Problem A. Help a PhD candidate out!

Input file:	add-help.in
Output file:	add-help.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Jon Marius forgot how to add two numbers while doing research for his PhD. And now he has a long list of addition beginproblems that he needs to solve, in addition to his computer science ones! Can you help him?

On his current list Jon Marius has two kinds of beginproblems: addition beginproblems on the form "$a + b$"and the ever returning beginproblem "P=NP". Jon Marius is a quite distracted person, so he might have so solve this last beginproblem several times, since he keeps forgetting the solution. Also, he would like to solve these beginproblems by himself, so you should skip these.

Input

The first line of input will be a single integer N ($1 \leq N \leq 1000$) denoting the number of testcases.
Then follow N lines with either "P=NP"or an addition beginproblem on the form "$a + b$where $a, b \in [0, 1000]$ are integers.

Output

OutputFile the result of each addition. For lines containing “P=NP”, output “skipped”.

Example

add-help.in	add-help.out
4 2+2 1+2 P=NP 0+0	4 3 skipped 0

Problem B. Borg Boogie

Input file:	borg.in
Output file:	borg.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Being a space captain can be a dangerous endeavour, especially when conducting missions in Borg space. Today your captain has been tasked with beaming aboard a Borg spaceship to see if he can discover anything useful. The Borg ship consist of rooms connected with passages of equal length, and with no more than one direct passage between any two rooms. It is possible to find a path from any room to any other.

On the Borg ship there is also a sentry, placed there by the Borg eons ago. This sentry operates according to a rather unsophisticated algorithm, namely walking at random. This gives the sentry one advantage, however: You will never know where it is! More precisely, once every minute, the sentry chooses one of the neighbouring rooms with uniform probability, and walks quickly to this room. Your captain will also be spending exactly one minute in each room, and to minimise the chances that he will meet the sentry, you time his movement such that he and the sentry move at exactly the same time. Thus he will be caught by the sentry if and only if the two of them move into the same room at the same time, or if they try to swap rooms.

Star Fleet has issued as a direct order the rooms your captain is to visit, and in the exact order. These rooms form a walk on the Borg ship, where each room may be visited several times. Star Fleet has also provided you with a map of the ship. The captain will be beamed down to the first room of the walk, and will be beamed back up from the last room. He risks capture in both these rooms.

Now the captain starts wondering about the chances of success on this mission, hoping them to be very low indeed. After all, that makes it all the more interesting! Getting curious yourself, you decide you want to figure this out. Unfortunately the ship’s android is experiencing beginproblems with his new emotion chip, and thus the task falls on you to compute the captain’s chances of success on this dangerous mission!

Input

One line with $2 \leq N \leq 500$ – the number of nodes.
One line with $1 \leq L \leq 500$ – the number of rooms the captain must visit.
One line with L numbers describing the captain’s walk, i.e. they give the exact walk the captain must perform.
N lines beginning with an integer n_i – the number of neighbours of node i – followed by n_i numbers – the neighbours of node i, 0-indexed.

Output

The chance the captain will be able to complete his mission without being discovered by the Borg sentry.

Examples

borg.in	borg.out
3 1 0 2 1 2 1 0 1 0	0.5

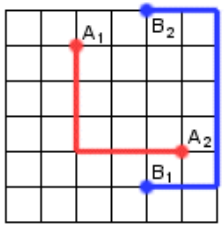
borg.in	borg.out
8 6 1 0 2 3 0 1 7 1 2 3 4 5 6 7 1 0 2 0 3 2 0 2 1 0 1 0 1 0 1 0	0.04464285714285711

borg.in	borg.out
8 7 1 0 2 3 2 0 1 7 1 2 3 4 5 6 7 1 0 2 0 3 2 0 2 1 0 1 0 1 0 1 0	0.177615

Problem C. Connect

Input file:	connect.in
Output file:	connect.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

When constructing electric circuits one has to connect pairs of points using wire, preferable as short as possible. In this beginproblem we have an empty circuit board of size $N \times M$ where we want to connect the two points A_1 and A_2 with each other using one wire, and the two points B_1 and B_2 with each other using another wire. The wires must go along the horizontal and vertical edges of the grid (see figure), and the two wires may not share a common vertex. Determine the minimum length of wire needed to do so. The wire may not go outside the circuit board.



