Problem A. Ancient Klingon Language

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Research on the civilization of old kingdom of Klingon has been carried out by the archaeologists. Their recent excavation found two expressions engraved on a stone wall at the ruins of Klingon.

The expressions are composed of n distinct Klingon letters $a_1, \dots, a_n$, symbols corresponding to inequality signs ('<' and '>') and parentheses ('(' and ')'). In what follows, they are represented by capital English letters and ordinary symbols for description ease. Expressions conform to the following grammar rules with the start symbol E.

```
E ::= F | '(' E '<' E ')' | '(' E '>' E ')'
F ::= a  | a  | ... |a
       1    2          n
```

Analyses unifying multifarious knowledge on the old kingdom and its civilization revealed the following evaluation rules.

A total order relation is defined on the set of letters $a_1, \dots, a_n$.

When an expression consists only of a single letter, the value of the expression is that very letter.

For two expressions P and Q with values a_i and a_j, respectively, the value of the expression $(P < Q)$ is one of the letters a_i or a_j that precedes in the total order. If the two letters are the same, the value is that letter.

Similarly, for two expressions P and Q with values a_i and a_j, respectively, the value of the expression $(P > Q)$ is one of the letters a_i or a_j that comes later in the total order. If the two letters are the same, the value is that letter.

It was also found that the values of the two expressions discovered must be equal. How many different total orders, among $n!$ possible total orders, make the values of the two expressions equal?

Input

Each test case consists of four lines. The first line contains n ($1 \leq n \leq 16$), the number of different capital English letters corresponding to Klingon characters. The second line contains n distinct capital English letters without any spaces. The third and the fourth lines contain two discovered expressions S and T, respectively. Both S and T conform to the grammar given above and neither of them has more than 100 characters.

The end of the input is indicated by a line containing a zero. The number of datasets does not exceed 50.

Output

For each dataset, output a single line containing the number of different total orders making the values of the two expressions equal.

Sample Input

Example

standard input	standard output
3 CIP ((I<C)>(P<C)) (P>C) 2 AB (A<B) (A>B) 3 ANY ((A<N)<Y) ((N<Y)<((A<A)<N)) 6 ANSWER A ((E>(A>((E<W)>(N<S))))>(A<W)) 0	1 0 6 300

Problem B. Build The Evacuation Site

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

The council of your home city decided to build a public evacuation site to prepare for natural disasters. The evacuation site is desired to be built at a location reachable from as many locations across the city as possible, even when many road segments are closed due to a natural disaster. Given the road network of the city, please find the best candidate locations.

The road network is specified as an undirected graph. The nodes of the graph correspond to various locations in the city area, and the edges are two-way road segments connecting the locations. Each road segment is assigned a positive integer representing the assumed strength against disasters. A road segment with assumed strength s is expected to be available on a disaster of level below s, but will be closed on a disaster of level above or equal to s.

Adequacy of a location as an evacuation site is assessed by the following ordering. Let us define $r_s(v)$ as the number of locations that is directly or indirectly connected to the location v via road segments available even on the level s disaster. We define the location u is more suitable than the location v as an evacuation site if and only if there exists some $k \geq 1$ satisfying

- $r_1(u) = r_1(v)$,
- $r_2(u) = r_2(v)$, ...
- $r_{k-1}(u) = r_{k-1}(v)$, and
- $r_k(u) > r_k(v)$.

Please find the best locations based on the above criterion. If two or more locations are rated equally as the best, enumerate all of them.

Input

The input consists of multiple test cases, each in the following format.

First line of the test case contains positive integer n less than or equal to 10^5, representing the number of locations and positive integer m less than or equal to 10^5, representing the number of road segments. The locations in the city are given ID numbers 1 through n.

Each of m next lines describe segments and contain three integers a_i, b_i and s_i. The i-th segment connects the locations a_i and b_i, and its assumed strength is s_i. Each of a_i, b_i, and s_i is an integer and they satisfy $1 \leq a_i < b_i \leq n$ and $1 \leq si \leq 10^5$. When $i \neq j$, either $a_i \neq a_j$ or $b_i \neq b_j$ holds. The given graph is connected. That is, there is at least one path between every pair of locations.

