк всё же было прочитано прошлой ночью...

Напишите программу, помогающую анализировать позиции в игре «Отражение».

Input

В первой строке находится целое положительное число N — количество тестов ($N \leq 100\,000$). В следующих N строках даны начальные позиции. Начальная позиция задается позициями двух белых ладей и двух черных ладей. Позиции ладей отделены друг от друга одним пробелом. Ладьи противников не стоят на одной горизонтали или вертикали. Две ладьи не находятся на одной клетке. В начальной позиции ход белых.

Output

Для каждой позиции в отдельной строке выведете сколько ходов будет продолжаться партия, если противники играют наилучшим образом. Если начальная позиция ничейна, выведите 0.

Example

mirror.in	mirror.out
4 a2 b3 g8 h7 a1 b2 e7 e8 e7 e6 a1 b2 a2 b1 g8 h7	1 2 3 0