Problem A. Arrange the Vertices

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a graph of N vertices and M edges. Cathy wants to arrange vertices on the circle of radius R and center placed in origin, so that edges correspond to segments connecting vertices, and herewith can cross only in vertices. Euclidean distance between any vertices must be not less than 1.

It is possible to arrange vertices in such a way?

Input

First line of the input contains three integers N, M and R ($1 \le N, R \le 1000$, $0 \le M \le 5000$). Next M lines contain the description of graph edges (one per line): two different integers a_i and b_i ($1 \le a_i, b_i \le N$). There are no multiedges in graph.

Output

In the first line print "`Yes`". if such an arrangement is poissible, otherwise print "`No`". If the answer is positive, then next N lines must contain coordinates of graph vertices x_i and y_i (while checking if a vertex lies on circumference, note that relative error should not be greater than 10^{-8}).

Examples

standard input	standard output
3 3 1 1 2 1 3 3 2	Yes 1.0000000000 0.0000000000 -0.5000000000 -0.8660254038 -0.5000000000 0.8660254038
4 6 100 1 2 1 3 1 4 2 3 2 4 3 4	No

Problem B. Bad String, Good String

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 megabytes

String s consists of lowercase English letters. Some positions of the string are *bad*. A substring of s is called *good* if it contains no more than one bad position and its left and right ends aren't bad positions. Your task is to count the number of distinct non-empty good substrings of s. Two substrings are considered equal if they are equal as strings.

Input

The first line contains a non-empty string s ($|s| \le 10^5$). The second line contains one integer n — the number of bad positions ($0 \le n \le |s|$). Finally, the third line contains n bad positions a_i in strictly ascending order ($1 \le a_i \le |s|$).

Output

Output one line containing the number of distinct non-empty good substrings of s.

Examples

standard input	standard output
abbababa 2 4 6	10

Note

The corresponding strings for the example are a, ab, abb, $abbab$, b, bb, $bbab$, bab, $baba$ and ba.

Problem C. Circle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Circle S_1 of radius r_1 is inscribed in angle A of triangle ABC (S_1 lies inside ABC). It doesn't have common points with segment BC. A tangent CD to circle S_1 (different from CA) is built from point C (where D lies on segment AB). Circle S_2 is inscribed in formed triangle CDB. A tangent AE to circle S_2 (different from AB) is built from point A (where E lies on segment BC). Circle S_3 is inscribed in formed triangle AEC, and so on. Calculate the radius of circle S_n.

Input

The first line contains four integers AB, AC, BC, r_1 ($1 \le AB, AC, BC, r_1 \le 10^5$) — triangle side lengths and radius of S_1. The second line contains a single integer n ($1 \le n \le 10^9$).

Output

You should output one number — the radius of S_n. The absolute error of your answer should not exceed 10^{-5}.

Examples

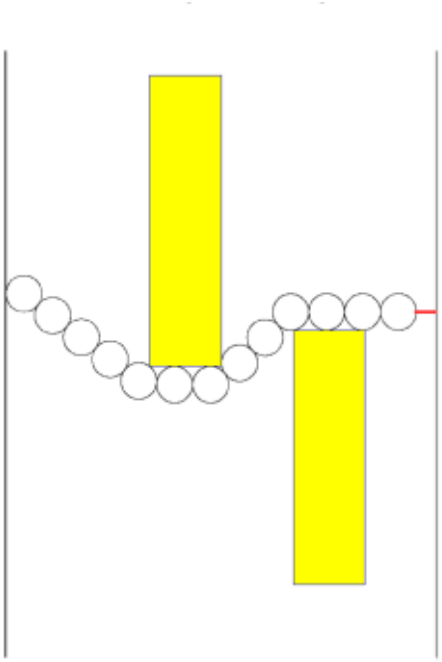
standard input	standard output
15 14 13 1 2	3.6521739130

Problem D. Decoration

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Suppose you want to decorate a party room wall with a string of rings, each having 1-foot diameter and each touching the next ring at precisely one point. You cannot necessarily string the rings in a straight line because there are rectangular pictures on the wall that you must avoid. From a given starting point against the left end of the wall you want to use as few rings as possible so that the last ring is less than one foot from the right end of the wall. Using the minimum number of rings, you also wish to minimize the distance between the final ring and the right wall.