The end of the input is indicated by a line containing two zeros. The number of datasets does not exceed 50.

Output

For each dataset, output the list of the ID numbers of all of the locations most suitable with the criterion described above separated by a space character, in the increasing order, in a line.

Example

standard input	standard output

3 3
1 2 1
1 3 2
2 3 3
3 3
1 2 3
1 3 3
2 3 3
5 5
1 2 5
2 3 1
3 4 2
3 5 1
4 5 2
0 0
	2 3
	1 2 3
	3 4 5

Problem C. Covid-19 Tracing

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

The SARS-nCOV-2 is known to transmit on close contact with carriers, whether or not with any symptoms. Therefore, it is effective for epidemic prevention to identify and test those who had close contact with persons confirmed or were judged highly probable to be infected. To realize this, a system is desired that perpetually records all the close contact events by an application on mobile phones and, when infection is confirmed, identifies all persons with direct or indirect infection risks based on the record.

You were asked to develop such a system, and have already finished the mobile phone part. When the installed application detects a close contact event with another person carrying a phone with the same application installed, it sends IDs of both to the surveillance center.

Your next task is to develop a program that, when infection of a user is confirmed, identifies users with risks of direct or indirect transmission from the infected user.

When a user of the system is confirmed to be infected, those users who made close contacts with the infected user within a certain time period (period of communicability) are suspected of infection. If a suspected user had close contact with still another user after that possible infection event, that user also is suspected of infection. The suspects are propagated repetitively in this manner.

When a user is confirmed infected, the ID of the user and the list of all the close contact events of all users with all users happened after the time when the confirmed user possibly becomes a carrier are given. All the events in the given list should be assumed to be within the period of communicability of the confirmed user. The output should be the number of users to whom the virus was possibly transmitted directly or indirectly from the infected user.

Input

The input consists of multiple test cases, each in the following format.

Each of the test cases starts with a line containing three integers: m ($1 \le m \le 100$) is the number of users, n ($0 \le n \le 1000$) is the number of events in the list, and p ($1 \le p \le m$) is the ID of the user confirmed to be infected.

The following n lines contain the close contact events between users, one event per line, in time order. Each line indicates that the users whose IDs are a_i and b_i had close contact. Here, a_i and b_i satisfy $1 \le a_i \le m$, $1 \le b_i \le m$, and $a_i \ne b_i$.

The end of the input is indicated by a line containing three zeros. The number of datasets does not exceed 50.

Output

For each test case, output a single line containing the total number of users including the user confirmed to be infected and users to whom the virus was possibly transmitted directly or indirectly from the confirmed user.

Example

standard input	standard output

3 2 1
1 2
2 3
4 5 2
1 3
3 4
2 1
3 1
1 2
100 0 100
0 0 0
	3
	3
	1

Problem D. Delivery Fare

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Ciel is estimating the delivery fare for a number of luggage pieces. The fare for a piece is determined by its width w, depth d and height h, each rounded up to an integer. The fare is proportional to the sum of them.

The list of the luggage pieces is at hand, but the list has the product $w \times d \times h$ for each piece, instead of the sum $w + d + h$ by mistake. This information is not enough to calculate the exact delivery fare.

Ciel therefore decided to estimate the minimum possible delivery fare. To this end, given the listed product p for each of the luggage pieces, the minimum possible sum $s = w+d+h$ satisfying $p = w \times d \times h$ should be found.

You are requested to help Ciel by writing a program that, when given the product p, computes the minimum sum s.

Input

The input consists of multiple datasets. Each of the datasets has one line containing an integer p ($0 < p < 10^{15}$).

The end of the input is indicated by a line containing a zero.

The number of datasets does not exceed 300.

Output

For each dataset, output a single line containing an integer s. s should be the minimum possible sum $w + d + h$ of three positive integers, w, d, and h, satisfying $p = w \times d \times h$.

Examples

standard input	standard output
1	3
2	4
6	6
8	6
729	27
47045881	1083
12137858827249	6967887
562949953421312	262144
986387345919360	298633
999730024299271	299973
999998765536093	999998765536095
0	

Problem E. Edge Path

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

A simple directed graph of N vertices and M edges is given.

