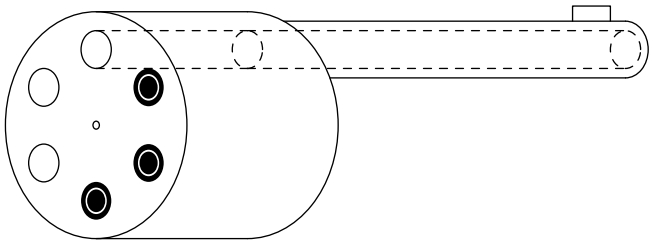
Problem A. Headshot

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

You have a revolver gun with a cylinder that has n chambers. Chambers are located in a circle on a cylinder. Each chamber can be empty or can contain a round. One chamber is aligned with the gun's barrel. When trigger of the gun is pulled, the gun's cylinder rotates, aligning the next chamber with the barrel, hammer strikes the round, making a shot by firing a bullet through the barrel. If the chamber is empty when the hammer strikes it, then there is no shot but just a "click".



You have found a use for this gun. You are playing Russian Roulette with your friend. Your friend loads rounds into some chambers, randomly rotates the cylinder, aligning a random chamber with a gun's barrel, puts the gun to his head and pulls the trigger. You hear "click" and nothing else — the chamber was empty and the gun did not shoot.

Now it is your turn to put the gun to your head and pull the trigger. You have a choice. You can either pull the trigger right away or you can randomly rotate the gun's cylinder and then pull the trigger. What should you choose to maximize the chances of your survival?

Input

The input file contains a single line with a string of n digits "0" and "1" ($2 \le n \le 100$). This line of digits represents the pattern of rounds that were loaded into the gun's chambers. "0" represent an empty chamber, "1" represent a loaded one. In this representation, when cylinder rotates before a shot, the next chamber to the right gets aligned with the barrel for a shot. Since the chambers are actually located on a circle, the first chamber in this string follows the last one. There is at least one "0" in this string.

Output

Write to the output file one of the following words (without quotes):

- "`SHOOT`" — if pulling the trigger right away makes you less likely to be actually shot in the head with the bullet (more likely that the chamber will be empty).
- "`ROTATE`" — if randomly rotating the cylinder before pulling the trigger makes you less likely to be actually shot in the head with the bullet (more likely that the chamber will be empty).
- "`EQUAL`" — if both of the above choices are equal in terms of probability of being shot.

Example

headshot.in	headshot.out
0011	EQUAL
0111	ROTATE
000111	SHOOT

Problem B. Alien Communication Masterclass

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

Andrea is a famous science fiction writer, who runs masterclasses for her beloved readers. The most popular one is the Alien Communication Masterclass (ACM), where she teaches how to behave if you encounter alien life forms or at least alien artifacts.

One of the lectures concerns retrieving useful information based on aliens' writings. Andrea teaches that based on alien mathematical formulas, one could derive the base of the numeral system used by the aliens, which in turn might give some knowledge about aliens' organisms. (For example, we use numeral system with base 10, due to the fact that we have ten fingers on our upper extremities).

Suppose for simplicity that aliens use the same digits as we do, and they understand and denote addition, subtraction, multiplication, parentheses and equality the same way as we do.

For her lecture, Andrea wants an example of a mathematical equality that holds in numeral systems with bases $a_1, a_2, \cdots, a_n$, but doesn't hold in numeral systems with bases $b_1, b_2, \cdots, b_m$. Provide her with one such formula.

Input

The first line of the input file contains two integer numbers, n and m ($1 \le n, m \le 8$). The second line contains n numbers, $a_1, a_2, \cdots, a_n$. The third line contains m numbers, $b_1, b_2, \cdots, b_m$.

All a_i and b_i are distinct and lie between 2 and 10, inclusive.

Output

Output any syntactically correct mathematical equality that holds in numeral systems with bases $a_1, a_2, \cdots, a_n$, but doesn't hold in numeral systems with bases $b_1, b_2, \cdots, b_m$. The equality can contain only digits 0 through 9, addition ('`+`'), subtraction and unary negation ('`-`'), multiplication ('`*`'), parentheses ('`(`' and '`)`') and equality sign ('`=`'). There must be exactly one equality sign in the output.

