Problem A
A New Alphabet

A New Alphabet has been developed for Internet communications. While the glyphs of the new alphabet don't necessarily improve communications in any meaningful way, they certainly make us *feel cooler*.

Photo by r. nial bradshaw

You are tasked with creating a translation program to speed up the switch to our more *elite* New Alphabet by automatically translating ASCII plaintext symbols to our new symbol set.

The new alphabet is a one-to-many translation (one character of the English alphabet translates to anywhere between 1 and 6 other characters), with each character translation as follows:

Original	New	English Description	Original	New	English Description
`a`	`@`	at symbol	`n`	`[]\[]`	brackets, backslash, brackets
`b`	`8`	digit eight	`o`	`0`	digit zero
`c`	`(`	open parenthesis	`p`	`|D`	bar, capital D
`d`	`|)`	bar, close parenthesis	`q`	`(,)`	parenthesis, comma, parenthesis
`e`	`3`	digit three	`r`	`|Z`	bar, capital Z
`f`	`#`	number sign (hash)	`s`	`$`	dollar sign
`g`	`6`	digit six	`t`	`'][' `	quote, brackets, quote
`h`	`[-]`	bracket, hyphen, bracket	`u`	`|_|`	bar, underscore, bar
`i`	`|`	bar	`v`	`\/`	backslash, forward slash
`j`	`_|`	underscore, bar	`w`	`\/\/`	four slashes
`k`	`|<`	bar, less than	`x`	`}{`	curly braces
`l`	`1`	digit one	`y`	`` `/ ``	backtick, forward slash
`m`	`[]\/[]`	brackets, slashes, brackets	`z`	`2`	digit two

For instance, translating the string "Hello World!" would result in:

```
[-]3110 \/\/0|Z1|)!
```

Note that uppercase and lowercase letters are both converted, and any other characters remain the same (the exclamation point and space in this example).

Input

Input contains one line of text, terminated by a newline. The text may contain any characters in the ASCII range 32–126 (space through tilde), as well as 9 (tab). Only characters listed in the above table (A–Z, a–z) should be translated; any non-alphabet characters should be printed (and not modified). Input has at most 10 000 characters.

Output

Output the input text with each letter (lowercase and uppercase) translated into its New Alphabet counterpart.

Sample Input 1

```
All your base are belong to us.
```

Sample Output 1

```
@11 ‘/0|_||Z 8@$3 @|Z3 8310[]\[]6 ’][’0 |_|$.
```

Sample Input 2

```
What’s the Frequency, Kenneth?
```

Sample Output 2

```
\/\/[-]@’][’’$ ’][’[-]3 #|Z3(,)|_|3[]\[](‘/, |<3[]\[][]\[]3’][’[-]?
```

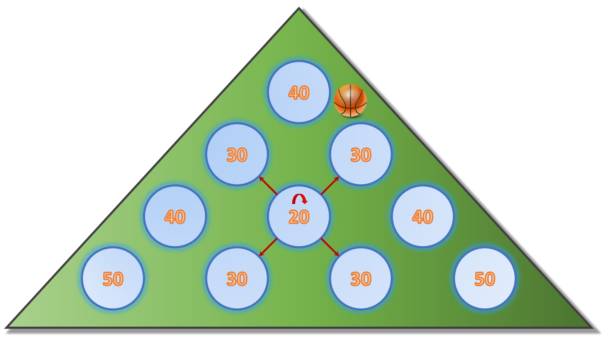
Sample Input 3

```
A new alphabet!
```

Sample Output 3

```
@ []\[]3\/\/ @1|D[-]@83’][’!
```

Problem B
Arcade!



Have you recently visited an arcade? Arcade games seem to have become more boring over the years, requiring less and less skill. In fact, most arcade games these days seem to depend entirely on luck. Consider the arcade game shown in the picture, which consists of different holes arranged in a triangular shape. A ball is dropped near the hole at the top. The ball either falls into the hole, in which case the game ends, or it bounces to one of its (up to) 4 neighbors, denoted by the red arrows. Different holes have different payouts — some may even be negative! If the ball reaches another hole, the process repeats: the ball either falls into the hole, ending the game — or it bounces to one of its neighbors, possibly ad infinitum!