The vertices of this graph are numbered from 1 to N, and the edges are numbered from 1 to M.

The edge i ($1 \le i \le M$) is going from vertex u_i to vertex v_i.

Determine if there is an edge path that meets the following condition: by reversing the orientation of all edges in the path, the entire graph becomes strongly connected.

Here, an edge path of length K ($K \ge 0$) is a empty sequence of edges (then $K = 0$) or sequence of edge numbers (e_1, e_2, …, e_K) that satisfy the following conditions: for $1 \le i \le K - 1$, $v_{e_i} = u_{e_{i+1}}$ and for any $1 \le i < j \le K$, $e_i \ne e_j$.

Input

First line of the input contains two integers N and M — number of edges and vertices in the graph ($2 \le N \le 10^5$, $1 \le M \le \min(10^5, N(N-1)$.

i-th of following M lines contains descriptions of edges. Each edge is given by two integers u_i and v_i — starting and ending vertices of the edge ($1 \le u_i, v_i \le N$, $u_i \ne v_i$, each pair (u_i, v_i) can meet in the list at most once.

Output

If there is no edge path that satisfies the condition, output -1.

Otherwise output the minimum path length that meets the condition

Examples

standard input	standard output

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

Problem F. Far Expedition

Input file:	*standard input*
Output file:	*standard output*
Time limit:	30 seconds
Memory limit:	512 mebibytes

Ten years after JAG (Japan Aerospace-exploration Group)'s asteroid explorer group Komiya took off from the earth in 2019, the day is approaching the target asteroid belt ICPC-2020.

In this exploration project, multiple probes will take samples from different asteroids. However, three days ago, a serious oversight by JAG's space observation department was noticed due to wrong translation to English.

According to their report, there are several obstacles between the current location of the group and ICPC-2020, and the probes may collide with them if they will keep the same traectory.

An emergency meeting was held immediately, and it was decided to control the probe by engine injection and avoid collisions. It has also been decided that you will be in charge of implementing the necessary control programs. The specifications of the program you are about to write will be as follows in the style of competitive programming.

The position of the probes and the asteroids targeted by each of the Q probes are given as points on the xy-plane (in this model we consider that probes, asteroids and obstacles are moving only on this plane). Obstacles are given as several non-intersecting simple polygons, each two of them does not have common points.

The probes can move along a po;ygonal path (sequence of segments) on this plane, and the goal for each probe is to reach the target asteroid without hitting obstacles (but they can touch them). Weak gravity acts on this space, and if the y coordinate does not increase when moving along the line segment, it is not necessary to inject, but if the y coordinate increases, it consumes 1 joule per increase of y coordinate by 1. If the probe's energy is used too much, it will be difficult to return, so we want to consume as little energy as possible.

For each probe find the minimum energy required to reach the target asteroid.

Input

The input data consists of no more than 50 test cases.

First line of each test case contains two integers — number of polygonal obstacles N ($1 \le N \le 10$) and number of probes Q ($1 \le Q \le 1000$) respectively.

Then N description of a polygons follow.

Description of a polygon starts with one integer v_i — number of vertices ($3 \le v_i \le 100$). Each of following v_i lines contains two integers x and y — coordinates of the next vertex of the i-th polygon. The polygon can be formed by connecting the vertices in order they are given in the input and connecting last and first vertices. You may assume that given polygon is simple and does not have self-intersections.

After that the probes are defined. Each probe is defined by its coordinates (s_{x_i}, s_{y_i}) and coordinates of the target asteroid (g_{x_i}, g_{y_i}).

All coordinates are integers does not exceeding -1000 by absolute value; you may assume that intersection of each two polygons (including vertices, edges and interior) is empty, i.e. polygons does not intersect nor overlap nor touch and that probes and asteroids are strictly outside the polygons.

The input ends with two space-separated zeroes.

Output

Print Q lines for each test case. i-th of those lines shall contain the minimum energy required for the i-th probe to reach the goal, in order they are listed in the input. The result should not contain more than 10^{-6} absolute error.