Input

The first line contains two integers, N ($2 \leqslant N \leqslant 100$) and M ($2 \leqslant M \leqslant 100$), the grid size of the circuit board.
Then follows four lines containing the coordinates for the points A_1, A_2, B_1 and B_2, respectively. Each coordinate pair will be described using two integers and will correspond to an intersection point in the grid. The first coordinate will be between 0 and N inclusive and the second coordinate between 0 and M inclusive. All coordinate pairs will be unique.

Output

A single line containing the minimum length of wire needed to connect the points, or "IMPOSSIBLE"if it's not possible to do so.

Examples

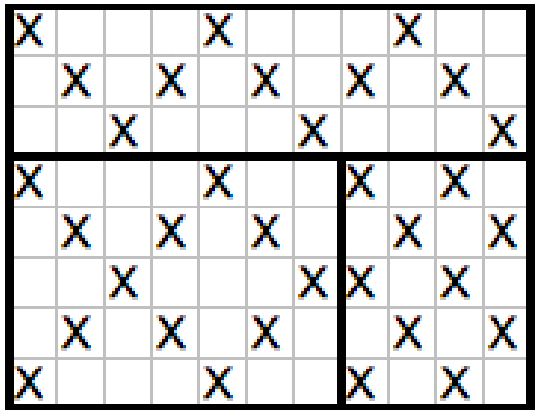
connect.in	connect.out
6 6 2 1 5 4 4 0 4 5	15

connect.in	connect.out
6 3 2 3 4 0 0 2 6 1	IMPOSSIBLE

Problem D. Doodling

Input file:	doodling.in
Output file:	doodling.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

When thinking about a hard beginproblem a lot of people like to doodle, to create "an unfocused drawing that can help the memory and improve abstract thinking". The most basic form of doodle is a repetitive pattern covering the whole page. One way to create such a pattern is to take a graphing paper and start in the top-left corner $(0, 0)$ and fill out the square, then move down and right one square $(1, 1)$, fill it out, and so on. Every time you hit the edge of the paper you reverse direction, until you are back at the starting point. This will create a very soothing pattern.



However, to ensure you don't spend the whole competition doodling you need to figure out how how many squares you will have to fill in the paper to complete the doodle before you even start doodling.

Input

$1 \leq n \leq 4000$
The number of testcases
For each n:
$2 \leq x, y \leq 20000$
The height and width (in squares) of the graphing paper.

Output

The number of unique squares you will have filled in before you are done with your doodle.

Example

doodling.in	doodling.out
9 2 2 2 20000 20000 2 19000 19001 9000 9000 2001 39 2001 41 2001 43 3010 3001	2 20000 20000 180509500 9000 29010 2001 32011 2508502

Problem E. Dirty Driving

Input file:	driving.in
Output file:	driving.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Like all other good drivers, you like to curse, swear and honk your horn at your fellow automobile drivers. Today you're at the rear of a long line, brooding over the others' inability to keep proper distance to the car in front. But are you really keeping your own distance?

You have calculated that in order to never have to use your breaks, you must keep a distance to any car x in front of you at least $p(n+1)$ where n is the number of cars between you and x, and p is an integer constant determined by which of your cars you are currently driving.

Given the value of p and the current distances (in random order) to each of the cars ahead of you, calculate the minimum distance you *should* be keeping to the car directly in front, in order to not have to use your breaks.

Input

A line with $1 \leq n \leq 100000$ – the number of cars ahead of you – and $1 \leq p \leq 20$ – the deceleration constant.
A single line with n unique integers denoting the current distance to each of the cars ahead of you. Each of these integers are in the interval $[1, 10^7]$

Output

The minimum distance you must keep to the car directly in front, in order to not have to use your breaks.

