Problem A
Aaah!

Jon Marius shouted too much at the recent Justin Bieber concert, and now needs to go to the doctor because of his sore throat. The doctor's instructions are to say "aaah". Unfortunately, the doctors sometimes need Jon Marius to say "aaah" for a while, which Jon Marius has never been good at. Each doctor requires a certain *level* of "aah" – some require "aaaaaah", while others can actually diagnose his throat with just an "h". (They often diagnose wrongly, but that is beyond the scope of this problem.) Since Jon Marius does not want to go to a doctor and have his time wasted, he wants to compare how long he manages to hold the "aaah" with the doctor's requirements. (After all, who wants to be all like "aaah" when the doctor wants you to go "aaaaaah"?)

Photo by Unknown

Each day Jon Marius calls up a different doctor and asks them how long his "aaah" has to be. Find out if Jon Marius would waste his time going to the given doctor.

Input

The input consists of two lines. The first line is the "aaah" Jon Marius is able to say that day. The second line is the "aah" the doctor wants to hear. Only lowercase 'a' and 'h' will be used in the input, and each line will contain between 0 and 999 'a's, inclusive, followed by a single 'h'.

Output

Output "`go`" if Jon Marius can go to that doctor, and output "`no`" otherwise.

Sample Input 1	Sample Output 1
`aaah` `aaaaah`	`no`

Sample Input 2	Sample Output 2
`aaah` `ah`	`go`

This page is intentionally left blank.

Problem B
Birds on a Wire

There is a long electrical wire of length ℓ centimetres between two poles where birds like to sit. After a long day at work you like to watch the birds on the wire from your balcony. Some time ago you noticed that they don't like to sit closer than d centimetres from each other. In addition, they cannot sit closer than 6 centimetres to any of the poles, since there are spikes attached to the pole to keep it clean from faeces that would otherwise damage and weaken it. You start wondering how many more birds can possibly sit on the wire.

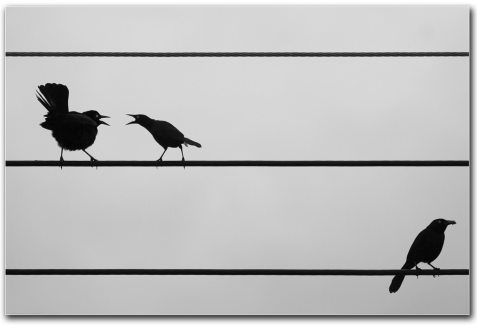


Photo by Tarik Browne

Task

Given numbers ℓ and d, how many additional birds can sit on the wire given the positions of the birds already on the wire? For the purposes of this problem we assume that the birds have zero width.

Input

The first line contains three space separated integers: the length of the wire ℓ, distance d and number of birds n already sitting on the wire. The next n lines contain the positions of the birds in any order. All number are integers, $1 \leq \ell, d \leq 1\,000\,000\,000$ and $0 \leq n \leq 20\,000$. (If you have objections to the physical plausibility of fitting that many birds on a line hanging between two poles, you may either imagine that the height of the line is 0 cm above ground level, or that the birds are ants instead.) You can assume that the birds already sitting on the wire are at least 6 cm from the poles and at least d centimetres apart from each other.

Output

Output one line with one integer – the maximal number of additional birds that can possibly sit on the wire.

Sample Input 1	Sample Output 1
22 2 2 11 9	3

Sample Input 2	Sample Output 2
47 5 0	8

This page is intentionally left blank.

Problem C
Black Friday


Black friday by Powhusku

Black Friday is the Friday following Thanksgiving Day in the United States (the fourth Thursday of November). Since the early 2000s, it has been regarded as the beginning of the Christmas shopping season in the US, and most major retailers open very early and offer promotional sales. (Source: Wikipedia)

You work in the IT support division of an electronics store. This year, in an attempt to prevent overcrowding, the management has decided to limit the number of people entering the store. They divide the people at the entrance into groups of size n and process them as follows: all n participants roll a die, and report the outcomes $a_1, a_2, \ldots, a_n$. To prevent cheating, instead of selecting the one with the highest outcome, the rule is to select the participant with the highest unique outcome. Everybody not selected has to move to the back of the queue. If there is no winner, the experiment is repeated.

For example, if there are three players and the outcomes are 6, 6 and 5, then the third player wins, because the first and second player lose even though their outcomes are higher, since they both have the same outcome. If instead the third player also rolls 6, then nobody wins.