Examples

standard input	standard output

1 1
4
1 2
3 2
3 -1
1 -1
0 0 5 0
2 1
4
0 -1
-1 -4
0 -2
1 -4
4
0 1
-1 4
0 2
1 4
0 -3 0 3
4 3
6
0 2
3 2
2 0
3 -2
0 -2
1 0
6
3 0
4 -2
5 -2
4 0
5 2
4 2
7
6 2
7 2
9 1
7 0
7 -2
6 -2
5 0
6
9 0
10 -2
11 -2
10 0
11 2
10 2
0 -1 11 1
11 1 0 -1
11 1 11 -1
0 0
	1.000000000000
	8.000000000000
	3.000000000000
	1.000000000000
	0

Problem G. Graph Game

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

In 2020, the world is in the age of large graphs. The world was divided into two groups, one that claimed that "graphs were born first from the sides" and the other that claimed that "graphs were born first from the vertices", and the world was extremely chaotic. The ultimate form of the battle between the two factions who collide with each other is a two-player game using a graph. Today, Edge Man and Vertexko, who are the top sect, will compete in this game from now on.

In this game, a directed graph of N vertices and M sides is given first. This game consists of three phases, which progress in the order of the first phase, the second phase, and the evaluation phase.

First move phase: The first move is the Edge Man, and in this phase, the Edge Man sets the probability to randomly select exactly one from the M edges. That is, the probability e_i that the i-th edge is selected is assigned to all edges as desired. Here, each e_i is a real number greater than or equal to 0 and less than or equal to 1, and the sum of all e_i must be exactly 1.

Second Phase: Then Vertexko sets the probability to randomly select exactly one vertex from N vertices. That is, the probability v_j that the j-th vertex is selected is assigned to all vertices as desired. Like edges, each v_j is a real number greater than or equal to 0 and less than or equal to 1, and the sum of all v_j's must be exactly 1. Here, the second move can allocate the probability after freely looking at the probability assigned to the side by the first move.

Evaluation Phase: Based on the assigned probabilities, one edge and one vertex are randomly determined independently. The score of the game is determined as follows according to the relationship between the selected sides and vertices.

- If the vertex is the starting point of the directed edge, the score is -1.
- If the vertex is the end point of the directed edge, the score is 1.
- If the vertices are neither the start nor the end of the directed edge, the score is 0.

In this game Vertexko, allocates probabilities to minimize the expected score. On the other hand, the first player, Edge Man, allocates the probabilities to maximize the expected value of the score, based on the fact that the second player, Vertexko, takes a strategy to minimize the expected value of the score. Find the expected value of the score when two players assign probabilities to edges and vertices based on the above strategy in a given graph.

Input

The input consists of 50 or less test cases.

The first line of the test case consists of the number of vertices N ($2 \le N \le 10^4$) and the number of edges M ($1 \le M \le 10^4$) of the directed graph used in the game.

The following M lines represents the information of the directed edge of the graph. Each line contains two integers — starting vertex of the edge a_i and endpoint b_i ($1 \le a_i, b_i \le N$). Given graph have no self-loops and no multiple edges.

The end of the input is represented by a line of two zeros.

Output

In a given graph, output the expected value of the game score on one line when both take the optimal strategy for each of the above with absolute error 10^{-10} or better.

Example

standard input	standard output
4 4	-0.25
1 2	-0.333333333333333
1 3	0
2 4	-0.333333333333333
3 4	-0.5
3 3	
1 2	
1 3	
2 3	
4 4	
1 2	
2 3	
3 4	
4 1	
4 3	
1 2	
1 4	
2 3	
5 2	
1 2	
5 4	
0 0	

Problem H. Highways

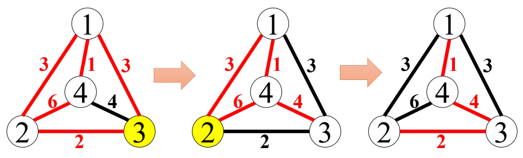
Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

The map of the kingdom can be represented as undirected complete graph. Every pair of the citied (nodes in the graph) is connected by an edge, colored either red (representing highway) or black (representing plain road). Each edge is associated with an integer value called penalty.