A figure below shows such an optimal solution, with the minimum number of rings and with the distance to the right wall (shown as a red segment) minimal among those solutions with the fewest rings.



The pictures on the wall will have some characteristics upon which you may depend:

- Each picture is at least two feet wide.
- Each picture is at least two feet away from the left wall, the right wall, the floor, and the ceiling.
- For each picture, the floor-to-ceiling region that spans from two feet to the left of the picture to two feet to the right of the picture is unobstructed by other pictures.

Input

All distances are measured in feet. The first line of input contains three integers, N, S, and W, where $1 \le N \le 6$ is the number of pictures on the wall, $1 \le S \le 99$ is the starting ring's center height above the floor, and $1 \le W \le 99$ is the width between the left and right walls. The center of the starting disk will be 0.5 feet from the left wall.

The second line contains picture position information for the N pictures in order from the left side of the room to the right. Each picture is given by four integer distances in feet, L, R, B and T. Here L and R are the distances from the left end of the wall to the left and right edges of the picture, and B and T are

the distances from the floor to the bottom and top edges of the picture. For each picture $L + 2 \le R$ and $2 \le B < T$.

For the leftmost picture, $2 \le L$. For the rightmost picture, $R \le W - 2$. The R distance for one picture is always at least 2 less than the L distance for the next picture to the right.

Assume the ceiling of the room is at least two feet higher than the center of the initial ring and the tops of all the pictures, so it is out of the way, and its exact height is not relevant.

Output

Print a single floating-point number: $M + d$, where M is the minimal number of rings in a string ending with rightmost point less than 1 foot from the right wall, and d is the minimal distance from the right wall to the rightmost point on the M-th ring. The number that you output should be accurate to within an absolute or relative error of 10^3.

Note that the form of the output is such that there is no special roundoff issue if you have a ring's right point at a distance very close to 1 foot from the right wall: With a distance 1 or over, there is room for 1 more ring, and the remaining distance to the wall goes down by 1 to compensate.

Example

standard input	standard output
2 10 12 4 6 8 16 8 10 2 9	13.573069918537234

Problem E. Explosion

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

One day, Alice found a bag. Suddenly Alice heard tick-tack sound... a bomb! Fortunately, the bomb has not exploded yet, and there may be a little time remained to hide behind the buildings.

The appearance of the city can be viewed as an infinite plane and each building forms a polygon. Alice is considered as a point and the bomb blast may reach Alice if the line segment which connects Alice and the bomb does not intersect an interior position of any building. Assume that the speed of bomb blast is infinite; when the bomb explodes, its blast wave will reach anywhere immediately, unless the bomb blast is interrupted by some buildings.

Note that even if the line segment connecting Alice and the bomb touches a border of a building, bomb blast still can blow off Alice. Since Alice wants to escape early, she wants to minimize the length she runs in order to make herself hidden by the building.

Your task is to write a program which reads the positions of Alice, the bomb and the buildings and calculate the minimum distance required for Alice to run and hide behind the building.

Input

The input contains no more than 40 test cases. Each test case has the following format:

The first line of each test case contains an integer N ($1 \leq N \leq 100$), which denotes the number of buildings. In the second line, there are two integers b_x and b_y ($-10^4 \leq b_x, b_y \leq 10^4$), which means the location of the bomb. Then, N lines follows, indicating the information of the buildings.

The information of building is given as a polygon which consists of points. For each line, it has an integer m_i ($3 \leq m_i \leq 100$, sum of all m_i does not exceed 500) meaning the number of points the polygon has, and then m_i pairs of integers $x_{i,j}$, $y_{i,j}$ follow providing the x and y coordinate of the point.