Write a program that computes the expected payout when dropping a ball into the machine!

Input

The input consists of a single test case. The first line contains an integer N ($1 \leq N \leq 32$) describing the number of rows of the arcade machine. The second line contains $H = N(N+1)/2$ integers v_i ($-100 \leq v_i \leq 100$) describing the payout (positive or negative) if the ball drops into hole i. Holes are numbered such that hole 1 is in the first row, holes 2 and 3 are in the second row, etc. The k^{th} row starts with hole number $k(k-1)/2+1$ and contains exactly k holes.

These two lines are followed by H lines, each of which contains 5 real numbers p_0 p_1 p_2 p_3 p_4, denoting the probability that the ball bounces to its top-left (p_0), top-right (p_1), bottom-left (p_2), or bottom-right (p_3) neighbors or that the ball enters the hole (p_4). Each probability is given with at most 3 decimal digits after the period. It is guaranteed that $0.0 \leq p_i \leq 1.0$ and $\sum p_i = 1.0$. If a hole does not have certain neighbors because it is located near the boundary of the arcade machine, the probability of bouncing to these non-existent neighbors is always zero. For instance, for hole number 1, the probabilities to jump to the top-left and top-right neighbors are both given as 0.0.

You can assume that after the ball has bounced b times, the probability that it has not fallen into a hole is at most $(1-10^{-3})^{\lfloor b/H \rfloor}$.

Output

Output a single number, the expected value from playing one game. Your answer is considered correct if its absolute or relative error is less than 10^{-4}.

Hint: Using Monte Carlo-style simulation (throwing many balls in the machine and simulating which hole they fall into using randomly generated choices) does not yield the required accuracy!

Sample Input 1

```
4
40 30 30 40 20 40 50 30 30 50
0.0 0.0 0.45 0.45 0.1
0.0 0.3 0.3 0.3 0.1
0.3 0.0 0.3 0.3 0.1
0.0 0.3 0.3 0.3 0.1
0.2 0.2 0.2 0.2 0.2
0.3 0.0 0.3 0.3 0.1
0.0 0.8 0.0 0.0 0.2
0.4 0.4 0.0 0.0 0.2
0.4 0.4 0.0 0.0 0.2
0.8 0.0 0.0 0.0 0.2
```

Sample Output 1

```
32.6405451448
```

Sample Input 2

```
2
100 50 50
0.0 0.0 0.45 0.45 0.1
0.0 0.90 0.0 0.0 0.10
0.90 0.0 0.0 0.0 0.10
```

Sample Output 2

```
76.31578947368
```

Problem C
Big Truck

Photo by Phil Whitehouse

Your boss has hired you to drive a big truck, transporting items between two locations in a city. You're given a description of the city, with locations of interest and the lengths of roads between them. Your boss requires that you take a shortest path between the starting and ending location, and she'll check your odometer when you're done to make sure you didn't take any unnecessary side trips. However, your friends know you have plenty of unused space in the truck, and they have asked you to stop by several locations in town, to pick up items for them. You're happy to do this for them. You may not be able to visit every location to pick up everything your friends want, but you'd like to pick up as many items as possible on your trip, as long as it doesn't make the path any longer than necessary.

