Problem A. Balanced Strings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

You are given an undirected tree with n nodes. The nodes are numbered 1 through n. Each node is labeled with either '(' or ')'. Let $l[u \to v]$ denote the string obtained by concatenating the labels of the nodes on the simple path from u to v. (Note that the simple path between two nodes is uniquely determined on a tree.)

A balanced string is defined as follows:

- The empty string is balanced.
- For any balanced string s, the string "(" s ")" is balanced.
- For any balanced strings s and t, the string st (the concatenation of s and t) is balanced.
- Any other string is NOT balanced.

Calculate the number of the ordered pairs of the nodes (u, v) such that $l[u \to v]$ is balanced.

Input

The input consists of a single test case. The input starts with an integer n ($2 \leq n \leq 10^5$), which is the number of nodes of the tree. The next line contains a string of length n, each character of which is either '(' or ')'. The x-th character of the string represents the label of the node x of the tree. Each of the following $n - 1$ lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq n$), which represents that the node a_i and the node b_i are connected by an edge. The given graph is guaranteed to be a tree.

Output

Display a line containing the number of the ordered pairs (u, v) such that $l[u \to v]$ is balanced.

Examples

standard input	standard output
2 () 1 2	1
4 (()) 1 2 2 3 3 4	2
5 ()()) 1 2 2 3 2 4 1 5	4

Problem B. Card Game Strategy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

Alice and Bob are going to play a card game. There are n cards, each having an integer written on it. The game proceeds as follows:

1. Alice chooses an integer between a and b, inclusive. Call this integer t. Alice then tells Bob the value of t.
2. Bob chooses k out of the n cards. Compute the sum of the integers written on the k cards Bob chooses. Call this sum u.

Alice's objective is to make $|t - u|$ as large as possible and Bob's is to make $|t - u|$ as small as possible. Prior to the game, both Alice and Bob know the values of n, k, a, and b, and also the integers on the cards. Both Alice and Bob will play optimally. In particular, Alice will make a choice, knowing that Bob will surely minimize $|t - u|$ for told t. Additionally, assume that Alice prefers to choose smaller t if she has multiple equally good choices.

Your task is to determine the outcome of the game: the value of t Alice will choose and the k cards Bob will choose for that t.

Input

The input consists of two lines representing a single test case. The first line contains four integers n, k, a, and b ($1 \le k \le n \le 600$, $0 \le a \le b \le 1.8 \cdot 10^5$). The second line contains n integers $x_1, \ldots, x_n$ ($0 \le x_i \le 300$), denoting that x_i is written on the i-th card.

Output

Display two lines: The first line should contain an integer representing the value of t Alice will choose. The second line should contain k distinct integers between 1 and n, inclusive, representing the indices of the cards Bob will choose. If Bob has multiple equally good choices, display any one of them.

Examples

standard input	standard output
4 2 58 100 10 10 50 80	75 2 3
8 3 1300 1800 2 0 1 5 0 4 1 9	1800 4 6 8

Problem C. Casino

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Taro, who owes a debt of n dollars, decides to make money in a casino, where he can double his wager with probability p percent in a single play of a game. Taro is going to play the game repetitively. He can choose the amount of the bet in each play, as long as it is a positive integer in dollars and at most the money in his hand.

Taro possesses m dollars now. Find out the maximum probability and the optimum first bet that he can repay all his debt, that is, to make his possession greater than or equal to his debt.

Input

The input consists of a single test case, which consists of three integers p, m, and n separated by single spaces ($0 \leq p \leq 100$, $0 < m < n \leq 10^9$).

Output

Display three lines: The first line should contain the maximum probability that Taro can repay all his debt. This value must have an absolute error at most 10^{-6}. The second line should contain an integer representing how many optimum first bets are there. Here, a first bet is optimum if the bet is necessary to achieve the maximum probability. If the number of the optimum first bets does not exceed 200, the third line should contain all of them in ascending order and separated by single spaces. Otherwise the third line should contain the 100 smallest bets and the 100 largest bets in ascending order and separated by single spaces.