You can assume that:

- the polygon does not have self-intersections;
- any two polygons do not have a common point;
- the set of points is given in the counter-clockwise order;
- the initial positions of Alice and bomb will not by located inside the polygon;
- there are no bomb on the any extended lines of the given polygon's segment;

Alice is initially located at (0, 0). It is possible that Alice is located at the boundary of a polygon.

$N = 0$ denotes the end of the input. You may not process this as a test case.

Output

For each test case, output one line which consists of the minimum distance required for Alice to run and hide behind the building. An absolute error or relative error in your answer must be less than 10^{-6}.

Example

standard input	standard output
1 1 1 4 -1 0 -2 -1 -1 -2 0 -1 1 0 3 4 1 1 1 2 -1 2 -1 1 1 -6 -6 6 1 -2 2 -2 2 3 -2 3 -2 1 1 1 1 -10 0 4 0 -5 1 -5 1 5 0 5 1 10 1 4 5 1 6 2 5 3 4 2 2 -47 -37 4 14 3 20 13 9 12 15 9 4 -38 -3 -34 -19 -34 -14 -24 -10 0	1.00000000 0.00000000 3.23606798 5.00000000 1.00000000 11.78517297

Problem F. Favorite Restaurants

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

In the Bytesdorf village there are N restaurants numbered with sequential integers between 1 and N. The owners of the restaurants have their favorite restaurants they like to visit: if we will ask the owner of some restaurant for a recommendation, he will, besides his restaurant, recommend:

- his favorite restaurants;
- all favorite restaurants of owners of those restaurants
- and all favorite restaurants of owners of restaurants from previous step
- ...and so on...

Alice plans to visit several restaurants in the following way:

- The first restaurant will be chosen arbitrarily.
- Each next restaurant will be chosen from list of restaurants, recommended by owner of current restaurant, which was not visited by Alice before.
- Alice may decide to stop her restaurant trip at any time

Each restaurant have two prices for the main menu: X_i and Y_i. When the guest enters the restaurant, the owner asks who recommended the restaurant. If this person is in the recommendation list of owner of this restaurant, then guest pays X_i, otherwise (including the case when it is starting restaurant and there was no recommendation) guest pays Y_i.

Let M is the maximum number of restaurants that Alice can visit in this way. For each integer K between 1 and M you need to calculate minimum amount of money Alice need to visit exactly K restaurants.

Input

First line of the input contains an integer N ($1 \le N \le 1000$) — the number of restaurants. Each of following N lines begins with two integers X_i, Y_i — prices for two types of guests ($1 \le X_i, Y_i \le 10^4$), then the integer F_i follows — number of favorite restaurants for owner of i-th restaurant, then F_i integers are listed — numbers of the restaurants from favorite list. It is guaranteed that all those numbers are pairwise distinct and not equal to i.

Output

Print K lines (where K is maximum number of restaurants Alice can visit in a way, described in problem statement), j-th of those lines must contain one integer — minimum amount of money Alice need to pay to visit exactly j restaurants.

Example

standard input	standard output
4 100 200 1 2 200 300 1 3 200 250 2 2 4 200 300 0	200 450 650 950
9 100 100 0 300 400 1 4 350 500 1 2 550 600 3 7 3 2 900 300 2 7 6 250 400 1 5 900 900 2 9 8 400 500 1 9 500 400 0	100 550 950 1450 2150 3050

Problem G. Groups and Teams

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

There are $3 \cdot N$ same year students studying in two specializations — mathematics (`M`) and programming (`P`). Let's call students of the first specialization mathematicians and students of the second specialization programmers. There are several groups of each specialization. The number of groups is the same for both specializations. The number of students in each group is greater than or equal to three.

The students are loaded much by the studies, that's why they can meet each other only during classes. They have separate classes for each group except physical education (P. E.). P. E. classes are held for pairs of groups of different specialization and each group share its P. E. classes with only one other group. All groups attend P. E. classes.

All the students are very responsible and attend all the classes. When two students meet each other at the same class, they become familiar.

