Problem A: A Journey to Greece

For a long time Tim wanted to visit Greece. He has already purchased his flight to and from Athens. Tim has a list of historical sites he wants to visit, e.g., Olympia and Delphi. However, due to recent political events in Greece, the public transport has gotten a little complicated. To make the Greek happy and content with their new government, many short-range bus and train lines have been created. They shall take the citizens around in their neighborhoods, to work or to their doctor. At the same time, long-range trains that are perfect for tourists have been closed down as they are too expensive. This is bad for people like Tim, who really likes to travel by train. Moreover, he has already purchased the Greece' Card for Public Conveyance (GCPC) making all trains and buses free for him.

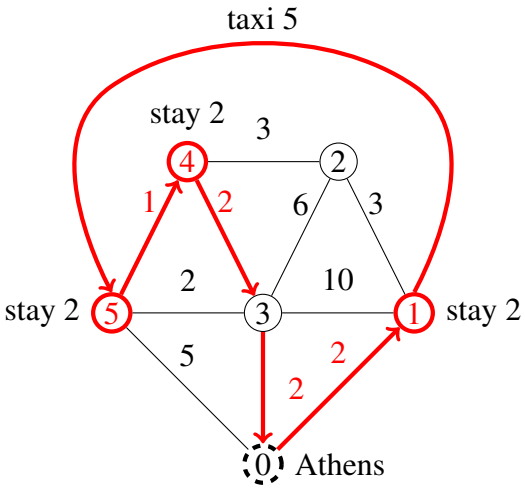


Figure A.1: Visual representation of the Sample Input: Tim's tour has length 18.

Despite his preferred railway lines being closed down, he still wants to make his travel trough Greece. But taking all these local bus and train connections is slower than expected, so he wants to know whether he can still visit all his favorite sites in the timeframe given by his flights. He knows his schedule will be tight, but he has some emergency money to buy a single ticket for a special Greek taxi service. It promises to bring you from any point in Greece to any other in a certain amount of time.
For simplicity we assume, that Tim does never have to wait for the next bus or train at a station. Tell Tim, whether he can still visit all sites and if so, whether he needs to use this taxi ticket.

Input

The first line contains five integers N, P, M, G and T, where N denotes the number of places in Greece, P the number of sites Tim wants to visit, M the number of connections, G the total amount of time Tim can spend in Greece, and T the time the taxi ride takes ($1 \leq N \leq 2 \cdot 10^4$; $1 \leq P \leq 15$; $1 \leq M, G \leq 10^5$; $1 \leq T \leq 500$).
Then follow P lines, each with two integers p_i and t_i, specifying the places Tim wants to visit and the time Tim spends at each site ($0 \leq p_i < N$; $1 \leq t_i \leq 500$). The sites p_i are distinct from each other.
Then follow M lines, each describing one connection by three integers s_i, d_i and t_i, where s_i and d_i specify the start and destination of the connection and t_i the amount of time it takes ($0 \leq s_i, d_i < N$; $1 \leq t_i \leq 500$).
All connections are bi-directional. Tim's journey starts and ends in Athens, which is always the place 0.

Output

Print either "`impossible`", if Tim cannot visit all sites in time, "`possible without taxi`", if he can visit all sites without his taxi ticket, or "`possible with taxi`", if he needs the taxi ticket.

Sample Input 1

```
6 3 10 18 5
1 2
4 2
5 2
0 1 2
1 2 3
2 4 3
1 3 10
2 3 6
0 3 2
3 4 2
4 5 1
3 5 2
0 5 5
```

Sample Output 1

```
possible with taxi
```

Problem B: Bounty Hunter II

Spike the bounty hunter is tracking another criminal through space. Luckily for him hyperspace travel has made the task of visiting several planets a lot easier. Each planet has a number of Astral Gates; each gate connects with a gate on another planet. These hyperspace connections are, for obvious safety reasons, one-way only with one gate being the entry point and the other gate being the exit point from hyperspace. Furthermore, the network of hyperspace connections must be loop-free to prevent the Astral Gates from exploding, a tragic lesson learned in the gate accident of 2022 that destroyed most of the moon.
While looking at his star map Spike wonders how many friends he needs to conduct a search on every planet. Each planet should not be visited by more than one friend otherwise the criminal might get suspicious and flee before he can be captured. While each person can start at a planet of their choosing and travel along the hyperspace connections from planet to planet they are still bound by the limitations of hyperspace travel.