Examples

standard input	standard output
60 2 3	0.789473 1 1
25 3 8	0.109375 2 1 3

Problem D. Content Delivery

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

You are given a computer network with n nodes. This network forms an undirected tree graph. The i-th edge connects the a_i-th node with the b_i-th node and its distance is c_i. Every node has different data and the size of the data on the i-th node is s_i. The network users can deliver any data from any node to any node. Delivery cost is defined as the product of the data size the user deliver and the distance from the source to the destination. Data goes through the shortest path in the delivery. Every node makes cache to reduce the load of this network. In every delivery, delivered data is cached to all nodes which relay the data including the destination node. From the next time of delivery, the data can be delivered from any node with a cache of the data. Thus, delivery cost reduces to the product of the original data size and the distance between the nearest node with a cache and the destination.

Calculate the maximum cost of the m subsequent deliveries on the given network. All the nodes have no cached data at the beginning. Users can choose the source and destination of each delivery arbitrarily.

Input

The input consists of a single test case. The first line contains two integers n ($2 \leq n \leq 2000$) and m ($1 \leq m \leq 10^9$). n and m denote the number of the nodes in the network and the number of the deliveries, respectively. The following $n-1$ lines describe the network structure. The i-th line of them contains three integers a_i, b_i ($1 \leq a_i, b_i \leq n$) and c_i ($1 \leq c_i \leq 10^4$) in this order, which means the a_i-th node and the b_i-th node are connected by an edge with the distance c_i. The next line contains n integers. The j-th integer denotes s_j ($1 \leq s_j \leq 10^4$), which means the data size of the j-th node. The given network is guaranteed to be a tree graph.

Output

Display a line containing the maximum cost of the m subsequent deliveries in the given network.

Examples

standard input	standard output
3 2 1 2 1 2 3 2 1 10 100	320
2 100 1 2 1 1 1	2

Problem E. Cost Performance Flow

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Yayoi is a professional of money saving. Yayoi does not select items just because they are cheap; her motto is "cost performance". This is one of the reasons why she is good at cooking with bean sprouts. Due to her saving skill, Yayoi often receives requests to save various costs. This time, her task is optimization of "network flow".

Network flow is a problem on graph theory. Now, we consider a directed graph $G = (V, E)$, where $V = \{1, 2, \ldots, |V|\}$ is a vertex set and $E \in V \times V$ is an edge set. Each edge e in E has a capacity $u(e)$ and a cost $c(e)$. For two vertices s and t, a function $f_{s,t} : E \to R$, where R is the set of real numbers, is called an $s - t$ flow if the following conditions hold:

- For all e in E, $f_{s,t}$ is non-negative and no more than $u(e)$. Namely, $0 \leq f_{s,t}(e) \leq u(e)$ holds.
- For all v in $V \backslash \{s, t\}$, the sum of $f_{s,t}$ of out-edges from v equals the sum of $f_{s,t}$ of in-edges to v. Namely, $\sum_{e=(u,v)\in E} f_{s,t}(e) = \sum_{e=(v,w)\in E} f_{s,t}(e)$ holds.

Here, we define flow $F(f_{s,t})$ and cost $C(f_{s,t})$ of $f_{s,t}$ as $F(f_{s,t}) = \sum_{e=(s,v)\in E} f_{s,t}(e) - \sum_{e=u,s\in E} f_{s,t}(e)$ and $C(f_{s,t}) = \sum_{e\in E} f_{s,t}(e)c(e)$, respectively.

Usually, optimization of network flow is defined as cost minimization under the maximum flow. However, Yayoi's motto is "cost performance". She defines a balanced function $B(f_(s, t))$ for $s - t$ flow as the sum of the square of the cost $C(f_{s,t})$ and the difference between the maximum flow $M = \max_{f:s-tflow} F(f)$ and the flow $F(f_{s,t})$, i.e. $B(f_{s,t}) = C(f_{s,t})^2 + (M - F(f_{s,t}))^2$. Then, Yayoi considers that the best cost performance flow yields the minimum of $B(f_{s,t})$.

Your task is to write a program for Yayoi calculating $B(f^*_{s,t})$ for the best cost performance flow $f^*_{s,t}$.