By repeating certain operations on the given graph, a *spanning tree* should be formed with only the red edges. That is, the number of red edges should be made exactly one less than the number of nodes, and all the nodes should be made connected only via red edges, directly or indirectly.

If two or more such trees can be formed, one with the least sum of penalties of red edges should be chosen.

In a single operation step, you choose one of the nodes and flip the colors of all the edges connected to it: Red ones will turn to black, and black ones to red.



For example, the leftmost graph of Fig. F-1 illustrates the first dataset of Sample Input. By flipping the colors of all the edges connected to the node 3, and then flipping all the edges connected to the node 2, you can form a spanning tree made of red edges as shown in the rightmost graph of the figure.

Input

First line contains one integer n ($2 \le n \le 300$), the number of cities. The cities are numbered from 1 to n. Then $n-1$ lines follow, i-th of them containing $n-i$ integers $e_{i,i+1} \ldots e_{i,n}$).

The integer $e_{i,k}$ ($1 \le |e_{i,k}| \le 10^5$) denotes the penalty and the initial color of the edge between the node i and the node k. Its absolute value $|e_{i,k}|$ represents the penalty of the edge. $e_{i,k} > 0$ means that the edge is initially red, and $e_{i,k} < 0$ means it is black.

The end of the input is indicated by a line containing a zero. The number of datasets does not exceed 50.

Output

For each dataset, print the sum of edge penalties of the red spanning tree with the least sum of edge penalties obtained by the above-described operations. If a red spanning tree can never be made by such operations, print -1.

Example

standard input

```
4
3 3 1
2 6
-4
3
1 -10
100
5
-2 -2 -2 -2
-1 -1 -1
-1 -1
1
4
-4 7 6
2 3
-1
0
```

standard output

```
7
11
-1
9
```

Problem I. Ideal Container

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	512 mebibytes

The Ideally Compact Packaging Company (ICPC) corporation is a package manufacturer specialized in cylindrical containers, designing and manufacturing containers satisfying clients' demands.

The corporation has taken an order of a container to pack two different products together. Both products have shapes of convex polygonal prisms with the same height. The inside of the container should be precisely a cylinder whose height is equal to the height of the products.

Products must be placed vertically in the container, that is, the bottoms of the products must meet the inner bottom of the container. Products can be placed at an arbitrary position and with an arbitrary direction in the container, but they should not be laid upside-down nor stacked one on the other.

Products and the inner surface of the container may touch one another.

The customer requests to make the container as small as possible. Your task is to find the smallest possible diameter of the container bottom that can contain the two products.

Input

The input consists of at most 50 test cases, each in the following format.

First line of the test case contains one integer n ($3 \le n \le 40$) — the number of vertices of the bottom polygon of one of the products. The following n lines have two integers each, which are the x- and y-coordinates of the vertices of the bottom polygon. They are between -1000 and 1000, inclusive. The vertices are listed in a counter-clockwise order.

The bottom polygon is guaranteed to be convex.

Then comes the description of the bottom polygon of the other product in exactly the same manner.

The end of the input is indicated by a line containing a zero.

Output

For each dataset, output the smallest possible diameter of the bottom circle of the container that accommodates the two products together. The output must not contain an absolute or relative error greater than 10^{-6}.

Example

standard input	standard output
3	10.625000000000
8 0	2.828427124746
7 7	5.656854249492
0 6	26.000000000000
4	
0 0	
5 0	
5 5	
0 5	
3	
0 0	
2 2	
0 2	
3	
3 1	
5 1	
5 3	
5	
0 0	
3 0	
3 1	
1 3	
0 3	
5	
0 0	
3 0	
3 1	
1 3	
0 3	
9	
0 0	
9 4	
13 13	
9 22	
6 24	
-6 24	
-9 22	
-13 13	
-9 4	
4	
1 0	
0 5	
-1 0	
0 -5	
0	