They asked you to write a program for selecting the winner.

Input

The first line of the input contains one integer n, $1 \leq n \leq 100$, the group size. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 6$ for all $1 \leq i \leq n$): the outcome of each participant's die roll.

Output

Output the index of the participant that has the highest unique outcome, or "`none`" (without the quotes) if nobody has a unique outcome.

Sample Input 1	Sample Output 1
8 1 1 1 5 3 4 6 6	4

Sample Input 2	Sample Output 2
3 4 4 4	none

This page is intentionally left blank.

Problem D
Chess

In chess the bishop is the chessman, which can only move diagonal. It is well known that bishops can reach only fields of one color but all of them in some number of moves (assuming no other figures are on the field). You are given two coordinates on a chess-field and should determine, if a bishop can reach the one field from the other and how. Coordinates in chess are given by a letter ('A' to 'H') and a number (1 to 8). The letter specifies the column, the number the row on the chessboard.

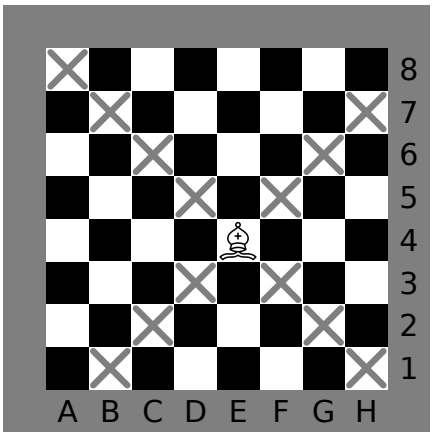


Figure D.1: Chessboard, bishop and fields the bishop can reach in one move

Input

The input starts with the number of test cases. Each test case consists of one line, containing the start position X and end position Y. Each position is given by two space separated characters. A letter for the column and a number for the row. There are no duplicate test cases in one input.

Output

Output one line for every test case. If it's not possible to move a bishop from X to Y in any number of moves output '`Impossible`'. Otherwise output one possible move sequence from X to Y. Output the number n of moves first (allowed to be 4 at most). Followed by $n + 1$ positions, which describe the path the bishop has to go. Every character is separated by one space. There are many possible solutions. Any with at most 4 moves will be accepted. Remember that in a chess move one chessman (the bishop in this case) has to change his position to be a valid move (i.e. two consecutive positions in the output must differ).

Sample Input 1	Sample Output 1
3 E 2 E 3 F 1 E 8 A 3 A 3	Impossible 2 F 1 B 5 E 8 0 A 3

This page is intentionally left blank.

Problem E
Extensive Or

Consider a very large number R in a compressed format. It is compressed as a binary string s, and an integer k. Start with the empty string, and append s to it k times to get the binary representation of R. The string s is guaranteed to have a leading 1. Now, with R, solve the following problem: How many sets of n distinct integers are there such that each integer is between 0 and $R-1$, inclusive, and the XOR of all those integers is equal to zero? Since this number can get very large, return it modulo $10^9 + 7$.

As a reminder, XOR is Exclusive Or. The XOR of two numbers is done bitwise. Using $\oplus$ for XOR:

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

XOR is associative, so $a \oplus (b \oplus c) = (a \oplus b) \oplus c$.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each input consists of exactly two lines. The first line has two space-separated integers n ($3 \leq n \leq 7$) and k ($1 \leq k \leq 100\,000$), where n is the number of distinct integers in the sets, and k is the number of times to repeat the string s in order to build R. The second line contains only the string s, which will consist of at least 1 and at most 50 characters, each of which is either `0` or `1`. s is guaranteed to begin with a `1`.

Output

Output a single line with a single integer, indicating the number of sets of n distinct integers that can be formed, where each integer is between 0 and $R - 1$ inclusive, and the XOR of the n integers in each set is 0. Output this number modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
3 1 100	1

Sample Input 2	Sample Output 2
4 3 10	1978

Sample Input 3	Sample Output 3
5 100 1	598192244

This page is intentionally left blank.

Problem F
What's on the Grille?

The *grille cipher* is a technique that dates back to 1550 when it was first described by Girolamo Cardano. The version we'll be dealing with comes from the late 1800's and works as follows. The message to be encoded is written on an $n \times n$ grid row-wise, top to bottom, and is overlaid with a card with a set of holes punched out of it (this is the grille).