Examples

driving.in	driving.out
3 1 1 2 4	1
6 3 2 3 4 5 6 1	13

Problem F. Great Geek Game-show 3000!

Input file:	gameshow.in
Output file:	gameshow.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Yes! You've finally been chosen to participate in the "Great Geek Game-show 3000". This is the moment you've been waiting for, ever since you puzzled out how to maximise your chances of winning. You will finally be rich, and able to buy all the algorithm books you want! Of course you will have to share the winnings with the other contestants, but since your algorithm is vastly superior to the randomness of all the other numb-skulls you are certain you will be able to keep most of the prize money for yourself, in exchange for telling them how you can all maximise your chances of winning.

The rules of the game are the following: There is a stage with N boxes, each containing the name of one of the N contestants, and such that each contestant has their name in exactly one box. The contestants enter the stage one at a time, and each one is allowed to peek inside K of the boxes. If they find their own name inside one of these boxes they can get off the stage, and the game continues with the next contestant. If all contestants find their own name, everyone wins. But if one contestant fails to find theirs, everyone loses. After the game has begun, no communication between the contestants is allowed. However you are all allowed to agree upon a strategy before the game begins, and this is where you explain to all the others that the algorithm of everyone choosing K boxes at random is a very bad one, since it gives a chance of winning only equal to $\frac{K}{N}^N$. Instead you suggest the following algorithm:

Assign to each player and each box a unique number in the range $1, \ldots, N$. *Then each player starts with opening the box with the same number as themselves. The next box the player opens is the box whose number is found inside the first box, then the box whose number is found inside the second box, and so on. The process goes on until the player has opened* K *boxes, or found their own number.*

Now to bring home your point of how superior your algorithm is, you will need to calculate the exact odds of winning if all the contestants follow your directions. Unfortunately, this is the only thing you haven't figured out yet ...

Input

One line with the following numbers:
$1 \leq N \leq 10000000$ – the number of contestants.
$1 \leq K \leq N$ – the number of boxes each contestant may check.

Output

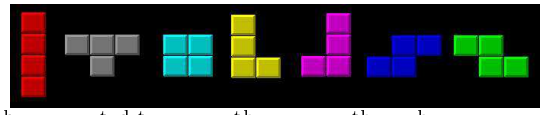
The chances you have of winning if everyone follows your algorithm.

Examples

gameshow.in	gameshow.out
2 1	0.500000
4 2	0.416667
6 5	0.833333
137 42	0.029351

Problem G. Polyomino Powers

Input file:	polyomino.in
Output file:	polyomino.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes



A polyomino is a polyform with the square as its base form. It is a connected shape formed as the union of one or more identical squares in distinct locations on the plane, taken from the regular square tiling, such that every square can be connected to every other square through a sequence of shared edges (i.e., shapes connected only through shared corners of squares are not permitted).
The most well-known polyominos are the seven tetrominos made out of four squares (see figure), famous from the Tetris® game, and of course the single domino consisting of two squares from the game with the same name. Some polyomino can be obtained by gluing several copies of the same smaller polyomino translated (but not rotated or mirrored) to different locations in the plane. We call those polyomino powers.

Input

One line with two positive integers $h, w \leq 10$.
Next follows an $h \times w$ matrix of characters '.' or 'X', the 'X"s describing a polyomino and '.' space.

Output

A k-power with $2 \leq k \leq 5$ copies of a smaller polyomino:
OutputFile a $h \times w$ matrix on the same format as the input with the 'X"s replaced by the numbers 1 through k in any order identifying the factor pieces.
Furthermore, if multiple solutions exist, any will do. Otherwise, output 'No solution' if no solution exists.

Examples

polyomino.in	polyomino.out
1 3 XXX	123
3 7 .XXXXX. .XX..X. XXXX...	No solution
4 9 .X..X.X.X .XX.X.X.X XXXXXXXXX .XX.....	.1..2.3.4 .15.2.3.4 115223344 .55.....