There is a head in each group. All the group heads attend head meetings, where all of them meet each other and become familiar.

You should form N teams consisting of three students each so that each student is used in exactly one group. There should be at least one mathematician and at least one programmer in each team. All the members of one team should be familiar to each other.

Input

The first line contains 2 integers K and M ($1 \le K, M \le 1000$, $K = 3 \cdot N$) — the total number of students and the number of pairs of familiar students correspondingly. The second line contains K characters '`M`' or '`P`', i-th character describes i-th student's specialization. The following M lines define pairs of familiar students. It's guaranteed that the input describes student's specializations and relationships that do not contradict the problem statement.

Output

The first line should contain "`Yes`" or "`No`" depending on the possibility to form N teams. If such possibility exists, the following N lines should contain numbers of the students of each team, three numbers in each line. If there are several solutions, output any.

Examples

standard input	standard output
12 34 MMPPPMMMMPPP 1 2 1 3 1 4 1 5 1 6 1 7 1 10 2 3 2 4 2 5 2 6 3 4 3 5 3 6 4 5 4 6 4 7 4 10 5 6 7 8 7 9 7 10 7 11 7 12 8 9 8 10 8 11 8 12 9 10 9 11 9 12 10 11 10 12 11 12	Yes 4 5 1 6 2 3 10 12 7 9 8 11

Problem H. Hacker Neo

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 megabytes

The Matrix generates new password in next way: it started with old password (a string consisting of lowercase Latin letters) and does some operations on it. Each operation involves turning the substring of characters with indices between L and R (inclusive) into a palindrome, i. e. the left part of the substring is reflected to the right one (while the left part remains the same as before performing the operation).

Hacker Neo got the old password of Matrix and log of all operations. Help him to find the new password, i.e. the string after performing all the operations.

Input

The first line contains two integers N and M ($1 \le N \le 200\,000$, $1 \le M \le 50\,000$) — the length of the string and the number of operations. The second line contains the initial string. Each of the following M lines contains two integers L and R ($1 \le L \le R \le N$) describing the left and the right boundaries of the substring to perform the operation on.

Output

Output the string after performing all the operations.

Examples

standard input	standard output
10 1 abccdeacbf 3 9	abccdedccf
10 3 abcdefghij 1 2 1 3 3 9	aaadefedaj

Problem I. Interesting Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Igor and Ilona have played the game Risk a thousand times, so now they are trying to invent a new game called Aggressor playing on the same board in the form of a geographic map. The board contains N countries enumerated by sequential integers from 1 to N, and it is known which pairs of countries are neighbors. Note that some countries may be neighbors even if they do not physically share the boundary.

Before the start of the game, a number of knights have been set up in each country, and some countries have declared as "Aggressors", while the remaining "peaceful".

Game processed in turns. Player who cannot make a turn loses the game. Ilona is first to move. At his turn player choose one of two types of actions:

1. Attack:
   - The player selects the country aggressive country A with T_A knights and neighborung peaceful country M with T_M knights.
   - Action can be done only when $T_M > 0$.
   - Then each knight from the country A destroys one knight from the country M.
   - At the end of the turn in country M will remain $T_M - T_A$ knights (or 0 in case when $T_A > T_M$).
2. Help:
   - The player selects two adjacent peaceful countries M and O with T_M and T_O knights, respectively.
   - Action can be done only when $T_M > 0$.
   - If the T_M is odd, the player adds a new knight to country M.
   - Then exactly half of the knights from country M is moved to country O.

Note that countries do not belong to individual players; each player in their turn can choose either which the two adjacent countries, provided that the move is allowed.

Since the board contains N countries, there is 2^N ways to set the countries to aggressive and peaceful. For every possible distribution, Ilona and Igor will play one game. They are interested in how number of games won by Ilona and number of games won by Igor provided both players play optimally. In some cases no player can secure a win (for example if there are no aggressive countries).

Input

First line of the input contains one integer N ($2 \le N \le 40$) — number of countries. Next line contains N integers T_i — number of knights in i-th country ($1 \le T_i \le 4 \cdot 10^4$).