The message is encrypted by writing down the letters that appear in the holes, row by row, then rotating the grille 90 degrees clockwise, writing the new letters that appear, and repeating this process two more times. Of course the holes in the grille must be chosen so that every letter in the message will eventually appear in a hole (this is actually not that hard to arrange).

An example is shown below, where the message "Send more monkeys" is encrypted as "noeesrksdmnyemoj", after adding a random letter to fill out the grid (this example corresponds to the first sample input.)

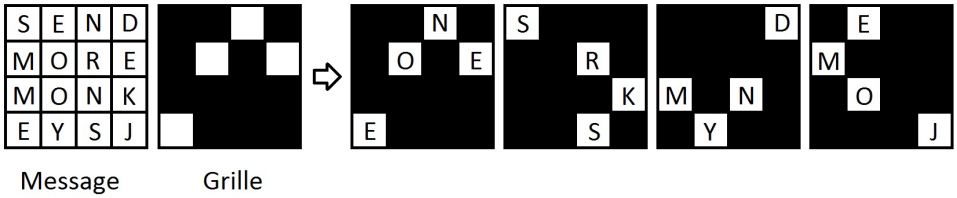


Figure I.1

If the message is larger than the $n \times n$ grid, then the first n^2 letters are written in the grid and encrypted, then the next n^2 are encrypted, and so on, always filling the last grid with random letters if needed. Here, we will only be dealing with messages of length n^2.

Your job, should you choose to accept it, is to take an encrypted message and the corresponding grille and decrypt it. And we'll add one additional twist: the grille given might be invalid, i.e., the holes used do not allow every location in the grid to be used during the encryption process. If this is the case, then you must indicate that you can't decrypt the message.

Input

The input starts with a line containing a positive integer $n \leq 10$ indicating the size of the grid and grille. The next n lines will specify the grille, using '.' for a hole and 'X' for a non-hole. Following this will be a line containing the encrypted message, consisting solely of lowercase alphabetic characters. The number of characters in this line will always be n^2.

Output

Output the decrypted text as a single string with no spaces, or the phrase "`invalid grille`" if the grille is invalid.

Sample Input 1	Sample Output 1
4 XX.X X.X. XXXX .XXX noeesrksdmnyemoj	sendmoremonkeysj

Sample Input 2	Sample Output 2
4 .XX. XXXX XXXX .XX. abcdefghijklmnop	invalid grille

Sample Input 3	Sample Output 3
2 X. XX aybb	baby

Problem G
Restaurant Orders

A friend of yours who is working as a waiter has a problem. A group of xkcd-fans have started to come to the restaurant and order food as in the comic strip below. Each order takes him a lot of time to figure out, but maybe you can help him.

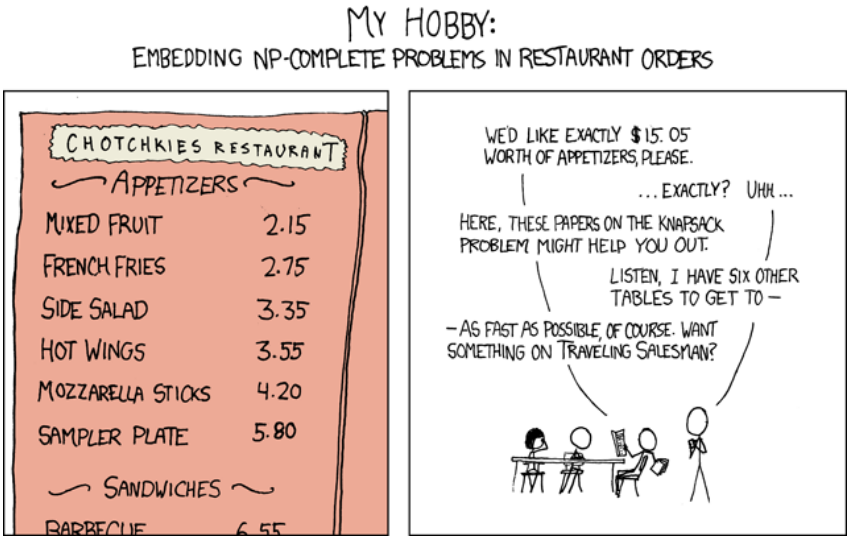


Figure G.1: Comic strip xkcd.com/287.

Task

You are to write a program that finds out what was ordered given the total cost of the order and the cost of each item on the menu.

Input

The input starts with a line containing one integer n ($1 \leq n \leq 100$), the number of items on the menu. The next line contains n space-separated positive integers $c_1, c_2, \ldots, c_n$, denoting the cost of each item on the menu in Swedish kronor. No item costs more than 1 000 SEK.