Problem J. Jigsaw Puzzle

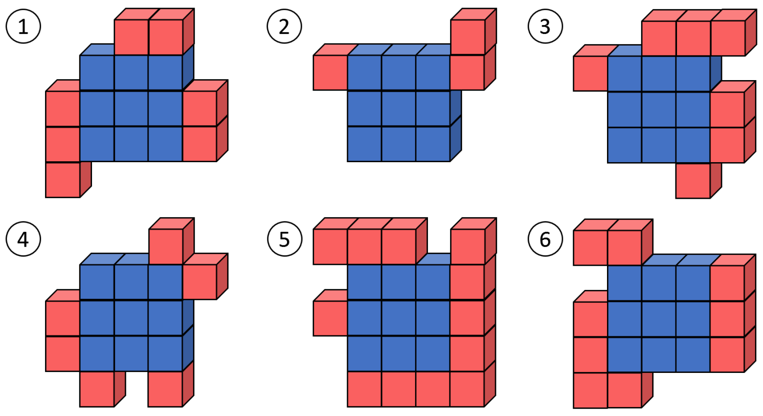
Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A 3D jigsaw puzzle is to construct a hollow cube with filled surface. Pieces of a puzzle is made of a number of small unit cubes grid-aligned on a plane. For a puzzle constructing a cube of its side length n, unit cubes are on either of the following two areas.

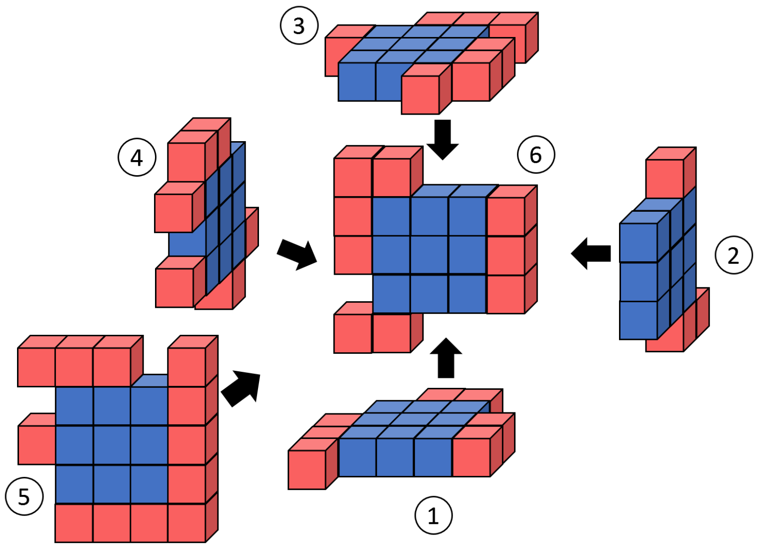
- Core: A square area with its side length $n - 2$. Unit cubes fill up this area.
- Border: The area of width 1 unit forming the outer fringe of the core. Each unit square in this area may be empty or with a unit cube on it.

Each piece is connected with faces of its unit cubes. Pieces can be arbitrarily rotated and either side of the pieces can be inside or outside of the constructed cube. The unit cubes on the core area should come in the centers of the faces of the constructed cube.

Consider that we have six pieces (corresponging to the first dataset of Sample Input):



Then, we can construct a cube as shown below:



John has collected a number of such puzzles. One day, his pet vugluscr mixed together his collection, so John cannot find yet from which six pieces he can construct a cube. Help John to check that.

Input

The input consists of at most 200 testcases, each in the following format.

The first line of the test case contains an integer n denoting the length of one side of the cube to be constructed ($3 \leq n \leq 9$, n is odd). The following $6n$ lines give the six pieces. Each piece is described in n lines. Each of the lines corresponds to one grid row contains the string of length n. Each of the characters in the string, either '`X`' or '`.`', indicates whether or not a unit cube is on the corresponding unit square: '`X`' means a unit cube is on the column and '`.`' means none is there.

The core area of each piece is centered in the data for the piece.

The end of the input is indicated by a line containing a zero.

Output

For each dataset, output "`Yes`" if we can construct a cube, or "`No`" if we cannot.