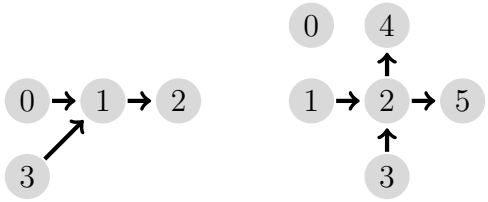


Figure B.1: Illustration of the Sample Inputs.

Input

The input begins with an integer N specifying the number of planets ($0 < N \leq 1000$). The planets are numbered from 0 to $N-1$. The following N lines specify the hyperspace connections. The i-th of those lines first contains the count of connections K ($0 \leq K \leq N-1$) *from* planet i followed by K integers specifying the destination planets.

Output

Output the minimum number of persons needed to visit every planet.

Sample Input 1

```
4
1 1
1 2
0
1 1
```

Sample Output 1

```
2
```

Sample Input 2

```
6
0
1 2
2 4 5
1 2
0
0
```

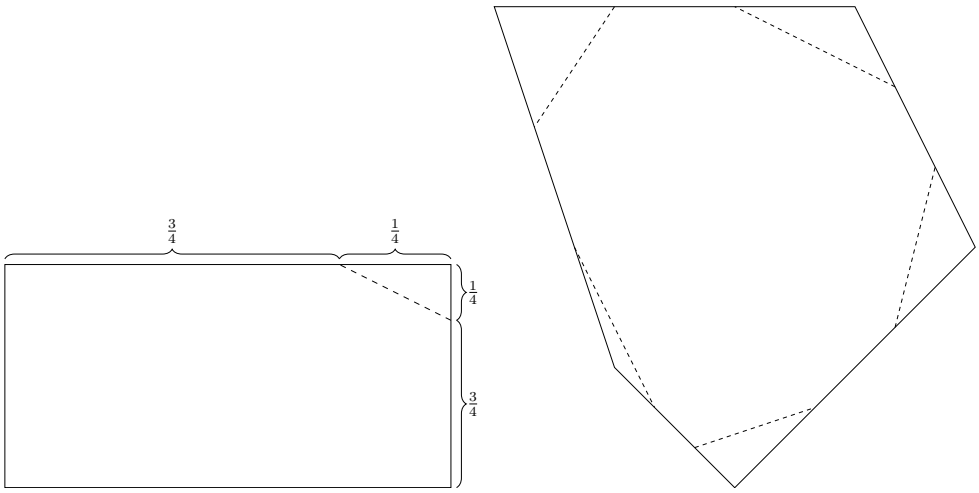
Sample Output 2

```
4
```

Problem C: Cake

Sophie loves to bake cakes and share them with friends. For the wedding of her best friend Bea she made a very special cake using only the best ingredients she could get and added a picture of the engaged couple on top of the cake. To make it even more special she did not make it round or square, but made a custom convex shape for the cake. Sophie decided to send the cake by a specialized carrier to the party. Unfortunately, the cake is a little too heavy for their default cake package and the overweight fees are excessive. Therefore, Sophie decides to remove some parts of the cake to make it a little lighter.

Sophie wants to cut the cake the following way: First, she chooses a real number $s \geq 2$. For each vertex and each incident edge of the cake she marks where $1/s$ of the edge's length is. Afterwards, she makes a direct cut between the two markings for each vertex and removes the vertex that way.