This is followed by a line containing m ($1 \leq m \leq 1\,000$), the number of orders placed, and a line with m orders. Each order is given as an integer s ($1 \leq s \leq 30\,000$), the total cost of all ordered items in SEK.

Output

For each order in the input output one line as follows. If there is one unique order giving the specified total cost, output a space-separated list of the numbers of the items on that order in ascending order. If the order contains more than one of the same item, print the corresponding number the appropriate number of times. The first item on the menu has number 1, the second 2, and so on.

If there doesn't exist an order that gives the specified sum, output `Impossible`. If there are more than one order that gives the specified sum, output `Ambiguous`.

Sample Input 1	Sample Output 1
3 4 5 8 3 11 13 14	Impossible Ambiguous 1 2 2

Sample Input 2	Sample Output 2
6 215 275 335 355 420 580 1 1505	Ambiguous

Problem H
Room Assignments

Once there was an inventor congress, where inventors from all over the world met in one place. The organizer of the congress reserved exactly one hotel room for each inventor. Each inventor, however, had its own preference regarding which room he would like to stay in. Being a clever inventor himself, the organizer soon found an objective way of doing the room assignments in a fair manner: each inventor wrote two different room numbers on a fair coin, one room number on each side. Then, each inventor threw his coin and was assigned the room number which was shown on the upper side of his coin. If some room had been assigned to more than one inventor, all inventors had to throw their coins again.

As you can imagine, this assignment process could take a long time or even not terminate at all. It has the advantage, however, that among all possible room assignments, one assignment is chosen randomly according to a uniform distribution. In order to apply this method in modern days, you should write a program which helps the organizer.

The organizer himself needs a hotel room too. As the organizer, he wants to have some advantage: he should be able to rate each of the rooms (the higher the rating, the better), and the program should tell him which two room numbers he should write on his coin in order to maximize the expected rating of the room he will be assigned to. The program also has access to the choices of the other inventors before making the proposal. It should never propose two rooms for the organizer such that it is not possible to assign all inventors to the rooms, if a valid assignment is possible at all.

Input

The input starts with a single number c ($1 \leq c \leq 200$) on one line, the number of test cases. Each test case starts with one line containing a number n ($2 \leq n \leq 50\,000$), the number of inventors and rooms. The following $n-1$ lines contain the choices of the $n-1$ guests (excluding the organizer). For each inventor, there is a line containing two numbers a and b ($1 \leq a < b \leq n$), the two room numbers which are selected by the inventor. The last line of each test case consists of n integers $v_1, \ldots, v_n$ ($1 \leq v_i \leq 1\,000\,000$), where v_i is the organizer's rating for room i.

Output

For each test case, print a single line containing the two different room numbers a and b which should be selected by the organizer in order to maximize the expected rating of the room he will be assigned to. If there is more than one optimal selection, break ties by choosing the smallest a and, for equal a, the smallest b. If there is no way for the organizer to select two rooms such that an assignment of inventors to rooms is possible, print "`impossible`" instead.

Sample Input 1	Sample Output 1
3 4 1 2 2 3 1 3 2 3 4 1 3 1 2 2 3 100 40 70 5 1 2 1 2 1 2 3 4 1 1 1 1 1	1 4 1 3 impossible

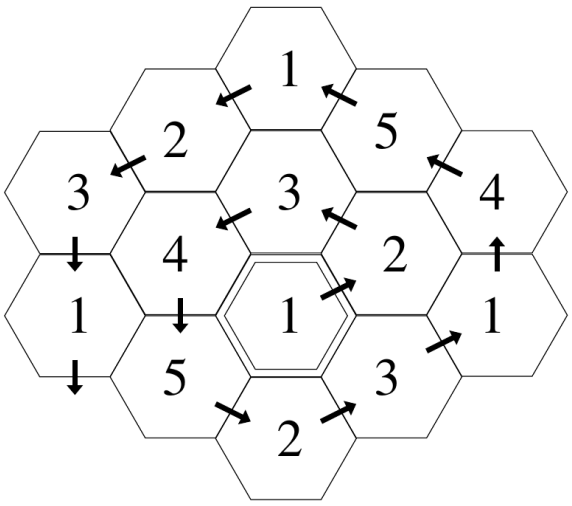
Problem I
Settlers of Catan

The popular board game "Settlers of Catan" starts by creating a random board. This board consists of hexagonal resource tiles containing five different resources: clay, lumber, wool, grain, and ore. For simplicity, we will denote these by the numbers 1 to 5.