Input

The input consists of a single test case. The first line gives two integers separated by a single space: the number of vertices N ($2 \leq N \leq 100$) and the number of edges M ($1 \leq M \leq 1000$). The second line gives two integers separated by a single space: two vertices s and t ($1 \leq s, t \leq N$, $s \neq t$). The i-th line of the following M lines describes the i-th edges as four integers a_i, b_i, u_i, and c_i: the i-th edge from a_i ($1 \leq a_i \leq N$) to b_i ($1 \leq b_i \leq N$) has the capacity u_i ($1 \leq u_i \leq 100$) and the cost c_i ($1 \leq c_i \leq 100$). You can assume that $a_i \neq b_i$ for all $1 \leq i \leq M$, and $a_i \neq a_j$ or $b_i \neq b_j$ if $i \neq j$ for all $1 \leq i, j \leq M$.

Output

Display $B(f^*_{s,t})$, the minimum of balanced function under $s-t$ flow, as a fraction in a line. More precisely, output u/d, where u is the numerator and d is the denominator of $B(f^*_{s,t})$, respectively. Note that u and d must be non-negative integers and relatively prime, i.e. the greatest common divisor of u and d is 1. You can assume that the answers for all the test cases are rational numbers.

Examples

standard input	standard output
2 1 1 2 1 2 1 1	1/2
3 3 1 2 1 2 1 1 1 3 3 1 3 2 3 2	10/1
3 3 1 2 1 2 1 1 1 3 7 1 3 2 7 1	45/1

Problem F. ICPC teams

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

You are a coach of the International Collegiate Programming Contest (ICPC) club in your university. There are $3N$ students in the ICPC club and you want to make N teams for the next ICPC. All teams in ICPC consist of 3 members. Every student belongs to exactly one team.

When you form the teams, you should consider several relationships among the students. Some student has an extremely good relationship with other students. If they belong to a same team, their performance will improve surprisingly. The contrary situation also occurs for a student pair with a bad relationship. In short, students with a good relationship must be in the same team, and students with a bad relationship must be in different teams. Since you are a competent coach, you know all M relationships among the students.

Your task is to write a program that calculates the number of possible team assignments. Two assignments are considered different if and only if there exists a pair of students such that in one assignment they are in the same team and in the other they are not.

Input

The input consists of a single test case. The first line contains two integers N ($1 \leq N \leq 10^6$) and M ($1 \leq M \leq 18$). The i-th line of the following M lines contains three integers A_i, B_i ($1 \leq A_i, B_i \leq 3N$, $A_i \neq B_i$), and C_i ($C_i \in \{0, 1\}$). A_i and B_i denote indices of the students and C_i denotes the relation type. If C_i is 0, the A_i-th student and the B_i-th student have a good relation. If C_i is 1, they have a bad relation. You can assume that $\{A_i, B_i\} \neq \{A_j, B_j\}$ if $i \neq j$ for all $1 \leq i, j \leq M$.

Output

Display a line containing the number of the possible team assignments modulo $10^9 + 9$.

Examples

standard input	standard output
2 2 1 2 0 3 4 1	2

Problem G. JAG Channel II

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

JAG (Japan Alumni Group) is a group of N members that devotes themselves to activation of the competitive programming world. The JAG staff members talk every day on the BBS called JAG-channel. There are several threads in JAG-channel and these are kept sorted by the time of their latest posts in descending order.

One night, each of the N members, identified by the first N uppercase letters respectively, created a thread in JAG-channel. The next morning, each of the N members posted in exactly K different threads which had been created last night. Since they think speed is important, they viewed the threads from top to bottom and posted in the thread immediately whenever they came across an interesting thread. Each member viewed the threads in a different period of time, that is, there was no post of other members while he/she was submitting his/her K posts.

Your task is to estimate the order of the members with respect to the periods of time when members posted in the threads. Though you do not know the order of the threads created, you know the order of the posts of each member. Since the threads are always kept sorted, there may be invalid orders of the members such that some members cannot post in the top-to-bottom order of the threads due to the previous posts of other members. Find out the lexicographically smallest valid order of the members.

Input

The input consists of a single test case. The first line contains two integers separated by a space: N ($4 \leq N \leq 16$) and K ($N - 3 \leq K \leq N - 1$). Then N lines of strings follow. Each of the N lines consists of exactly K distinct characters. The j-th character of the i-th line denotes the j-th thread in which the member denoted by the i-th uppercase letter posted. Each thread is represented by its creator (e.g. '`B`' represents the thread created by member B, the second member). It is guaranteed that at least one valid order exists.