(a) Cutting the upper-right corner of a rectangle with $s = 4$

(b) Cutting a cake with $s = 3$

Figure C.1: Illustration of the first two Sample Inputs.

Sophie does not want to cut more from the cake than necessary for obvious reasons. Can you tell her how to choose s?

Input

The first line contains a floating point number a and an integer N, where a denotes the ratio of the cake's weight allowed by the carrier and N the number of vertices of the cake ($0.25 \leq a < 1$; $3 \leq N \leq 100$). a will be specified with at most 7 digits after the decimal point.

Then follow N lines, each describing one of the cake's vertices with two integers x_i and y_i, the coordinates of the vertex ($0 \leq x_i, y_i \leq 10^8$ for all $1 \leq i \leq N$). The vertices are given in the order in which they form a strictly convex shape.

You may safely assume that the weight is uniformly distributed over the area of the cake. Furthermore, it will always be possible to cut the cake with some $2 \leq s \leq 1000$ such that the proportion of the remaining cake is a of the original weight.

Output

Print a line containing s, the biggest value as specified above such that the remaining cake's weight is at most the proportion a of its original weight.

Your answer will be considered correct if the absolute error is at most 10^{-4}.

Sample Input 1

```
0.875 4
0 0
8 0
8 4
0 4
```

Sample Output 1

```
4.000000
```

Sample Input 2

```
0.85 5
6 0
12 6
9 12
0 12
3 3
```

Sample Output 2

```
3.000000
```

Sample Input 3

```
0.999998 4
20008 10000
15004 15005
10001 20009
15005 15004
```

Sample Output 3

```
1000.000000
```

Problem D: Carpets

The computer science Professor Toving Liles loves the floor tiles in his office so much that he wants to protect them from damage by careless students. Therefore, he would like to buy cheap small rectangular carpets from the supermarket and cover the floor such that:

1. The entire floor is covered.
2. The carpets do not overlap.
3. The carpets are rotated arbitrarily.
4. No carpet is cut into pieces.

But when checking the supermarket's stock he begins to wonder whether he can accomplish his plan at all. Can you help him?

Input

The first line contains two integers W and H describing the size of his room ($1 \leq W, H \leq 100$).
The second line contains an integer c, denoting the number of different carpet colors the supermarket has in stock ($1 \leq c \leq 7$).
Each of the following c lines consists of three integers a_i, w_i, and h_i, which means: the supermarket's stock contains a_i carpets of size w_i, h_i and color i ($1 \leq a_i \leq 7$; $1 \leq w_i \leq 100$; $1 \leq h_i \leq 100$).
The supermarket has at most 7 carpets, i.e. $\sum_i a_i \leq 7$.

Output

For the given room dimensions and the supermarket's stock of carpets, print "`yes`" if it is possible to cover the room with carpets as specified above and "`no`" otherwise.

Sample Input 1

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

Sample Output 1

```
yes
```

Sample Input 2

```
100 100
3
4 42 42
1 100 16
1 32 42
```

Sample Output 2

```
no
```

Problem E: Change of Scenery

Every day you drive to work using the same roads as it is the shortest way. This is efficient, but over time you have grown increasingly bored of seeing the same buildings and junctions every day. So you decide to look for different routes. Of course you do not want to sacrifice time, so the new way should be as short as the old one. Is there another way that differs from the old one in at least one street?

Input

The first line of the input starts with three integers N M and K, where N is the number of junctions and M is the number of streets in your city, and K is the number of junctions you pass every day ($1 \leq K \leq N \leq 10\,000,\ 0 \leq M \leq 1\,000\,000$).
The next line contains K integers, the (1-based) indices of the junctions you pass every day. The first integer in this line will always be 1, the last integer will always be N. There is a shortest path from 1 to N along the K junctions given.
M lines follow. The i-th of those lines contains three integers a_i b_i c_i and describes a street from junction a_i to junction b_i of length c_i ($1 \leq a_i, b_i \leq N,\ 1 \leq c_i \leq 10\,000$). Streets are always undirected.
Note that there may be multiple streets connecting the same pair of junctions. The shortest path given uses for every pair of successive junctions a and b a street of minimal length between a and b.