Random boards, however, often have multiple equal resource tiles next to each other. This annoys some players. Therefore, we have invented a new way of creating the playing board. Starting in the middle and spiraling outwards, each time we add a new tile to the board we choose the resource of the tile according to the following rules:

- the new tile must be different from its neighboring tiles on the board so far;
- in case multiple tiles are possible, we choose a resource that occurs the least number of times on the board so far;
- in case multiple tiles are still possible, the new resource must have the lowest number possible.

The figure underneath shows how to spiral outwards and which resource tiles are chosen first. We are curious what the number of the resource is on the nth tile that is added to the board (starting with $n = 1$).



Input

On the first line of the input there is one integer c ($1 \leq c \leq 200$), the number of test cases. Each following test case consists of a single line with one integer n ($1 \leq n \leq 10\,000$), the number of the tile we are curious about.

Output

For each test case, print a single line with one integer, specifying the resource of the nth tile.

Sample Input 1	Sample Output 1
4 1 4 10 100	1 4 5 5

Problem J
Being Solarly Systematic

Professor Braino Mars is one of the top researchers in the field of solar system creation. He runs various simulations to test out his theories on planet formation, but he's old school and all of these simulations are done by hand. It's time for Braino to enter the 21^{st} century, and he's asked you to help automate his simulations.

One of Prof. Mars' simulations models how small planetoids collide over time to form larger planets. To model this process he divides the space which the planetoids inhabit into an $n_x \times n_y \times n_z$ grid of cubes, where each cube can hold at most one planetoid. Each planetoid has an initial mass m, an initial location (x, y, z) in the grid and a velocity (v_x, v_y, v_z) indicating the number of cubes per second the planetoid travels through in each dimension. For example, if a planetoid is initially in location $(1, 3, 2)$ and has velocity $(3, -1, 2)$, then after 1 second it will be in location $(4, 2, 4)$, after 2 seconds it will be in location $(7, 1, 6)$, and so on. The planetoid paths wrap around in all dimensions, so if, for example, the planetoid described above resides in an $8 \times 8 \times 8$ space, its next two locations will be $(2, 0, 0)$ and $(5, 7, 2)$ (note that all cube indices start at 0). When two or more planetoids collide, they form one larger planetoid which has a mass equal to the sum of the colliding planetoids' masses and a velocity equal to the average of the colliding velocities, truncating to the nearest integer. So if a planetoid of mass 12 with velocity $(5, 3, -2)$ collides with another planetoid of mass 10 and velocity $(8, -6, 1)$ the resulting planetoid has mass 22 and velocity $(6, -1, 0)$ (these values correspond to the first sample input.) For simplicity, Prof. Mars only considers collisions that happen at integer time steps, and when no more collisions are possible, the planetoids are then considered full-fledged planets.

Given an initial set of planetoids, Prof. Mars is interested in determining how many planets will form and what their orbits are. Armed with your implementation of his model, he should now be able to answer these questions much more easily.

Input

The input will start with a line containing four positive integers $n\ n_x\ n_y\ n_z$, where $n \leq 100$ is the number of planetoids, and n_x, n_y and n_z are the dimensions of the space the planetoids reside in, where $n_x, n_y, n_z \leq 1000$.

After this are n lines of the form $m\ x\ y\ z\ v_x\ v_y\ v_z$, specifying the mass, initial location and initial velocity of each planetoid at time $t = 0$, where $1 \leq m \leq 100$, $0 \leq x < n_x, 0 \leq y < n_y, 0 \leq z < n_z$, and $-1000 \leq v_x, v_y, v_z \leq 1000$. No two planetoids will start in the same initial location.

Output

Output an integer p indicating the number of planets in the system after no more collisions can occur. After this output p lines, one per planet, listing a planet identifier Pi, $(0 \leq i < p)$, the mass, location and velocity of each planet. Use the location of the planets at the time that the last collision occurred.

If no collisions occur, then use their location at time $t = 0$.

The planets should be ordered from largest mass to smallest; break ties by using the lexicographic ordering of the x, y, z location of the planet, starting with the smallest x value.

