Problem A. Balls (Division 1 Only!)

Developer:	Anton Maydell
Input file:	`balls.in`
Output file:	`balls.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Breadth-first search can work up to ten times longer, as some PHP scripts do.

Consider a rectangular board N cells in width and M cells in height. Two to four balls are placed in the cells of the board. Each ball is either black or white. At any moment of time, no two balls can occupy the same cell. Cells with no balls in them are either empty or occupied by a wall. A move consists in pushing a ball in one of the directions: up, down, left or right. After the ball is pushed, it moves in the given direction until it reaches either the edge of the board, another ball or a wall.

The goal is to make the minimal number of moves so that the balls are placed as in the given pattern. The position on the board matches the pattern if there exists a parallel shift of the pattern such that after the shift, the positions and colors of all balls in the pattern and in the labyrinth coincide. The pattern can be neither rotated nor reflected. Parts of the pattern that do not contain any balls can coincide with the exterior of the board.

Input

The input consists of a single test case. The first line of input contains two integers M and N, height and width of the board ($5 \le M, N \le 8$). The next M lines contain N characters each and describe the board. Walls are marked with a hash ('`#`'), empty cells are marked with a dot ('`.`'), black balls with digit '`0`' and white balls with digit '`1`'. The total number of balls on the board is no less than two and no more than four.

The next line contains two integers H and W, height and width of the pattern ($1 \le H, W \le 5$). The next H lines contain W characters each and describe the pattern. The colors of balls in the pattern are also marked with digits '`0`' and '`1`', all other characters of the pattern are asterisks ('`*`'). Asterisks correspond to either empty cells, walls or cells outside the board. The number and colors of balls in the pattern and on the board are equal. It is guaranteed that the initial position on the board does not match the pattern.

Output

On the first line of output, write the minimal number of moves required to solve the problem. On the next lines, describe the moves themselves. The description of a single move consists of the position (row number and column number) of a ball and the direction in which it is pushed. The rows are numbered from highest to lowest starting from 1, the columns are numbered from leftmost to rightmost starting from 1. The direction is written as follows: '`D`' means pushing down, '`U`' means pushing up, '`L`' to the left and '`R`' to the right. It is guaranteed that in all tests given to your program, the solution exists.

Examples

balls.in	balls.out
5 6 .....1 .####. ....#. .####. 0....0 2 2 00 1*	8 5 1 U 1 1 R 5 6 L 5 1 U 1 5 L 1 6 D 5 6 L 5 1 U
5 5 ....0 #...1 .... 3 4 **** *0** **1*	2 1 5 L 2 5 L

Problem B. Bridges in a Tree (Division 1 Only!)

Developer:	Sergey Kopeliovich
Input file:	`bridges2.in`
Output file:	`bridges2.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

December is here, it's time for... Exams!

Santa Claus

Boy Serezha has almost graduated. There are two huge tasks left: first, to pass the winter exams, and second, to write the research work. The exams are not a problem: Serezha has already tried to pass them a year ago. But the research work is much more complicated.

Serezha selected the research topic two years ago: it is the "Dynamic 2-Connectivity Problem". This is a problem about dynamically changing a graph and responding to queries like "the number of bridges in the graph at the given moment".

Serezha has invented several solutions. One of them, the simpliest one for coding, has an unpleasant subtask. Serezha wrote the code, but it seemed to be overcomplicated.

So, he wondered, if there's another simplier implementation. A shorter, but as fast as his one. To find out the answer for this question, he decided to offer this task to a University Championship.

There are only a few months left before the time to present the research work, and Serezha is going to code much more funny algorithms, and the time is running out. And if he does not manage to finish his work in time, he will have to pass the whole fifth year of study. Again.

Please help Serezha to face this unpleasant subtask.

Recall that a *tree* is an undirected connected graph without cycles. A *bridge* is an edge of an undirected graph that increases the number of its connected components when removed.

Given is a tree `YourTree` of N ($1 \le N \le 100\,000$) vertices. Your task is to calculate the number of bridges in the graphs which are produced by adding sets of K_i edges to the tree `YourTree`.

Input

The first line contains integers N and M ($1 \le N \le 100\,000$, $1 \le M \le 100\,000$): the number of vertices and the number of requests.

The second line contains the description of the tree `YourTree`: there are $N - 1$ integers p_2, p_3, ..., p_N separated by single spaces which represent edges $(2, p_2)$, $(3, p_3)$, ..., (N, p_N). It is guaranteed that these edges form a tree. You may assume that $1 \le p_i < i$.