Examples

standard input	standard output

`5`
`..XX.`
`.XXX.`
`XXXXX`
`XXXXX`
`X....`
`....X`
`XXXXX`
`.XXX.`
`.XXX.`
`.....`
`..XXX`
`XXXX.`
`.XXXX`
`.XXXX`
`...X.`
`...X.`
`.XXXX`
`XXXX.`
`XXXX.`
`.X.X.`
`XXX.X`
`.XXXX`
`XXXXX`
`.XXXX`
`.XXXX`
`XX...`
`.XXXX`
`XXXXX`
`XXXXX`
`XX...`
`0`
	`Yes`

standard input	standard output
5	No
..XX.	
.XXX.	
XXXXX	
XXXX.	
X....	
....X	
XXXXX	
.XXX.	
.XXX.	
.....	
.XXXX	
XXXX.	
.XXXX	
.XXXX	
...X.	
...X.	
.XXXX	
XXXX.	
XXXX.	
.X.X.	
XXX.X	
.XXXX	
XXXXX	
.XXXX	
.XXXX	
XX...	
XXXXX	
XXXXX	
.XXXX	
XX...	
0	

Problem K. Knight Move

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	512 mebibytes

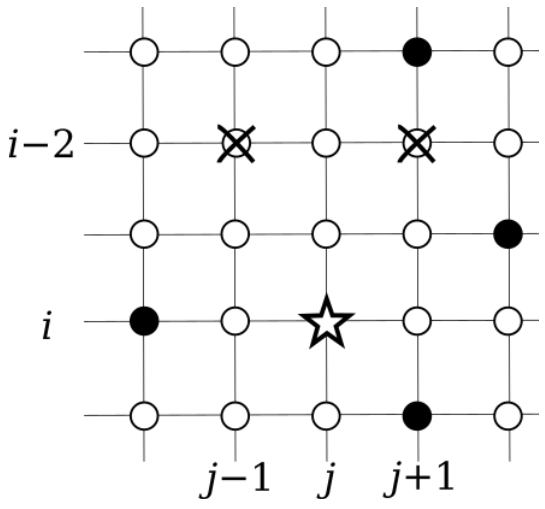
You are given a set of points aligned in a grid of N rows and M columns. Each of the points is named $P_{i,j}$ with its row number i ($1 \le i \le N$) and its column number j ($1 \le j \le M$).

Each point is colored either white or black. Your task is to compute the number of subsets T of white points satisfying the following conditions:

For any $3 \le i \le N$ and any $2 \le j \le M$ if the point (i, j) is in T, then point $(i-2, j-1)$ cannot be in T.

For any $3 \le i \le N$ and any $1 \le j \le M-1$ if the point (i, j) is in T, then point $(i-2, j+1)$ cannot be in T.

For example, in the figure below, if you include the point marked with a star to the set T, you can include neither of the two crossed points.



Because the answer can be too big, output the remainder divided by 998 244 353.

Input

The input consists of multiple datasets, each in the following format.

The first line of each dataset contains two integers N ($2 \le N \le 60$) and M ($2 \le M \le 60$) representing the numbers of rows and columns of the grid, respectively. Each of the following N lines contains M characters. The j-th character of the i-th line, $a_{i,j}$, represents the color of the point (i, j), where '`.`' denotes white and '`X`' denotes black.

The end of the input is indicated by a line containing two zeros.

The number of datasets does not exceed 100.

Output

For each dataset, output one line containing the number of subsets satisfying the conditions modulo 998 244 353.

Examples

standard input	standard output
3 2 .X XX X. 3 3 .X. X.X ..X 6 5 .X..X ...X. X.... ..X.X X.... ..X.. 0 0	3 20 103320

Problem L. Linear Combination of Polygons

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

The sum of two points in the Cartesian plane, (x_1, y_1) and (x_2, y_2) is defined as $(x_1 + x_2, y_1 + y_2)$. A polygon is treated as the set of all the points on its boundary and in its inside. Here, the sum of two polygons S_1 and S_2, $S_1 + S_2$, is defined as the set of all the points $v_1 + v_2$ satisfying $v_1 \in S_1$ and $v_2 \in S_2$.

Sums of two convex polygons, thus defined, are always convex polygons! The multiplication of a non-negative integer and a polygon is defined inductively by $0 \cdot S = \{(0, 0)\}$ and, for a positive integer k, $k \cdot S = (k - 1) \cdot S + S$. In this problem, line segments and points are also considered as convex polygons.