Sample Input 1	**Sample Output 1**

`2 8 8 8`
`12 4 1 4 5 3 -2`
`10 1 2 1 8 -6 1`
	`1`
	`P0: 22 1 4 2 6 -1 0`

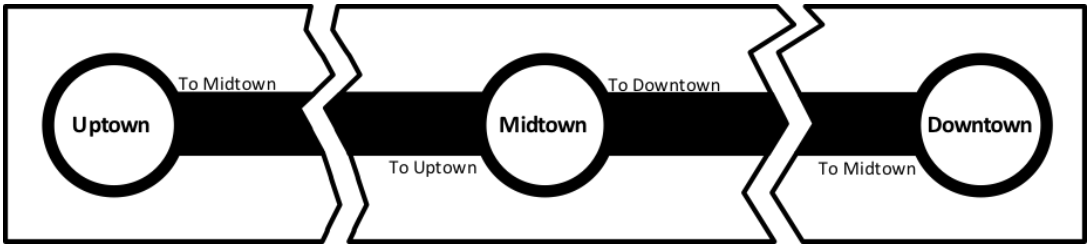
Sample Input 2	**Sample Output 2**

`2 10 20 30`
`10 1 0 0 2 0 0`
`15 2 0 0 4 0 0`
	`2`
	`P0: 15 2 0 0 4 0 0`
	`P1: 10 1 0 0 2 0 0`

Problem K
Torn To Pieces

You have arrived in The Big City but your journey is not yet complete. You must still navigate the subway and get to your final destination. The information booth in the subway station is unattended and fresh out of maps of the subway system. On the floor you notice fragments of a map. Can you piece together enough of the map to figure out how to get to your final destination?

Each fragment of the map happens to perfectly contain a single subway station while also identifying all of the other stations that it connects to. Each connection between stations is bi-directional such that it can be travelled going either direction. Using all of the available fragments, your task is to determine the sequence of stations you must pass through in order to reach your final destination or state that there is no route if you don't have enough information to complete your journey.



Input

The first line of input has an integer, $2 \leq N \leq 32$, that identifies the number of pieces of the map that were found.

The following N lines each describe a station depicted on one of those pieces. Each of these lines starts with the name of the station they describe and is followed by a space-separated list of all of the station names that are directly connected to that station (there may be as many as $N - 1$).

The final line identifies a starting station and a destination station. The destination station is guaranteed to be different than the starting station.

Each station name is a string of up to 20 characters using only letters a–z and A–Z. It is guaranteed that there is at most one simple route (without revisiting stations) from the starting station to the destination station.

Output

Give the sequence of stations that leads from the starting station to the destination station. Separate station names with spaces. If there are not enough pieces of the map to find a route from the starting station to the destination station then output "`no route found`".

Sample Input 1	Sample Output 1
3 Uptown Midtown Midtown Uptown Downtown Downtown Midtown Uptown Downtown	Uptown Midtown Downtown

Sample Input 2

```
6
A B
B A D
C D
E D F G
F E
G E
F A
```

Sample Output 2

```
F E D B A
```

Sample Input 3

```
4
FirstStop SecondStop
SecondStop FirstStop ThirdStop
FifthStop FourthStop SixthStop
SixthStop FifthStop
FirstStop FifthStop
```

Sample Output 3

```
no route found
```

Problem L
Triangles

You got a very strange gift for your birthday: two triangles in the three-dimensional space. Each triangle consists of three infinitely thin segments, and each segment stays straight no matter how hard you press it. Now, you actually wanted to get just one triangle, so you try to move the triangles far apart from each other, possibly after rotating one or both of them, so that you can throw away one of them. Is it possible? Or are they tangled?

Input

The input consists of several lines. The first line contains $1 \leq T \leq 1000$, the number of test cases. Each test case consists of two lines. The first line contains 9 integers $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3 \in [-1000, 1000]$ denoting the vertices of the first triangle. The second line contains another 9 integers $x'_1, y'_1, z'_1, x'_2, y'_2, z'_2, x'_3, y'_3, z'_3 \in [-1000, 1000]$ denoting the vertices of the second triangle. Both triangles will be non-degenerate, which means that the corresponding triples of points will not be colinear. Moreover, it is guaranteed that no pair of segments from two different triangles intersects, and there is no common plane containing both triangles at once.

Output

For each test case, output one line containing YES if the triangles are tangled, and NO if it is possible to move them very far apart from each other.

Sample Input 1	Sample Output 1
2 0 0 0 10 0 0 0 10 0 1 1 10 1 1 -10 10 10 0 0 0 0 10 0 0 0 10 0 11 0 0 0 11 0 11 11 1	YES NO

This page is intentionally left blank.