Problem H. Around the track

Input file:	roundtrip.in
Output file:	roundtrip.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

After The Stig's identity was revealed, the TV show Top Gear is in dire need of a new, tame racing driver to replace him. And of course you have been asked to take the job. However, you are not very fond of driving quickly, and especially not around the twisting and turning tracks they use in the show. To help you alleviate this beginproblem, one of your algorithmic friends has suggested that you should calculate the roundtrip with the least possible amount of turning required.

The driving track consists of unique, straight lines, and there are always exactly 2 or 4 roads heading out from each node. A roundtrip must be an Eulerian circuit, i.e. it must traverse each edge of the graph exactly once, and end up where it started. (Such a circuit is guaranteed to exist in the input graphs.) Thus the total amount of turning is the sum of the turning required at each node, where continuing in a straight line means a turn of 0. The roads on the track can be driven in any direction.

Input

One line with $3 \leq N \leq 10000$ – the number of nodes – and $N \leq M \leq 2N$ – the number of edges.
N lines with the x and y coordinates of each node, in order. $0 \leq x, y \leq 10000$. The nodes have unique coordinate pairs.
M lines with two space separated numbers i and j, denoting an edge between nodes i and j. The nodes are 0-indexed.

Output

The least amount of turning you must do to complete an Eulerian circuit, in radians.

Examples

roundtrip.in	roundtrip.out
3 3 0 0 0 1 1 0 0 1 0 2 1 2	6.283185

roundtrip.in	roundtrip.out
12 19 2 0 0 3 1 7 8 10 8 5 6 3 10 1 11 5 13 3 12 7 16 5 17 9 0 1 0 5 1 5 1 2 1 3 2 3 3 4 3 9 4 5 4 9 4 6 5 6 6 7 6 8 7 8 8 9 8 10 9 11 10 11	22.428486

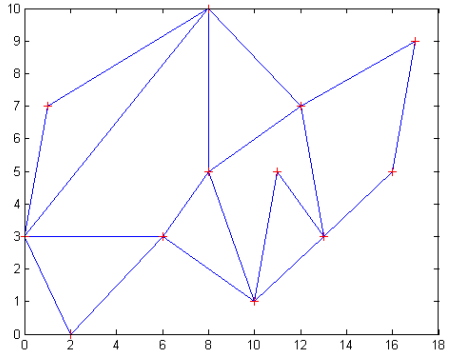


Рис. 1: Figure of the second sample case

Problem I. Skyline

Input file:	`skyline.in`
Output file:	`skyline.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Last time I visited Shanghai I admired its beautiful skyline. It also got me thinking, "Hmm, how much of the buildings do I actually see?"since the buildings wholly or partially cover each other when viewed from a distance.

In this beginproblem, we assume that all buildings have a trapezoid shape when viewed from a distance. That is, vertical walls but a roof that may slope. Given the coordinates of the buildings, calculate how large part of each building that is visible to you (i.e. not covered by other buildings).

Input

The first line contains an integer, N ($2 \leqslant N \leqslant 100$), the number of buildings in the city.
Then follows N lines each describing a building. Each such line contains 4 integers, x_1, y_1, x_2, and y_2 ($0 \leqslant x_1 < x_2 \leqslant 10000, 0 < y_1, y_2 \leqslant 10000$). The buildings are given in distance order, the first building being the one closest to you, and so on.

Output

For each building, output a line containing a floating point number between 0 and 1, the relative visible part of the building. The absolute error for each building must be $< 10^{-6}$.