Next line contains one integer M ($1 \le M \le 780$) — number of pairs of neighbors. Each of next M lines contain two distinct integers between 1 and N — numbers of countries who are a neighbors. Each pair of countries is listed at most once.

Output

In first line of the output print number of games won by Ilona, in second — number of games won by Igor.

Examples

standard input	standard output
2 100 100 1 1 2	2 1
5 7 5 3 4 5 5 1 2 2 3 3 1 3 4 3 5	5 8

Problem J. Jatlin

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

In the Jatlin programming language, as in most existing programming languages, is possible to create more than one function in a program.

Unfortunately, in community version of Jatlin each function should have been defined before it can called by any other function (but recursive calls is allowed in Jatlin).

There is a certain program that need to be compiled. The programmers wrote that program using the commercial version, where function prototyping exists, so some functions are not defined before other function that call them.

So, they need your help to write a specific program that can rearrange functions in their code so it can be compiled. But programming environment of community version of Jatlin is not so user-friendly, each rearrangement take some time, so the total time for rearrangement should be minimized!

The time of moving a certain function is calculated by multiplying the number of lines in that function to the total number of skipped lines. For example, let there are four functions called A, B, C and D and defined in that order (i.e A at the top and D at the bottom) which have 10, 4, 6 and 3 lines respectively. Moving A below D will take $A \cdot (B + C + D) = 10 \cdot (4 + 6 + 3) = 130$. Moving D above B (between A and B) will take $D \cdot (C + B) = 3 \cdot (6 + 4) = 30$.

Input

Input begins with an integer N ($1 \le N \le 18$), the number of functions.

The next line contains N integers M_i ($1 \le M_i \le 100$) representing the number of lines in i-th function ($i = 1 \dots N$). Each of the next N lines each contains i-th function dependency list and will start with an integer C ($0 \le C < N$), the number of functions called from i-th function following by C integers F_j ($1 \le Fj \le N$) — numbers of called functions. The last line if the input contains the permutation of first N positive integers, representing the initial order of functions in the program

Output

For each case, print in a single line the minimum time needed to rearrange functions so program can be compiled by community version, or print "`-1`" if there's no such arrangement.

Example

standard input	standard output
4 7 3 12 8 1 3 1 4 2 1 3 0 1 2 3 4	-1
5 7 3 12 8 4 3 1 2 4 0 1 2 0 1 4 1 2 3 4 5	161

Problem K. Kings and Rooks

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

On the chessboard $N \times M$ K kings are placed. Find number of ways to place 3 rooks, so no two rooks attack each other. Note that two pieces cannot be placed in the same cell and that two rooks attack each other if they are on same row or column and there are no kings between them.

Input

First line of the input contains three integers N, M and K ($1 \le M, N \le 10^9$, $1 \le K \le 10^5$). Then K pairs of integers x_i and y_i follow ($0 \le x_i \le N - 1$, $0 \le y_i \le M - 1$) — coordinates of the kings.

Output

Print one integer — number of possible ways to place rooks modulo $10^9 + 7$.

Examples

standard input	standard output
3 3 1 0 0	4
5 8 2 2 2 4 5	3424

Problem L. Logical Expression

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Given the expression $(X_{1,1} \vee X_{1,2}) \wedge (X_{2,1} \vee X_{2,2}) \wedge \ldots \wedge (X_{M,1} \vee X_{M,2})$, where $X_{i,j} \in \{x_1, x_2, ..., x_N, \neg x_1, \neg x_2, \ldots, \neg x_N\}$ such as any x_i cannot appear in the expression more than twice.

Calculate number of ways to assign boolean values to x_i to make given logical expression true.

Input

First line of the input contains two integers N and M ($1 \le N \le 1000$, $N/2 \le M \le N$. Then M pairs of integers $Y_{i,j}$ follow ($1 \le |Y_{i,j}| \le N$). If $Y_{i,j} > 0$ then $X_{i,j} = x_{Y_{i,j}}$, if $Y_{i,j} < 0$ then $X_{i,j} = \neg x_{Y_{i,j}}$.