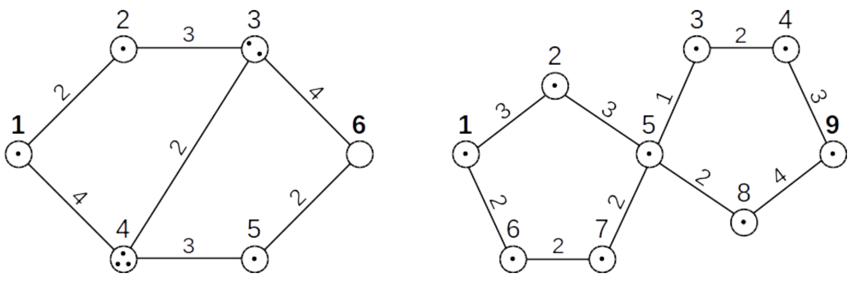


Figure C.1: Illustrations of the first two sample inputs

The two graphs above show examples of what the city may look like, with nodes representing locations, edges representing roads and dots inside the nodes representing items your friends have asked you to pick up. Driving through a location allows you to pick up all the items there; it's a big truck, with no limit on the items it can carry. In the graph on the left, for example, you have to drive the big truck from location 1 to location 6. If you follow the path $1 \to 2 \to 3 \to 6$, the length is 9, and you'll get to pick up 4 items. Of course, it would be better to drive $1 \to 4 \to 5 \to 6$; that's still a length of 9, but going this way instead lets you pick up an additional item. Driving $1 \to 4 \to 3 \to 6$ would let you pick up even more items, but it would make your trip longer, so you can't go this way.

Input

The first line of input contains an integer, n ($2 \le n \le 100$), giving the number of locations in the city. Locations are numbered from 1 to n, with location 1 being the starting location and n being the destination. The next input line gives a sequence of n integers, $t_1 \dots t_n$, where each t_i indicates the number of items your friends have asked you to pick up from location i. All the t_i values are between 0 and 100, inclusive. The next input line contains a non-negative integer, m, giving the number of roads in the city. Each of the following m lines is a description of a road, given as three integers, a b d. This indicates that there is a road of length d between location a and location b. The values of a and b are in the range $1 \dots n$, and the value of d is between 1 and 100, inclusive. All roads can be traversed in either direction, there is at most one road between any two locations, and no road starts and ends at the same location.

Output

If it's not possible to travel from location 1 to location n, just output out the word "impossible". Otherwise, output the length of a shortest path from location 1 to location n, followed by the maximum number of items you can pick up along the way.

Sample Input 1

```
6
1 1 2 3 1 0
7
1 2 2
2 3 3
3 6 4
1 4 4
4 3 2
4 5 3
5 6 2
```

Sample Output 1

```
9 5
```

Sample Input 2

```
9
1 1 1 1 1 1 1 1 1
10
1 2 3
2 5 3
1 6 2
6 7 2
7 5 2
5 3 1
3 4 2
4 9 3
5 8 2
8 9 4
```

Sample Output 2

```
12 7
```

Sample Input 3

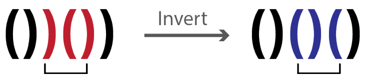
```
2
5 5
0
```

Sample Output 3

```
impossible
```

Problem D
Brackets

A bracket sequence consisting of '(' and ')' is defined to be *valid* as follows:



1. An empty sequence is valid.
2. If X is a valid bracket sequence, then (X) is a valid bracket sequence.
3. If X and Y are valid bracket sequences, then the concatenation of X and Y, $Z = XY$, is a valid bracket sequence.

For example, "(())", "()()", and "(()())()" are all valid bracket sequences, while "(" and "())" are invalid bracket sequences.

You get a bracket sequence from the professor of length n. However, it might not be valid at the moment. The professor asks you to check if it is possible to make the sequence valid by performing *at most one* segment inversion operation. That is, you may choose two 1-based indices l and r ($1 \le l \le r \le n$) and invert each bracket with an index in the closed interval $[l, r]$. After the inversion, a left bracket '(' becomes a right bracket ')', and a right bracket ')' becomes a left bracket '('.

You can make "())(" valid by inverting the segment $[3, 4]$. You can make "()))" valid by inverting the segment $[3, 3]$, or alternatively by inverting the segment $[2, 2]$. However, there does not exist a segment you can invert to make ")))(" valid.

Input

The input consists of one line containing between 1 and $5\,000$ brackets.

Output

Output "possible" if you can make the bracket sequence valid by performing at most one segment inversion, or "impossible" otherwise.

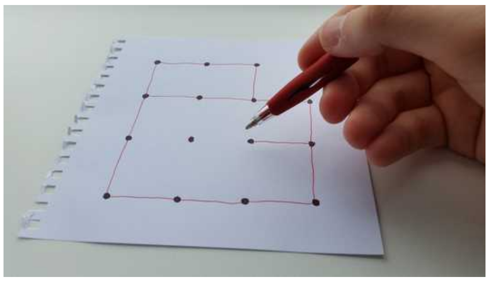
Sample Input 1	Sample Output 1
`()))`	`possible`

Sample Input 2	Sample Output 2
`)))(`	`impossible`

Sample Input 3	Sample Output 3
`()`	`possible`

This page is intentionally left blank.