Output

Display a string that consists of exactly N characters in a line, which denotes the valid order in which the members posted in the threads. The i-th character of the output should represent the member who posted in the i-th period. In case there are multiple valid orders, output the lexicographically smallest one.

Examples

standard input	standard output
7 4 DEFG FEDA EFGB BGEA AGFD DABC CADE	ABCDEFG
4 3 CDB DAC BAD ABC	DCBA
16 13 NDHPFJIBLMCGK CMDJKPOLGIHNE MOLBIEJFPHADN KPNAOHBLMCGEI FCMLBHDOANJPK NHIGLOAPKJDMC KMLBIPHDEOANJ IEGCMLBOAPKJD JNAOEDHBLMCGF OEDHPFIBLMGKC GMLBIFPHDNAEO ENHGOPKJDMCAF JKPAOBLGEIHNF HPKFGJEIBLCOM LBINEJDAGFKPH FGMOCADJENIBL	PONCAKJGIEDHMFBL

Problem H. Kimagure Cleaner

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Ichiro won the newest model cleaner as a prize of a programming contest. This cleaner automatically moves around in a house for cleaning. Because Ichiro's house is very large, it can be modeled as an infinite two-dimensional Cartesian plane, whose axes are called X and Y. The positive direction of the Y-axis is to the left if you face the positive direction of the X-axis.

The cleaner performs a sequence of actions for cleaning. Each action consists of a turn and a run. In an action, first the cleaner turns right or left by 90 degrees, and then runs straight by an integer length to the direction that the cleaner faces. At the end of a day, the cleaner reports the log of its actions in the day to Ichiro, in order to inform him where it has cleaned and where it hasn't.

Unlike common cleaners, this cleaner has human-like artificial intelligence. Therefore, the cleaner is very forgetful (like humans) and it is possible that the cleaner forgets the direction of a turn, or the cleaner only remembers the length of a run as a very rough range. However, in order to pretend to be operating correctly, the cleaner has to recover the complete log of actions after finishing the cleaning. The cleaner was initially at the point $(0, 0)$, facing the positive direction of X-axis. You are given the cleaner's location after cleaning, (X, Y), and an incomplete log of the cleaner's actions that the cleaner remembered. Please recover a complete log from the given incomplete log. The actions in the recovered log must satisfy the following constraints:

- The number of actions must be the same as that in the incomplete log.
- The direction of the i-th turn must be the same as that in the incomplete log if it is recorded in the incomplete log.
- The length of the i-th run must be within the range of the length specified in the incomplete log.
- The cleaner must be at (X, Y) after finishing all actions. The direction of the cleaner after cleaning is not important and you do not have to care about it, because the cleaner can turn freely after cleaning, though it cannot run after cleaning. You are not required to recover the actual path, because Ichiro only checks the format of the log and the location of the cleaner after cleaning.

Input

The input consists of a single test case. The first line contains three integers N ($1 \leq N \leq 50$), X, and Y ($-10^9 \leq X, Y \leq 10^9$). N represents the number of actions in the incomplete log. X and Y represent the cleaner's location (X, Y) after cleaning. The i-th line of the following N lines contains a character D_i and two integers LL_i and LU_i. D_i represents the direction of the i-th turn: '`L`', '`R`', and '`?`' represents left, right, and not recorded respectively. LL_i and LU_i represent a lower and upper bound of the length of the i-th run, respectively. You can assume $1 \leq LL_i \leq LU_i \leq 55\,555\,555$.

Output

Display the recovered log. In the first line, display N, the number of actions in the log. In the i-th line of the following N lines, display the direction of the i-th turn in a character and the length of the i-th run separated by a single space. Represent a right turn by a single character '`R`', and a left turn by a single character '`L`'. The recovered log must satisfy the constraints in the problem. If there are multiple logs that satisfy the constraints, you can display any of them. Display -1 in a line if there is no log that satisfies the constraints.