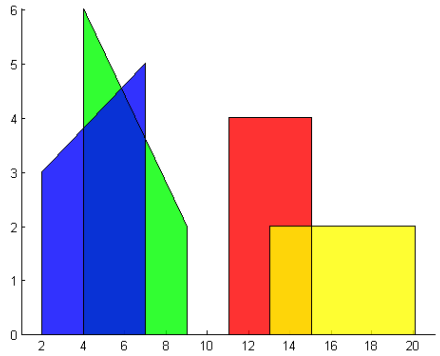


Рис. 2: Figure of the first sample case

Examples

skyline.in	skyline.out
4 2 3 7 5 4 6 9 2 11 4 15 4 13 2 20 2	1.00000000 0.38083333 1.00000000 0.71428571
5 200 1200 400 700 1200 1400 1700 900 5000 300 7000 900 8200 400 8900 1300 0 1000 10000 800	1.00000000 1.00000000 1.00000000 1.00000000 0.73667852

Problem J. Statisticians

Input file:	statisticians.in
Output file:	statisticians.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Statisticians like to create a lot of statistics. One simple measure is the mean value: the sum of all values divided by the number of values. Another is the median: the middle among all values when they have been sorted. If there are an even number of values, the mean of the two middle values will form the median.

These kinds of measures can be used for example to describe the population in a country or even some parts of the population in the country. Anne Jensen, Maria Virtanen, Jan Hansen, Erik Johansson and Jon Porsson want to find a statistical measurement of how many statisticians there are in the Nordic countries. To be more precise, they want to find out how many statisticians there are per unit area. As the population in the Nordic countries are well spread out they will try the new measurement MAD, Median of All Densities. First put a square grid on the map. Then draw a rectangle aligned with the grid and calculate the density of statisticians in that area, i.e. the mean number of statisticians per area unit. After that, repeat the procedure until all possible rectangles have been covered. Finally the MAD is the median of all statistician densities.

Input

The first line of the input contains of two space separated numbers h and w describing the height and width of the square grid, where $1 \leq h \leq 140$ and $1 \leq w \leq 120$. The next line contains two space separated numbers a and b which are the lower and upper bound of the allowed rectangle areas, i.e. $1 \leq a \leq rectanglearea \leq b \leq w \times h$. Then there will follow h lines with w space separated numbers s describing the number of statisticians in each square of the map, $0 \leq s \leq 10000$. There will always exist a rectangle which area is in $[a, b]$.

Output

The output contains of one line with the MAD. The number should be printed in number of statisticians per square and contain three decimals. The absolute error must be $< 10^{-3}$.

Examples

statisticians.in	statisticians.out
2 3 2 4 6 1 4 2 7 1	3.667
4 2 1 8 6 5 2 5 2 9 7 13	5.250

Problem K. Succession

Input file:	succession.in
Output file:	succession.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

The king in Utopia has died without an heir. Now several nobles in the country claim the throne. The country law states that if the ruler has no heir, the person who is most related to the founder of the country should rule.

To determine who is most related we measure the amount of blood in the veins of a claimant that comes from the founder. A person gets half the blood from the father and the other half from the mother. A child to the founder would have 1/2 royal blood, that child's child with another parent who is not of royal lineage would have 1/4 royal blood, and so on. The person with most blood from the founder is the one most related.

Input

The first line contains two integers, N ($2 \leqslant N \leqslant 50$) and M ($2 \leqslant M \leqslant 50$).
The second line contains the name of the founder of Utopia.
Then follows N lines describing a family relation. Each such line contains three names, separated with a single space. The first name is a child and the remaining two names are the parents of the child.
Then follows M lines containing the names of those who claims the throne.
All names in the input will be between 1 and 10 characters long and only contain the lowercase English letters 'a'-'z'. The founder will not appear among the claimants, nor be described as a child to someone else.

Output

A single line containing the name of the claimant with most blood from the founder. The input will be constructed so that the answer is unique.
The family relations may not be realistic when considering sex, age etc. However, every child will have two unique parents and no one will be a descendent from themselves. No one will be listed as a child twice.

Examples

succession.in	succession.out
9 2 edwardi charlesi edwardi diana philip charlesi mistress wilhelm mary philip matthew wilhelm helen edwardii charlesi laura alice laura charlesi helen alice bernard henrii edwardii roxane charlesii elizabeth henrii charlesii matthew	matthew
4 5 andrew betsy andrew flora carol andrew betsy dora andrew carol elena andrew dora carol dora elena flora gloria	elena