Problem E
Dots and Boxes


A game of Dots and Boxes.

Alice and Bob are playing Dots and Boxes. The game is played on an $N \times N$ square lattice of dots, and they alternate drawing a line segment between horizontally or vertically adjacent dots that haven't been connected before. Every time a unit square is formed by four line segments, the player who put down the last segment scores one point for that square. The game ends when the square lattice has been completely filled with line segments, and whoever scored the most points wins.

Alice and Bob aren't really in a competitive mood today, so they've just been playing for fun. Hence they aren't following any specific game strategy, and, in particular, even if it's possible to make a move that scores a point or is clearly superior in some way, they won't necessarily make that move. But now they've been playing for a while and neither of them has scored a single point. If neither of them scores a point pretty soon, they may get bored. Given the current state of the game, how many moves could be made, in the worst case, before either Alice or Bob is guaranteed to have scored a point?

Input

Input starts with a line containing an integer N ($2 \leq N \leq 80$), the size of the square lattice. Then follows an ASCII representation of the current state of the game, $2N-1$ rows high and $2N-1$ columns wide, listed in row-major order. There are cells of four types ($1 \leq i, j \leq N$):

- Cell $(2i-1, 2j-1)$ is '$\star$', representing dot (i, j).
- Cell $(2i, 2j)$ is '`.`', representing empty space.
- Cell $(2i, 2j-1)$ is '`|`' if dots (i, j) and $(i+1, j)$ have been connected by a line segment, and '`.`' otherwise.
- Cell $(2i-1, 2j)$ is '`-`' if dots (i, j) and $(i, j+1)$ have been connected by a line segment, and '`.`' otherwise.

It is guaranteed that no player has scored a point, meaning that no unit squares have been formed.

Output

Output the number of moves that can be made, in the worst case, before either Alice or Bob is guaranteed to have scored a point.

Sample Input 1

```
3
*-*.*
|.|.|
*.*-*
|...|
*.*.*
```

Sample Output 1

```
3
```

Sample Input 2

```
2
*.*
...
*.*
```

Sample Output 2

```
4
```

Sample Input 3

```
4
*-*-*.*
|...|..
*-*-*-*
|.....|
*.*.*-*
|.....|
*-*-*-*
```

Sample Output 3

```
5
```

Problem F
Free Desserts

Photo by Lotus Head

Quido has lunch in Hugo's restaurant every day. He likes the restaurant because all of its prices are expressed as integers, and for each possible price (i.e. \$1, \$2, \$3, etc.) there is at least one beverage and at least one main dish on the menu. Every day there are three entries printed on Quido's lunch bill: the beverage price, the main dish price, and the total price. Hugo knows of Quido's interest in computational problems and he offered Quido a free dessert each time his lunch bill meets the following three constraints:

- the bill is not identical to any of Quido's previous bills,
- the price of the beverage is less than the price of the main dish, and
- the prices listed on the bill cannot mutually use the same digit. In essence, any digit which occurs in any of the entries (beverage, main dish, total) must be different from any of the digits of the other two entries.

Quido is on a budget and he pays the same price for his lunch every day. How many times can he have a free dessert?

Input

The input consists of a single line with one integer representing the price P which Quido pays for each lunch. The value of P is positive and less than 10^{18}.

Output

Output the maximum number of times Quido can have a free dessert at Hugo's restaurant, provided that the price of his lunch is always P. Next, the possible bills which result in a free dessert are listed in ascending order with respect to the beverage price. Each bill consists of the price of the beverage followed by the price of the main dish. For simplicity, the value P, which is always the same, is not included in the bill.

If there are more than 5 000 possible bills, then output only the first 5 000 bills (but still report the total number of possible bills before the list of bills).