Output

Print one line of output containing "`yes`" if there is another way you can take without losing time, "`no`" otherwise.

Sample Input 1

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

Sample Output 1

```
yes
```

Sample Input 2

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

Sample Output 2

```
no
```

Problem F: Divisions

David is a young boy and he loves numbers. Recently he learned how to divide two numbers. David divides the whole day. He is happy if the result of the division is an integer, but he is not very amused if this is not the case. After quite a while he decided to use only a single dividend each day.

The parents of David are very careful and they would like to ensure that David experiences enough happiness. Therefore they decide which number David will use as the dividend for this day.

There is still a problem: The parents are not very good at math and don't know how to calculate the number of positive integral divisors for a given dividend N, which lead to an integral result. Now it's up to you to help David's parents.

Input

The single input line contains the single integer N, where N is chosen as a dividend ($1 \leq N \leq 10^{18}$).

Output

Print the number of positive integral divisors of N that lead to an integral result of the division.

Sample Input 1

```
12
```

Sample Output 1

```
6
```

Sample Input 2

```
999999999999999989
```

Sample Output 2

```
2
```

Sample Input 3

```
1000000077000000049
```

Sample Output 3

```
4
```

Problem G: Extreme Sort

John likes sorting algorithms very much. He has studied quicksort, merge sort, radix sort, and many more.
A long time ago he has written a lock-free parallel string sorting program. It was a combination of burstsort and multi-key quicksort. To implement burstsort you need to build a tree of buckets. For each input string you walk through the tree and insert part of the string into the right bucket. When a bucket fills up, it "bursts" and becomes a new subtree (with new buckets).

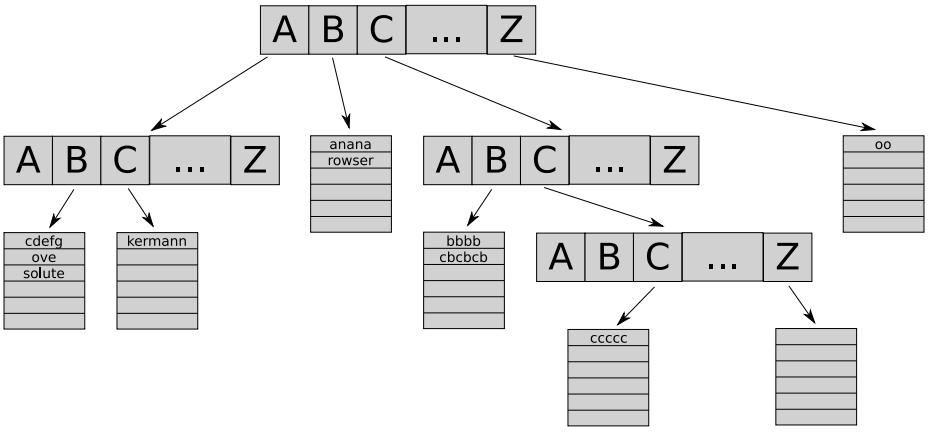


Figure G.1: Burstsort data structure

Well, enough about the past. Today John is playing with sorting algorithms again. This time it's numbers. He has an idea for a new algorithm, "extreme sort". It's extremely fast, performance levels are OVER NINETHOUSAND. Before he tells anyone any details, he wants to make sure that it works correctly.
Your task is to help him and verify that the so-called *extreme property* holds after the first phase of the algorithm. The extreme property is defined as $\min(x_{i,j}) \geq 0$, where

$$x_{i,j} = \begin{cases} a_j - a_i & \text{for} \quad 1 \leq i < j \leq N \\ 9001 & \text{otherwise} \end{cases}$$

Input

The first line contains a single integer N ($1 \leq N \leq 1024$). The second line contains N integers $a_1\ a_2\ \ldots\ a_N$ ($1 \leq a_i \leq 1024$).