Examples

standard input	standard output
2 -3 4 L 2 5 ? 3 5	2 L 4 L 3
5 3 -4 ? 1 5 ? 1 5 ? 1 5 ? 1 5 ? 1 5	5 L 1 R 2 R 4 L 1 R 1

Problem I. Midpoint

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	256 mebibytes

One day, you found $L + M + N$ points on a 2D plane, which you named $A_1, \ldots, A_L$, $B_1, \ldots, B_M$, $C_1, \ldots, C_N$. Note that two or more points of them can be at the same coordinate. These were named after the following properties:

- the points $A_1, \ldots, A_L$ were located on a single straight line,
- the points $B_1, \ldots, B_M$ were located on a single straight line, and
- the points $C_1, \ldots, C_N$ were located on a single straight line.

Now, you are interested in a triplet (i, j, k) such that C_k is the midpoint between A_i and B_j. Your task is counting such triplets.

Input

The first line contains three space-separated positive integers L, M, and N ($1 \leq L, M, N \leq 10^5$). The next L lines describe A. The i-th of them contains two space-separated integers representing the x-coordinate and the y-coordinate of A_i. The next M lines describe B. The j-th of them contains two space-separated integers representing the x-coordinate and the y-coordinate of B_j. The next N lines describe C. The k-th of them contains two space-separated integers representing the x-coordinate and the y-coordinate of C_k.

It is guaranteed that the absolute values of all the coordinates do not exceed 10^5.

Output

Print the number of the triplets which fulfill the constraint.

Examples

standard input	standard output
2 2 3 0 0 2 0 0 0 0 2 0 0 1 1 1 1	3
4 4 4 3 5 0 4 6 6 9 7 8 2 11 3 2 0 5 1 4 3 7 4 10 5 1 2	8
4 4 4 0 0 3 2 6 4 9 6 7 14 9 10 10 8 13 2 4 2 5 4 6 6 8 10	3

Problem J. Yet Another Game AI

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Aoba is a beginner programmer who works for a game company. She was appointed to develop a battle strategy for the enemy AI (Artificial Intelligence) in a new game. In this game, each character has two parameters, hit point (hp) and defence point (dp). No two characters have the same hp and dp at the same time. The player forms a party by selecting one or more characters to battle with the enemy. Aoba decided to develop a strategy in which the AI attacks the weakest character in the party: that is, the AI attacks the character with the minimum hit point in the party (or, if there are several such characters, the character with the minimum defense point among them). She wrote a function $selectTarget(v)$ that takes an array of characters representing a party and returns a character that her AI will attack.

However, the project manager Ms. Yagami was not satisfied with the behavior of her AI. Ms. Yagami said this AI was not interesting.

Aoba struggled a lot, and eventually she found that it is interesting if she substitutes one of the constant zeros in her program with a constant C. The rewritten program is as follows. Note that $Character$ is a type representing a character and has fields hp and dp which represent the hit point and the defense point of the character respectively.

```
int C = <constant integer>;
Character selectTarget(Character v[]) {
  int n = length(v);
  int r = 0;
  for (int i = 1; i < n; i++) {
    if (abs(v[r].hp - v[i].hp) > C) {
      if (v[r].hp > v[i].hp) r = i;
    } else {
      if (v[r].dp > v[i].dp) r = i;
    }
  }
  return v[r];
}
```

By the way, this function may return different characters according to the order of the characters in v, even if v contains the same set of characters. Ms. Yagami wants to know how many characters in a party may become the target of the new AI. Aoba's next task is to write a program that takes a given party v and a constant C, and then counts the number of characters that may become the return value of $selectTarget(v)$ if v is re-ordered.

Input

The input consists of a single test case. The first line contains two integers N ($1 \leq N \leq 5 \cdot 10^4$) and C ($0 \leq C \leq 10^9$). The first integer N represents the size of v. The second integer C represents the constant C in Aoba's program. The i-th line of the following N lines contains two integers hp_i and dp_i ($0 \leq hp_i, dp_i \leq 10^9$). hp_i represents the hit point of the i-th character in v, and dp_i represents the defense point of the i-th character in v. You can assume that $hp_i \neq hp_j$ or $dp_i \neq dp_j$ if $i \neq j$ for any $1 \leq i, j \leq N$.

Output

Display the number of characters that may become the return value of $selectTarget(v)$, if v is shuffled in an arbitrary order.