Sample Input 1

```
37
```

Sample Output 1

```
4
8 29
9 28
11 26
15 22
```

Sample Input 2

```
30014
```

Sample Output 2

```
7
85 29929
88 29926
785 29229
788 29226
7785 22229
7788 22226
7789 22225
```

Sample Input 3

```
202020202020202058
```

Sample Output 3

```
3
7676767676767667 194343434343434391
37373737373737397 164646464646464661
67676767676767667 134343434343434391
```

Problem G
Inverse Factorial

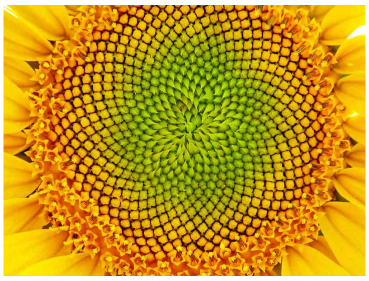


Photo by Ginette

A factorial $n!$ of a positive integer n is defined as the product of all positive integers smaller than or equal to n. For example,

$$21! = 1 \times 2 \times 3 \times \cdots \times 21 = 51\,090\,942\,171\,709\,440\,000.$$

It is straightforward to calculate the factorial of a small integer, and you have probably done it many times before. In this problem, however, your task is reversed. You are given the value of $n!$ and you have to find the value of n.

Input

The input contains the factorial $n!$ of a positive integer n. The number of digits of $n!$ is at most 10^6.

Output

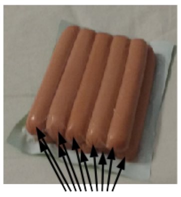
Output the value of n.

Sample Input 1	Sample Output 1
`120`	`5`

Sample Input 2	Sample Output 2
`51090942171709440000`	`21`

Sample Input 3	Sample Output 3
`10888869450418352160768000000`	`27`

Problem H
Nine Packs


10 hotdogs

8 buns

"It's like how hot dogs come in packs of ten, and buns come in packs of eight or twelve — you have to buy nine packs to make it come out even."

This is a quote from the 1986 movie, "True Stories", and it's true; well, almost true. You could buy four packs of 10 hotdogs and five packs of 8 buns. That would give you exactly 40 of each. However, you can make things even with fewer packs if you buy two packs of 10 hotdogs, along with a pack of 8 buns and another pack of 12 buns. That would give you 20 of each, using only 4 total packs.

For this problem, you'll determine the fewest packs you need to buy to make hotdogs and buns come out even, given a selection of different bun and hotdog packs available for purchase.

Input

The first input line starts with an integer, H, the number of hotdog packs available. This is followed by H integers, $h_1 \ldots h_H$, the number of hotdogs in each pack. The second input line starts with an integer, B, giving the number of bun packs available. This is followed by B integers, $b_1 \ldots b_B$, indicating the number of buns in each pack. The values H and B are between 0 and 100, inclusive, and the sizes of the packs are between 1 and $1\,000$, inclusive. Every available pack is listed individually. For example, if there were five eight-bun packs available for purchase, the list of bun packs would contain five copies of the number eight.

Output

If it's not possible to purchase an equal number of one or more hotdogs and buns, just output "impossible". Otherwise, output the smallest number of total packs you can buy (counting both hotdog and bun packs) to get exactly the same number of hotdogs and buns.

Sample Input 1

```
4 10 10 10 10
10 8 8 8 12 12 12 8 8 12 12
```

Sample Output 1

```
4
```

Sample Input 2

```
4 7 7 14 7
3 11 22 11
```

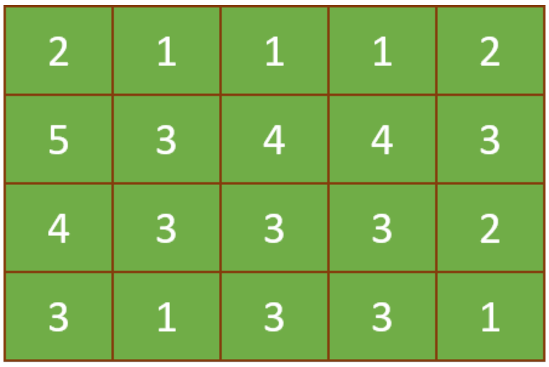
Sample Output 2

```
impossible
```

Problem I
Primonimo

Primonimo is a game played on an $n \times m$ board filled with numbers taken from the range $1 \ldots p$ for some prime number p. At each move, a player selects a square and adds 1 to the numbers in all squares in the same row and column as the selected square. If a square already shows the number p, it wraps around to 1.