Output

Print one line of output containing "`yes`" if the extreme property holds for the given input, "`no`" otherwise.

Sample Input 1

```
2
1 2
```

Sample Output 1

```
yes
```

Sample Input 2

```
4
2 1 3 4
```

Sample Output 2

```
no
```

Problem H: Legacy Code

Once again you lost days refactoring code, which never runs in the first place. Enough is enough – your time is better spent writing a tool that finds unused code!

Your software is divided into packages and executables. A package is a collection of methods. Executables are packages defining among other methods exactly one method with name `PROGRAM`. This method is executed on the start of the corresponding executable. Ordinary packages have no method named `PROGRAM`.

Each method is uniquely identified by the combination of package and method names. E.g. the method with the identifier `SuperGame::PROGRAM` would be the main method of the executable `SuperGame`.

For every method in your software you are given a list of methods directly invoking it. Thus you can easily identify methods, that are never called from any method. However, your task is more challenging: you have to find unused methods. These are methods that are never reached by the control flow of any executable in your software.

Input

The first line of the input contains an integer N, the number of methods in your software ($1 \leq N \leq 400$).

Each method is described by two lines, totaling in $2 \cdot N$ lines. The first line consists of the unique identifier of the method and k_i, the number of methods directly invoking this one ($0 \leq k_i \leq N$). The second line consists of a set of k_i identifiers of these calling methods or is empty if there are no such methods, i.e. $k_i = 0$.

Method identifiers consist of a package name followed by two colons and a method name like `Packagename::Methodname`. Both strings, the package and the method name, each consist of up to 20 lowercase, uppercase characters or digits (`a-z`, `A-Z`, `0-9`).

There will be exactly N different method identifiers mentioned in the input.

Output

A line containing the number of unused methods in your software.

Sample Input 1

```
2
SuperGame::PROGRAM 0

HelpPackage::HelpFunction 2
HelpPackage::HelpFunction SuperGame::PROGRAM
```

Sample Output 1

```
0
```

Sample Input 2

```
2
Loop::CallA 1
Loop::CallB
Loop::CallB 1
Loop::CallA
```

Sample Output 2

```
2
```

Sample Input 3

```
2
SuperGame::PROGRAM 1
SuperServer42::PROGRAM
SuperServer42::PROGRAM 1
SuperServer42::PROGRAM
```

Sample Output 3

```
0
```

Greater New York Region 2015

acm International Collegiate Programming Contest

Event Sponsors

IBM QUEENS COLLEGE

I • Farey Sequence Length

Given a positive integer, ***N***, the sequence of all fractions ***a*** / ***b*** with (***0*** < ***a*** ≤ ***b***), (***1*** < ***b*** ≤ ***N***) and ***a*** and ***b*** relatively prime, listed in increasing order, is called the *Farey Sequence of order* ***N***.

For example, the *Farey Sequence of order* ***6*** is:

0/1, **1/6**, **1/5**, **1/4**, **1/3**, **2/5**, **1/2**, **3/5**, **2/3**, **3/4**, **4/5**, **5/6**, **1/1**

For this problem, you will write a program to compute the length of the *Farey sequence of order* ***N*** (input).

Input

The first line of input contains a single integer P, ($1 \le P \le 10000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, followed by the order N, N ($2 <= N <= 10000$), of the *Farey Sequence* whose length is to be found.

Output

For each data set there is a single line of output. The single output line consists of the data set number, K, followed by a single space followed by the length of the *Farey Sequence* as a decimal integer.