It is guaranteed that each variable cannot appear more than twice, i.e. there are only one of two pairs (i, j) such as $X_{i,j} = x_k$ or $X_{i,j} = \neg x_k$ for any k ($1 \le k \le N$).

Output

Print the number of ways to assign boolean values to x_i when given expression becomes true. Since answer may be too big, print it modulo $10^9 + 7$.

Example

standard input	standard output
2 1 1 -2	3
3 2 -1 -2 1 -3	4

Problem M. Matrix

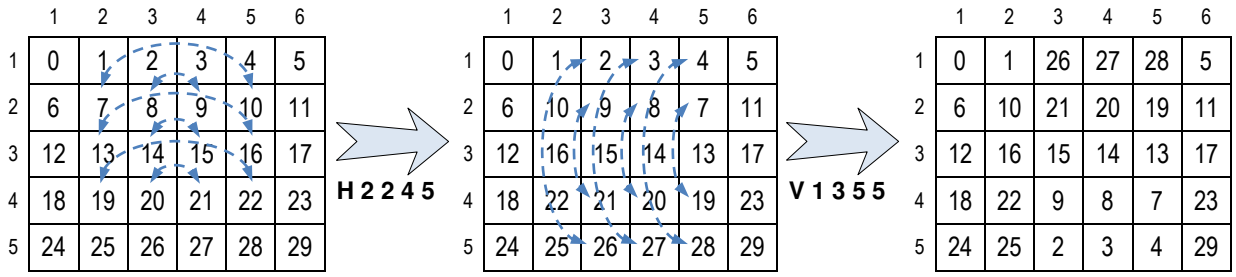
Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

You are given a matrix A containing N rows numbered from 1 to N and M columns numbered from 1 to M. Initially the element in row i, column j is $A_{i,j} = (i-1) \cdot M + j - 1$.

Someone has done several transformations to the matrix. Each transformation was either a horizontal reverse or a vertical reverse. Each reverse is described by four integers X_1, Y_1, X_2 and Y_2 such that $1 \le X_1 \le X_2 \le N$ and $1 \le Y_1 \le Y_2 \le M$.

After the horizontal reverse the element in row i, column j becomes equal to $A'_{i,j} = A_{i,Y_2-j+Y_1}$ iff $X_1 \le i \le X_2$ and $Y_1 \le j \le Y_2$, otherwise it remains the same ($A'_{i,j} = A_{i,j}$). After the vertical reverse the element in row i, column j becomes equal to $A'_{i,j} = A_{X_2-i+X_1,j}$ iff $X_1 \le i \le X_2$ and $Y_1 \le j \le Y_2$, otherwise it remains the same ($A'_{i,j} = A_{i,j}$).

Below you can see an example of applying a horizontal reverse and then a vertical reverse.



You should write a program that, given N, M and a sequence of operations, will be able to find the resulting matrix.

Input

The first line contains two space-separated integers N and M ($1 \le N, M \le 2000$). The next line contains an integer K ($1 \le K \le 2000$) — the number of operations. Each of the next K lines describes one operation. The first character of the line stands for the type of the operation ('`H`' for horizontal reverse or '`V`' for vertical reverse) after which there are four integers X_1, Y_1, X_2 and Y_2 — description of the selected rectangular range. Operations are given in order of their applying.

Output

The first line should contain N space-separated integers corresponding to the sum of the elements in the first, second, ..., N-th row of the matrix after applying all the operations. Similarly, the second line should contain M space-separated integers — the sum of the elements in the first, second, ..., M-th column of the resulting matrix.

Examples

standard input	standard output
5 6 2 H 2 2 4 5 V 1 3 5 5	87 87 87 87 87 60 74 73 72 71 85
5 4 6 H 1 1 1 1 V 4 4 5 4 H 1 2 4 4 V 1 2 5 3 H 3 1 5 4 H 2 3 3 4	36 42 38 38 36 41 54 41 54