For two polygons P and Q with coordinates in integer points, and four integers a, b, c and d Laura built two polygons $R = aP + bQ$, and $S = cP + dQ$.

You have only coordinates of all vertices of R and S, and know that $ad - bc = 1$. Calculate the minimum possible sum of the areas of P and Q satisfying the equation.

Input

The input consists of at most 40 datasets, each in the following format.

First line of each test case contains two integers n and m, where n is the number of vertices of the convex polygon R, and m is number of vertices of the convex polygon S. n and m are integers and satisfy $3 \le n, m \le 1000$. Then n lines follow; i-th of those lines, (x_i, y_i) represents the coordinates of the i-th vertex of the convex polygon R. Then m lines follow; i-th of those lines, (x'_i, y'_i) represents the coordinates of the i-th vertes of of S.

x_i, y_i, x'_i and y'_i are integers between -10^6 and 10^6, inclusive.

The vertices of each convex polygon are given in counterclockwise order. Three different vertices of one convex polygon do not lie on a single line.

The end of the input is indicated by a line containing two zeros, which should not be processed.

Output

For each dataset, output a single line containing an integer representing the double of the sum of the areas of P and Q. Note that the areas are always integers when doubled because the coordinates of each vertex of P and Q are integers.

Examples

standard input	standard output

5 3
0 0
2 0
2 1
1 2
0 2
0 0
1 0
0 1
4 4
0 0
5 0
5 5
0 5
0 0
2 0
2 2
0 2
3 3
0 0
1 0
0 1
0 1
0 0
1 1
3 3
0 0
1 0
1 1
0 0
1 0
1 1
4 4
0 0
2 0
2 2
0 2
0 0
1 0
1 2
0 2
4 4
0 0
3 0
3 1
0 1
0 0
1 0
1 1
0 1
0 0
	3
	2
	2
	1
	0
	2

Problem M. Modifications

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

Ciel has a tree with N vertices. The weight of vertex i ($1 \le i \le N$) is v_i. The j-th ($1 \le j \le N-1$) edge connects vertex a_i and vertex b_i, and its weight is w_i.

In addition, the weight of the entire tree is determined as the remainder of $\sum_{1 \le a < b \le N} dist(a,b) \cdot v_a \cdot v_b$ while divided by 998 244 353.

Here, $dist(a,b)$ represents the sum of the weights of the edges existing on the shortest path from vertex a to vertex b.

First of all, Ciel wants to know the weight of the whole tree he has now. In addition, you will be given Q change queries. There are two types of queries:

- Vertex query: Increases the weight of vertex c ($1 \le c \le N$) by x. After that, the weight of the entire current tree is printed.
- Edge query: Increases the weight of edge e ($1 \le e \le N-1$) by x. After that, the weight of the entire current tree is printed.

Your task is to process these queries.

Input

First line of the input contains one integer N ($1 \le N \le 2 \cdot 10^5$). Second line contains n integers v_i ($1 \le v_i \le 10^8$); i-th of those integers is weight of vertex i.

Each of next $N-1$ lines contains three integers a_i, b_i and w_i — numbers of the vertices, connected by the edge, and weight of the edge, respectively ($1 \le a_i, b_i \le N$, $1 \le w_i \le 10^8$, edges form a tree).

Then line with integer Q follows ($1 \le Q \le 10^5$).

Each of the following Q lines contains one query.

The vertex query have format `1` c x, where c is number of vertex ($1 \le c \le N$) and $1 \le x \le 10^8$.

The edge query have format `2` e x, where e is number of edge ($1 \le e \le N-1$) and $1 \le x \le 10^8$.

There are no more than $2 \cdot 10^4$ queries of type 1.

Output

Print $Q+1$ lines. At the first line, print the weight of the whole tree in the initial state. Then print Q lines — weight of the entire tree after each query.

Examples

standard input	standard output

3
1 2 4
1 2 1
2 3 2
2
1 1 1
2 2 2
	30
	44
	76

4
2 2 3 3
1 2 4
1 3 3
1 4 1
3
2 3 4
1 4 1
2 1 10
	148
	232
	284
	464