Sample Input	Sample Output
4 1 6 2 15 3 57 4 9999	1 13 2 73 3 1001 4 30393487

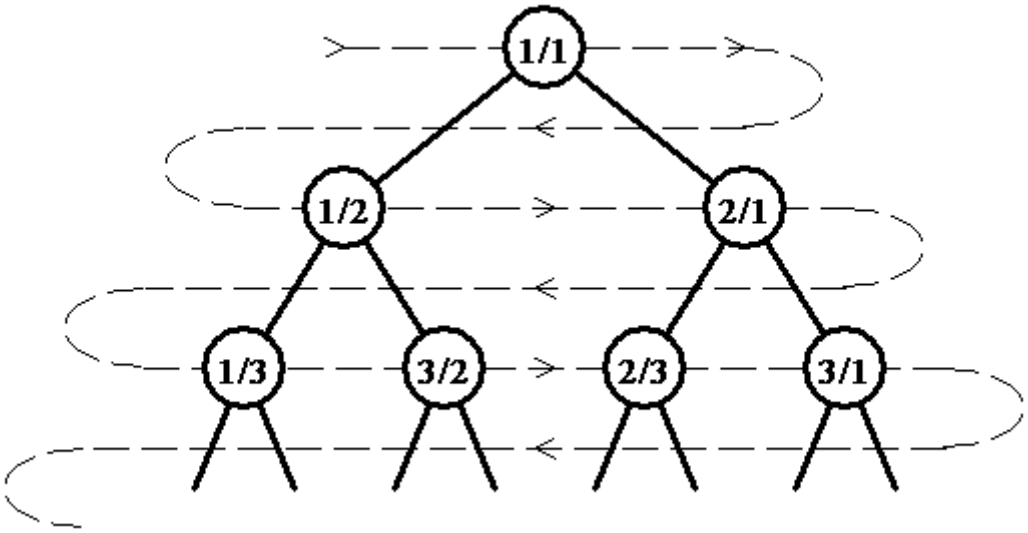
J • A Rational Sequence

A sequence of positive rational numbers is defined as follows:

An infinite full binary tree labeled by positive rational numbers is defined by:

- The label of the root is `1/1`.
- The left child of label `p/q` is `p/(p+q)`.
- The right child of label `p/q` is `(p+q)/q`.

The top of the tree is shown in the following figure:



The sequence is defined by doing a level order (breadth first) traversal of the tree (indicated by the light dashed line). So that:

`F(1) = 1/1, F(2) = 1/2, F(3) = 2/1, F(4) = 1/3, F(5) = 3/2, F(6) = 2/3, …`

Write a program which finds the value of ***n*** for which `F(n)` is **p/q** for inputs **p** and **q**.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, ***K,*** a single space, the numerator, **p**, a forward slash (*/*) and the denominator, **q**, of the desired fraction.

Output

For each data set there is a single line of output. It contains the data set number, K, followed by a single space which is then followed by the value of ***n*** for which F(**n)** is **p/q**. Inputs will be chosen so n will fit in a 32-bit integer.

Sample Input	Sample Output
4 1 1/1 2 1/3 3 5/2 4 2178309/1346269	1 1 2 4 3 11 4 1431655765

Problem K

Magic Trick

Your friend has come up with a math trick that supposedly will blow your mind. Intrigued, you ask your friend to explain the trick.

First, you generate a random positive integer, k, between 1 and 100 inclusive. Then, your friend will give you n operations to execute. An operation consists of one of the four arithmetic operations `ADD`, `SUBTRACT`, `MULTIPLY`, or `DIVIDE`, along with an integer-valued operand x. You are supposed to perform the requested operations in order.

You don't like dealing with fractions or negative numbers though, so if during the process, the operations generate a fraction or a negative number, you will tell your friend that he messed up.

You know the n operations your friend will give. How many of the first 100 positive integers, i.e. 1 to 100, will cause your friend to mess up?

Input

The first line of input contains a single positive integer n ($1 \leq n \leq 10$). Each of the next n lines consists of an operation, followed by an operand. The operation is one of the strings `ADD`, `SUBTRACT`, `MULTIPLY`, or `DIVIDE`. Operands are positive integers not exceeding 5.

Output

Print, on a single line, a single integer indicating how many of the first 100 positive integers will result in you telling your friend that he messed up.