Any whitespace characters in the output file will be ignored. The number of non-whitespace characters in the output file must not exceed 10 000.

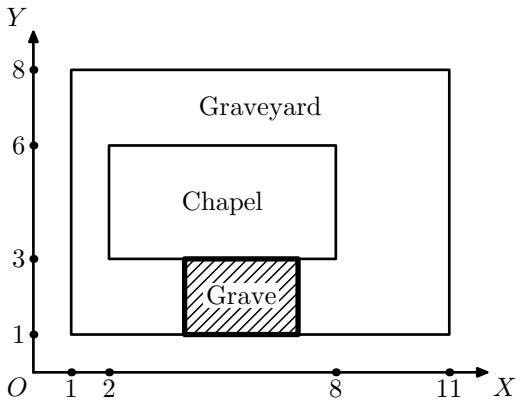
Examples

acm.in	acm.out
1 2 2 3 9	(10 - 1) * (10 - 1) + 1 = 10
2 2 9 10 2 3	2 + 2 = 4

Problem C. Grave

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

Gerard develops a Halloween computer game. The game is played on a rectangular graveyard with a rectangular chapel in it. During the game, the player places new rectangular graves on the graveyard. The grave should completely fit inside graveyard territory and should not overlap with the chapel. The grave may touch borders of the graveyard or the chapel.



Gerard asked you to write a program that determines whether it is possible to place a new grave of given size or there is no enough space for it.

Input

The first line of the input file contains two pairs of integers: x_1, y_1, x_2, y_2 ($-10^9 \le x_1 < x_2 \le 10^9$; $-10^9 \le y_1 < y_2 \le 10^9$) — coordinates of bottom left and top right corners of the graveyard. The second line also contains two pairs of integers x_3, y_3, x_4, y_4 ($x_1 < x_3 < x_4 < x_2$; $y_1 < y_3 < y_4 < y_2$) — coordinates of bottom left and top right corners of the chapel.

The third line contains two integers w, h — width and height of the new grave ($1 \le w, h \le 10^9$). Side with length w should be placed along OX axis, side with length h — along OY axis.

Output

The only line of the output file should contain single word: "`Yes`", if it is possible to place the new grave, or "`No`", if there is not enough space for it.

Examples

grave.in	grave.out
1 1 11 8 2 3 8 6 3 2	Yes
1 1 11 8 2 3 8 6 4 3	No

Problem D. Bureaucracy

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

Long ago, in a kingdom far, far away the king decided to keep a record of all laws of his kingdom. From that moment whenever a new law was passed, a corresponding record was added to the law archive.

Many centuries later lawyers discovered that there were only two types of laws in the kingdom:

- *direct law*, that states a new norm;
- *canceling law*, that cancels one of the previous laws.

The law is considered *active* if and only if there is no active law that cancels it.

You are to write program that finds out which laws are still active.

Input

The first line of the input file contains an integer number n ($1 \le n \le 100\,000$) — the number of passed laws.

The following n lines describe one law each. Each description has one of the following formats:

- "`declare`", meaning that a direct law was passed.
- "`cancel` i", where i is the number of law being cancelled by this one.

The laws are numbered from one.

Output

The first line of the output file must contain the number of active laws. Following lines must contain numbers of these laws listed in increasing order.

Example

bureau.in	bureau.out
5 declare cancel 1 declare cancel 2 cancel 3	3 1 4 5

Problem E. Ideal Contest

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

It's a pity that there is no higher-order contest for ACM ICPC regionals and subregionals; probably this is because it's very hard to rank them. But now we have an idea how to do this! The idea is based on a notion of *negidealness* — a number showing non-conformity of the contest results with "ideal" contest criteria. It is a weighted sum of the following specific negidealnesses (penalties).

Vainness penalty V. Each team should solve at least one problem. If a team solves no problems, a penalty of $1/T$ (where T is the number of teams that participated in the contest) for each such team is added.

Oversimplification penalty O. There should be no team with all the problems solved. If some teams solve all the problems, a penalty of $1/T$ is added for each such team.