Examples

standard input	standard output
5 3 1 5 3 4 5 3 7 2 9 1	5
3 2 1 2 3 1 5 1	2
4 1 2 0 0 4 1 1 5 9	3
5 9 4 10 12 4 2 14 9 19 7 15	3

Problem K. Runner and Sniper

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are escaping from an enemy for some reason. The enemy is a sniper equipped with a high-tech laser gun, and you will be immediately defeated if you get shot. You are a very good runner, but just wondering how fast you have to run in order not to be shot by the sniper. The situation is as follows: You and the sniper are on the xy-plane whose x-axis and y-axis are directed to the right and the top, respectively. You can assume that the plane is infinitely large, and that there is no obstacle that blocks the laser or your movement.

The sniper and the laser gun are at $(0, 0)$ and cannot move from the initial location. The sniper can continuously rotate the laser gun by at most ω degrees per unit time, either clockwise or counterclockwise, and can change the direction of rotation at any time. The laser gun is initially directed φ degrees counterclockwise from the positive direction of the x-axis.

You are initially at (x, y) on the plane and can move in any direction at speed not more than v (you can arbitrarily determine the value of v since you are a very good runner). You will be shot by the sniper exactly when the laser gun is directed toward your position, that is, you can ignore the time that the laser reaches you from the laser gun. Assume that your body is a point and the laser is a half-line whose end point is $(0, 0)$.

Find the maximum speed v at which you are shot by the sniper in finite time when you and the sniper behave optimally.

Input

The input consists of a single test case. The input contains four integers in a line, x, y, φ, and ω. The two integers x and y ($0 \leq |x|, |y| \leq 1000$, $(x, y) \neq (0, 0)$) represent your initial position on the xy-plane. The integer φ ($0 \leq \varphi < 360$) represents the initial direction of the laser gun: it is the counterclockwise angle in degrees from the positive direction of the x-axis. The integer ω ($1 \leq \omega \leq 100$) is the angle which the laser gun can rotate in unit time. You can assume that you are not shot by the sniper at the initial position.

Output

Display a line containing the maximum speed v at which you are shot by the sniper in finite time. The absolute error or the relative error should be less than 10^{-6}.

Examples

standard input	standard output
100 100 0 1	1.16699564

Problem L. Wall Making Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The game Wall Making Game, a two-player board game, is all the rage.

This game is played on an $H \times W$ board. Each cell of the board is one of *empty*, *marked*, or *wall*. At the beginning of the game, there is no wall on the board.

In this game, two players alternately move as follows:

1. A player chooses one of the empty cells (not marked and not wall). If the player can't choose a cell, he loses.
2. Towards each of the four directions (upper, lower, left, and right) from the chosen cell, the player changes cells (including the chosen cell) to walls until the player first reaches a wall or the outside of the board.

Note that marked cells cannot be chosen in step 1, but they can be changed to walls in step 2. The picture below shows an example of a move in which a player chooses the cell at the third row and the fourth column.

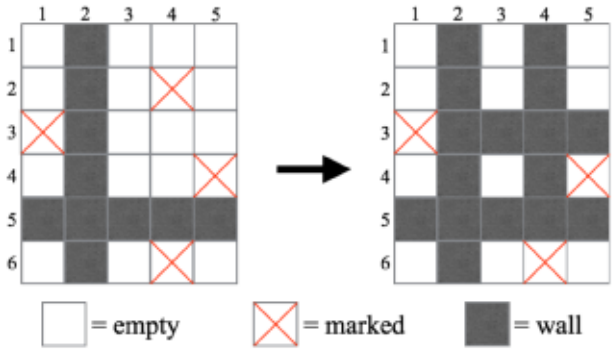

fig.1: An example of a move in Wall Making Game.

Your task is to write a program that determines which player wins the game if the two players play optimally from a given initial board.

Input

The first line of the input consists of two integers H and W ($1 \leq H, W \leq 20$), where H and W are the height and the width of the board respectively. The following H lines represent the initial board. Each of the H lines consists of W characters.

The j-th character of the i-th line is '`.`', if the cell at the j-th column of the i-th row is empty, or '`X`' if the cell is marked.

Output

Print "`First`" (without the quotes) in a line if the first player wins the given game. Otherwise, print "`Second`" (also without the quotes) in a line.

Examples

standard input	standard output
2 2	Second
2 2 X. ..	First
4 5 X.... ...X.	First