Sample Input and Output

Sample Input 1	Output for Sample Input
1 SUBTRACT 5	4

Sample Input 2	Output for Sample Input
1 DIVIDE 2	50

Sample Input 3	Output for Sample Input
2 ADD 5 DIVIDE 5	80

L: Post Office

Other than postcards, the post office department of some country recognizes three classes of mailable items: letters, packets and parcels. The three dimensions of a mailable item are called length, height and thickness, with length being the largest and thickness the smallest of the three dimensions.

A letter's length must be at least 125 mm but not more than 290 mm, its height at least 90 mm but not more than 155 mm and its thickness at least 0.25 mm but not more than 7 mm.

All three of a packet's dimensions must be greater than or equal to the corresponding minimum dimension for a letter and at least one of its dimensions must exceed the corresponding maximum for a letter. Furthermore, a packet's length must be no more than 380 mm, its height no more than 300 mm and its thickness no more than 50 mm.

All three of a parcel's dimensions must be greater than or equal to the corresponding minimum dimension for a letter and at least one of its dimensions must exceed the corresponding maximum for a packet. Furthermore, the parcel's combined length and girth may not exceed 2 100 mm. The girth is the full perimeter measured around the parcel, perpendicular to the length.

Input

The input will contain data for a number of problem instances. For each problem instance, the input will consist of the three dimensions of an item, measured in mm, in any order. The length and width will be positive integers. The thickness will be either a positive integer or a positive floating point number.

The input will be terminated by a line containing three zeroes.

Output

For each problem instance, output the classification of the item on a single line as `letter`, `packet`, `parcel` or `not mailable`. Use only lower case letters.

Sample Input and Output

Sample Input 1	Output for Sample Input
100 120 100 0.5 100 200 100 10 200 200 75 100 0 0 0	not mailable letter packet parcel

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

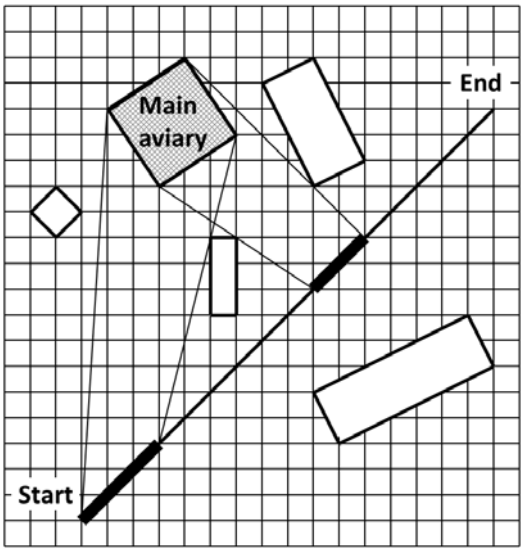
CTU Open Contest 2015

Problem M

Visitors' Train

An aviary or a flight cage is a big cage for birds. An usual ZOO aviary typically measures tens of meters in diameter. In the aviaries, the birds can fly around and live in conditions imitating the conditions in the wild as closely as possible. At least in theory. There is one main big and spectacular aviary in the ZOO and some other less important ones.

The ZOO is planning to build a short straight electric train track to help visitors to move easily from one part of the ZOO to another. It has to be decided which of the free areas of the ZOO will the track run through. The director had noticed during his trips to other ZOOs that the visitors are more happy when they can take more photos of important ZOO structures. Now he wants to measure the quality of the planned railway by this parameter. The most important structure in the vicinity of the track will be the main aviary. The director worries that the main aviary might be obscured by the less important aviaries along the track and the visitors might be less happy. Help the director to assess the quality of the planned track.



You are given the coordinates of all aviaries. Also, you are given the coordinates of the start and the end of the planned railway track. Find the total length of the segments on the track from which the main aviary is visible and is not obscured, even not partially, by any other aviary. We suppose that the visitors can look out from the train in any direction.