Evenness penalty E. The number of problems solved by different teams should increase evenly. If the difference in problems solved between two adjacent (in the standings) teams is greater than 1, then the penalty of $1/P$ (where P is the total number of problems) is added for each skipped number of problems solved. E.g., if a team solves 5 problems, and the next team solves 1 problem, then the penalty of $3/P$ should be added, since no team solved 2, 3 or 4 problems.

Unsolvability penalty U. Every problem should be solved by at least one team. If a problem is not solved, a penalty of $1/P$ for each such problem is added.

Instability penalties $I_1, I_2, \ldots, I_P$. If a problem p was solved by a team, then this problem should be solved by all the teams ranked above. For each team which did not solve problem p ranked above the lowest-ranked team that did solve problem p a penalty of $1/T$ is added to I_p.

The *total negidealness* N equals $1.03V + 3.141O + 2.171E + 1.414U + (I_1 + I_2 + \cdots + I_P)/P$.

Write a program that finds the negidealness of the given results table.

Input

The input file contains a contest results table in plain ASCII. The only whitespace symbol in the table is a space. There is always at least one space separating columns. The problems are named with capital English letters in the alphabetical order. There are at most 26 problems and at most 300 teams.

Output

The output data should contain the penalties for each criterion (values V, O, E, U, $I_1, \ldots, I_P$) and the total negidealness. All the real numbers should be precise up to 3 digits after the decimal point.

Example

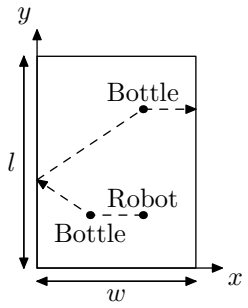
ideal.in	ideal.out
The contest header may contain arbitrary number of lines Team A B C D E = Time R --- Revda STU + + +2 +1 -9 4 9274 1 Girvas NU #1 + + -1 . -11 2 321 2 Kargopol SU + -3 + . -4 2 321 2 Utorgosh SU . . . + -5 1 122 4 Dubrovno SU . + -1 . -4 1 123 5 Girvas NU - 2 . . . -5 -99 0 0 6	Vainness = 0.167 Oversimplification = 0.000 Evenness = 0.200 Unsolvability = 0.200 Instability 1 = 0.000 Instability 2 = 0.333 Instability 3 = 0.000 Instability 4 = 0.333 Instability 5 = 0.000 Negidealness = 1.022

Problem F. Kitchen Robot

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

Robots are becoming more and more popular. They are used nowadays not only in manufacturing plants, but also at home. One programmer with his friends decided to create their own home robot. As you may know most programmers like to drink beer when they gather together for a party. After the party there are a lot of empty bottles left on the table. So, it was decided to program robot to collect empty bottles from the table.

The table is a rectangle with the length l and width w. Robot starts at the point (x_r, y_r) and n bottles are located at points (x_i, y_i) for $i = 1, 2, \ldots, n$. To collect a bottle robot must move to the point where the bottle is located, take it, and then bring to some point on the border of the table to dispose it. Robot can hold only one bottle at the moment and for simplicity of the control program it is allowed to release bottle only at the border of the table.



You can assume that sizes of robot and bottles are negligibly small (robot and bottles are points), so the robot holding a bottle is allowed to move through the point where another bottle is located.

One of the subroutines of the robot control program is the route planning. You are to write the program to determine the minimal length of robot route needed to collect all the bottles from the table.

Input

The first line of the input file contains two integer numbers w and l — the width and the length of the table ($2 \le w, l \le 1000$).

The second line of the input contains an integer number n — the number of bottles on the table ($1 \le n \le 18$). Each of the following n lines contains two integer numbers x_i and y_i — coordinates of the i-th bottle ($0 < x_i < w$; $0 < y_i < l$). No two bottles are located at the same point.

The last line of the input file contains two integer numbers x_r and y_r — coordinates of the robot's initial position ($0 < x_r < w$; $0 < y_r < l$). Robot is not located at the same point with a bottle.

Output

Output the length of the shortest route of the robot. Your answer should be accurate within an absolute error of 10^{-6}.