The game is won if all squares show p. Given an initial board, find a sequence of moves that wins the game!



2	1	1	1	2
5	3	4	4	3
4	3	3	3	2
3	1	3	3	1

Primonimo board (the web version shows an animated version of the game).

Input

The input consists of a single test case. The first line contains three numbers n m p denoting the number of rows n ($1 \leq n \leq 20$), the number of columns m ($1 \leq m \leq 20$), and a prime number p ($2 \leq p \leq 97$). Each of the next n lines consists of m numbers in the range $1 \ldots p$.

Output

If a winning sequence of at most $p \cdot m \cdot n$ moves exists, output an integer $k \leq p \cdot m \cdot n$ denoting the number of moves in the sequence. Then output k moves as a sequence of integers that numbers the board in row-major order, starting with 1. If there are multiple such sequences, you may output any one of them. If no winning sequence exists, output `-1`.

Sample Input 1

```
4 5 5
2 1 1 1 2
5 3 4 4 3
4 3 3 3 2
3 1 3 3 1
```

Sample Output 1

```
6
19 12 2 18 5 5
```

Sample Input 2

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

Sample Output 2

```
13
4 2 6 1 9 7 5 5 7 1 2 3 3
```

Sample Input 3

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

Sample Output 3

```
-1
```

Sample Input 4

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

Sample Output 4

```
1
6
```

Problem J
Quick Estimates

Photo by Simon A. Eugster

Let's face it... you are not that handy. When you need to make a major home repair, you often need to hire someone to help. When they come for the first visit, they make an estimate of the cost. Here they must be careful: if they overestimate the cost, it might scare you off, but if they underestimate, the work might not be worth their time.

Because the worker is so careful, it can take a long time for them to produce the estimate. But that's frustrating — when you ask for an estimate, you really are asking for the magnitude of the cost. Will this be \$10 or \$100 or \$1 000? That's all you really want to know on a first visit.

Please help the worker make the type of estimate you desire. Write a program that, given the worker's estimate, reports just the magnitude of the cost — the number of digits needed to represent the estimate.

Input

Input begins with a line containing an integer N ($1 \le N \le 100$). The next N lines each contain one estimated cost, which is an integer between 0 and 10^{100}. (Some of the workmen overcharge quite a bit.)

Output

For each estimated cost, output the number of digits required to represent it.

Sample Input 1	Sample Output 1
5 314 1 5926 5 35897	3 1 4 1 5

Sample Input 2	Sample Output 2
3 0 10 100	1 2 3

This page is intentionally left blank.

Problem K
Robotopia

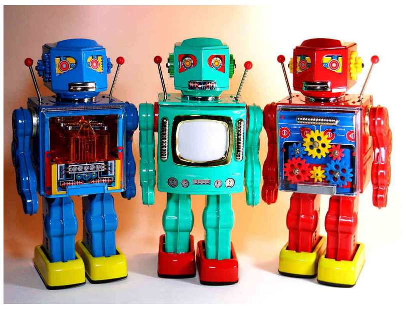


Photo by DJ Shin

In the wonderful land of Robotopia, life is better than ever. This is thanks to all the robots, who work hard together. Different types of robots come in different forms, that is, different numbers of arms and different numbers of legs. By day, when they are working, robots are grouped so they can be assigned to different tasks. But by night, it is your job to count how many robots there are of each type within each group. Robots are naughty and energetic; they don't like to stand still and it is difficult to count them. After many exhausting attempts, you are finally able to count how many arms and how many legs there are in total.

We know each group consists of exactly two different types of robots, with at least one of each type in each group. We also know the number of arms and legs that each type of robot has. Can you find out how many robots of each type there are within each group?

Input

The first line contains an integer n $(1 \leq n \leq 100)$. The following n lines each contain one test case. Each has six integers: l_1 a_1 l_2 a_2 l_t a_t, where l_1 and a_1 are the number of legs and arms (respectively) for the first type of robot, l_2 and a_2 are those for the second type, and l_t and a_t are the total number of observed legs and arms. All values are in the range $[1, 10\,000]$.