Input Specification

There are more test cases. Each case occupies more lines. The first line contains number N ($1 \leq N \leq 100$) of the aviaries. Next line contains the coordinates of the planned railway track in the format $x_1\ y_1\ x_2\ y_2$ where $[x_1, y_1]$ and $[x_2, y_2]$ are the coordinates of the start and the end of the track. The track is considered to be infinitely thin in this representation. Next, there are N

lines specifying the aviaries, each aviary is represented as a rectangle with nonzero area. Each of these lines specifies the coordinates of an aviary in the form $x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4$, where $[x_1, y_1]$, $[x_2, y_2]$, $[x_3, y_3]$, and $[x_4, y_4]$ are the coordinates of the aviary corners. The corners are presented in clockwise or anti-clockwise order. The main aviary is listed first. All coordinates are integers, their absolute value is less than 10 000. You may assume that no aviary intersects or touches the track or another aviary. There is no blank line between consecutive test cases. The input is terminated by a line with one zero.

Output Specification

For each test case print on a separate line the total length L of all segments of the planned track from which the main aviary is visible and it is not obscured, even not partially, by any other aviary. Your answer should not differ from the correct answer by more than 10^-4.

As shown in the third Sample Input, the main aviary is not considered obscured if *only its corners/edges are hidden.*

Sample Input

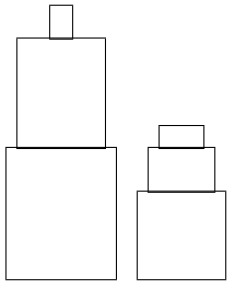
```
5
3 1 17 15
6 14 4 17 7 19 9 16
2 12 1 13 2 14 3 13
8 9 8 12 9 12 9 9
12 14 10 18 12 19 14 15
12 6 18 9 19 7 13 4
1
0 0 0 2
4 -1 4 1 5 1 5 -1
2
0 0 0 1
4 0 4 2 5 2 5 0
2 0 3 0 3 -1 2 -1
2
0 0 0 1
4 0 4 2 5 2 5 0
2 0 3 0 3 1 2 1
0
```

Output for Sample Input

```
7.07105
2.00
1.00
0.00
```

Problem N
A Towering Problem

You've been put in charge of an art exhibit from the famous minimalist sculptor J (even his name is minimalist!). J's work involves the careful layout of vertically dispositioned orthogonal parallelpipeds in a set of tapering obelisks — in other words, he puts smaller boxes on top of larger boxes. His most recent triumph is called "2 by 3's Decreasing," in which he has various sets of six boxes arranged in two stacks of three boxes each. One such set is shown below:



J has sent you the art exhibit and it is your job to set up each of the six-box sets at various locations throughout the museum. But when the sculptures arrived at the museum, uncultured barbarians (i.e., delivery men) simply dropped each set of six boxes on the floor, not realizing the aesthetic appeal of their original layout. You need to reconstruct each set of two towers, but you have no idea which box goes on top of the other! All you know is the following: for each set of six, you have the heights of the two towers, and you know that in any tower the largest height box is always on the bottom and the smallest height box is on the top. Armed with this information, you hope to be able to figure out which boxes go together before tomorrow night's grand opening gala.

Input

The input consists of eight positive integers. The first six represent the heights of the six boxes. These values will be given in no particular order and no two will be equal.

The last two values (which will never be the same) are the heights of the two towers.

All box heights will be ≤ 100 and the sum of the box heights will equal the sum of the tower heights.

Output

Output the heights of the three boxes in the first tower (i.e., the tower specified by the first tower height in the input), then the heights of the three boxes in the second tower. Each set of boxes should be output in order of decreasing height. Each test case will have a unique answer.

Sample Input 1	Sample Output 1
12 8 2 4 10 3 25 14	12 10 3 8 4 2

Sample Input 2	Sample Output 2
12 17 36 37 51 63 92 124	63 17 12 51 37 36