Example

kitchen.in	kitchen.out
3 4 2 1 1 2 3 2 1	5.60555127546399

Problem G. Asteroids

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

Association of Collision Management (ACM) is planning to perform the controlled collision of two asteroids. The asteroids will be slowly brought together and collided at negligible speed. ACM expects asteroids to get attached to each other and form a stable object.

Each asteroid has the form of a convex polyhedron. To increase the chances of success of the experiment ACM wants to bring asteroids together in such manner that their centers of mass are as close as possible. To achieve this, ACM operators can rotate the asteroids and move them independently before bringing them together.

Help ACM to find out what minimal distance between centers of mass can be achieved.

For the purpose of calculating center of mass both asteroids are considered to have constant density.

Input

Input file contains two descriptions of convex polyhedra.

The first line of each description contains integer number n — the number of vertices of the polyhedron ($4 \le n \le 60$). The following n lines contain three integer numbers x_i, y_i, z_i each — the coordinates of the polyhedron vertices ($-10^4 \le x_i, y_i, z_i \le 10^4$). It is guaranteed that the given points are vertices of a convex polyhedron, in particular no point belongs to the convex hull of other points. Each polyhedron is non-degenerate.

The two given polyhedra have no common points.

Output

Output one floating point number — the minimal distance between centers of mass of the asteroids that can be achieved. Your answer must be accurate up to 10^{-5}.

Example

asteroids.in	asteroids.out
8 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1 5 0 0 5 1 0 6 -1 0 6 0 1 6 0 -1 6	0.75

Problem H. Galaxy Interconnection

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

> And then, eight hundred and eight years ago, there occurred an event of such great importance that it marked a new era in the history of mankind — the Era of the Great Circle.
>
> Andromeda: A Space-Age Tale
> Ivan Yefremov

More than 1000 years passed from the first ACM ICPC. Several civilizations of our Galaxy, including the Earth, are united in the *Great Circle* whose members exchange and relay scientific and cultural information. Due to the distribution of gravity and dark energy flows in galaxy, there are bidirectional *communication channels* between some planets. Via these channels, civilizations can distribute knowledge across all the Great Circle.

It's now the time to distribute researches between civilizations: each planet has to choose one of k main research areas (such as *Repagular Calculus* or *Given String Theory*).

The choice of k was not accidental. The Great Circle started from *Initial Circle* — a set of k planets that were connected with channels forming a cycle. Later new channels were investigated and built and new planets were connected to the Great Circle. However due to high energy cost of communication channels, each planet of the Great Circle has strictly less than k channels.

Two planets connected with direct channel shouldn't choose the same research area: they'd better choose different ones and share research results.

There is one more limitation. Periodically civilizations send space expeditions in order to explore new planets and visit neighbors from the Great Circle. Expeditions are planned in such a way that spaceships can fly from one planet to another if these planets are connected with communication channel. This helps to prepare the expedition: to build refuel stations, optimize ship communication systems, etc.

Some expeditions, so-called *Research Audits*, are planned in such a way that a space ship starts from some planet, then makes $k - 1$ jumps, visiting total of k planets (including origin). These k planets together should provide all k research areas: the expedition will check the progress of researching.

Your task is to choose one research area for each planet in such a way that:

- there are no two planets connected by communication channel with the same research area;
- it is possible to send Research Audit expedition from every planet of the Great Circle.

Input

First line of the input file contains three integer numbers: n, k and m — the number of planets in the Great Circle, the number of research areas and the number of communication channels correspondingly ($3 \le n \le 5000$; $3 \le k \le \min(n, 10)$; $1 \le m \le 10000$).

The following m lines describe channels, one per line. Communication channel is described by two integer numbers — identifiers of the planets it connects. Planets are identified by integer numbers from 1 to n in such a way that planets from 1 to k form the Initial Cycle.

Output

Output exactly n integers in one line, i-th integer should identify a research area for planet i (an integer number from 1 to k).

Examples

galaxy.in	galaxy.out
5 4 5 1 2 2 3 3 4 4 1 5 3	1 2 3 4 2
6 4 9 1 2 2 3 3 4 4 1 5 2 5 3 6 5 6 4 6 1	1 2 3 4 4 2