Output

For each test case output two positive integers denoting the number of each of the two types of robots. Give first the count for the first type listed. If the test case has no solution or multiple solutions, output "?" (a question mark).

Sample Input 1

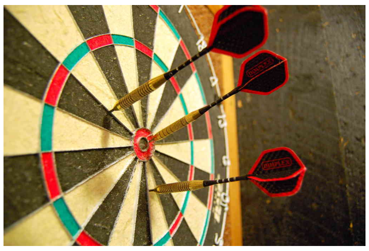
```
3
2 1 4 1 16 5
3 2 5 1 9 4
1 2 3 6 8 16
```

Sample Output 1

```
2 3
?
?
```

This page is intentionally left blank.

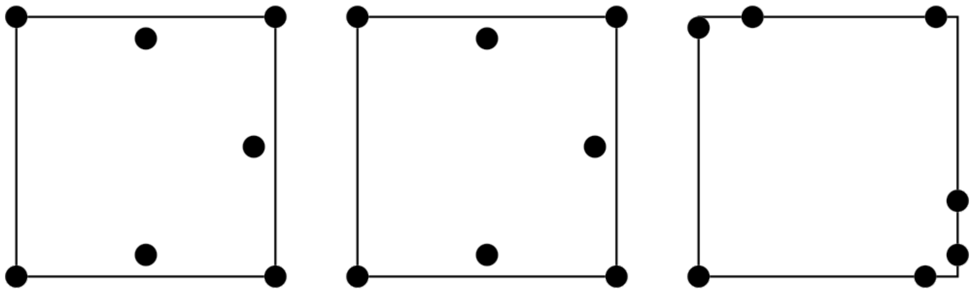
Problem L
Unusual Darts



In the game of Unusual Darts, Alice throws seven darts onto a 2-foot by 2-foot board, and then Bob may or may not throw three darts.

Alice's seven darts define a polygon by the order in which they are thrown, with the perimeter of the polygon connecting Dart 1 to Dart 2 to Dart 3 to Dart 4 to Dart 5 to Dart 6 to Dart 7, and back to Dart 1. If the polygon so defined is not simple (meaning it intersects itself) then Alice loses. If the polygon is simple, then Bob throws three darts. He is not a very good player, so although his darts always land on the board, they land randomly on the dart board following a uniform distribution. If all three of these darts land within the interior of the polygon, Bob wins, otherwise Alice wins.

For this problem you are given the locations of Alice's darts (which form a simple polygon) and the probability that Bob wins. Your job is to determine the order in which Alice threw her darts.


Sample Input 1 Sample Input 2 Sample Input 3

Input

The first line of input contains an integer N ($1 \leq N \leq 1\,000$), indicating the number of Darts games that follow. Each game description has 8 lines. Lines 1 through 7 each have a pair of real numbers with 3 digits after the decimal point. These indicate the x and y coordinates of Alice's seven darts (x_1 y_1 to x_7 y_7), which are all at distinct locations. All coordinates are given in feet, in the range ($0 \leq x_i, y_i \leq 2$). The 8th line contains a real number p with 5 digits after the decimal point, giving the probability that Bob wins. In all test cases, Alice's darts do form a simple polygon, but not necessarily in the order given.

Output

For each Darts game, output the order in which the darts could have been thrown, relative to the order they were given in the input, so that Bob wins with probability p. If several answers are possible, give the one that is lexicographically least. Any ordering that would give Bob a probability of winning within 10^{-5} of the given value of p is considered a valid ordering.

Sample Input 1

```
3
0.000 0.000
0.000 2.000
1.000 1.800
1.000 0.200
1.800 1.000
2.000 0.000
2.000 2.000
0.61413
0.000 0.000
0.000 2.000
1.000 1.800
1.000 0.200
1.800 1.000
2.000 0.000
2.000 2.000
0.12500
0.000 0.000
0.000 1.900
0.400 2.000
1.700 0.000
1.800 2.000
2.000 0.200
2.000 0.600
0.86416
```

Sample Output 1

```
1 2 3 7 5 6 4
1 4 3 2 7 5 6
1 2 3 5 7 6 4
```