M requests follow, one per line. The request number i is described by a non-negative integer K_i and K_i pairs of integers 1 through N. Each pair describes an edge to be added to the tree `YourTree`.

The sum K_i in all requests does not exceed 100 000.

Note that the graphs may contain loops and multiple edges. Remember that a multiple edge can never be a bridge.

Output

For each request, write a single integer: the number of bridges in the graph produced by adding the given edges to the tree `YourTree`.

Example

bridges2.in	bridges2.out
7 8 1 1 2 2 3 3 1 4 5 3 4 5 6 7 3 2 1 5 6 1 1 1 1 3 6 2 4 3 2 7 1 5 1 3 1 2 1 3 1 6	4 0 2 6 5 2 4 3

Problem C. Smooth Numbers (Division 1 Only!)

Developer:	Sergey Kopeliovich
Input file:	`bsmooth.in`
Output file:	`bsmooth.out`
Time limit:	4 seconds (*10 seconds for Java*)
Memory limit:	256 mebibytes

So smooth and downy...

Number theory book for high school

A natural number is said to be *B-smooth* if and only if all its prime divisors do not exceed B.

Write a program that will, given the numbers B and N, find the number of *B-smooth* numbers not exceeding N.

Input

The only line of input contains two integers N and B ($1 \le N \le 8 \cdot 10^{18}$, $2 \le B \le 100$).

Output

Write the only integer: the number of *B-smooth* numbers not exceeding N.

Example

bsmooth.in	bsmooth.out
1000 10	141

Problem D. Card House (Division 1 Only!)

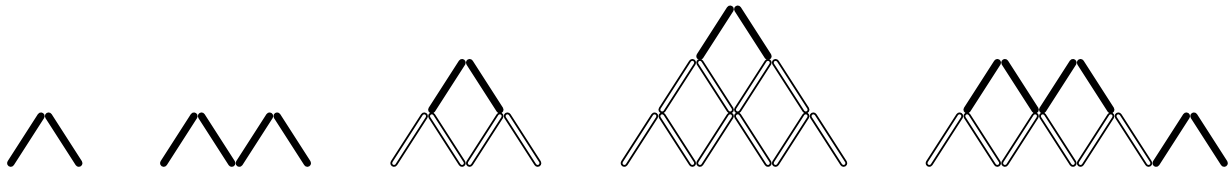
Developer:	Olga Bursian
Input file:	`cardhouse.in`
Output file:	`cardhouse.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Hedgehog wants to build a card house before Bear comes to visit him. This house is built according to the following recursive rules.

1. Two cards leaning towards each other which touch by their upper edges and form an angle are called a *card angle*.
2. A card angle is a card house.
3. Two card houses standing side by side form another card house.
4. If two card angles belong to a card house and stand side by side on the same height, putting a card angle on top of them so that the lower edges of its cards rest upon the upper edges of these card angles produces another card house.

A *peak* is a card angle in a card house which has no other card angles placed on top of it.

Various examples of card houses are pictured below. Darker card angles represent their peaks.



Hedgehog has $2N$ cards and wants to build a card house using all these cards in such a way that the number of peaks is minimal possible. Find this minimal number to help him.

Input

The input consists of one integer N ($0 \le N \le 1\,000\,000$).

Output

Write the minimal number of peaks in a card house which could be built using exactly $2N$ cards.

Examples

cardhouse.in	cardhouse.out
3	1
4	2

Problem E. Knights and Liars (Division 1 Only!)

Developer:	Ivan Kazmenko
Input file:	`knl.in`
Output file:	`knl.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

There are n inhabitants on the Hope Island. Each of them is either a knight who always speaks the truth, or a liar whose statements are always false. Each of these people has a unique integer number from 1 to n assigned to him.

Once a year the people gather at the center of the island, and after that each of them makes exactly one statement of exactly one of the following forms: "inhabitant with a certain number is a knight" or "inhabitant with a certain number is a liar".

This year, Vasya came to visit the island at the start of the meeting. He heard and recorded some statements, after which the people went to take a lunch break. Right now, Vasya wants to simulate the further development of events.

Let a *protocol* be a list of n statements: the statement of the first inhabitant of the island, the statement of the second one, the statement of the third one, ..., the statement of the last inhabitant. In Vasya's model, each of the inhabitants which did not yet say anything will say one of the possible $2n$ statements independently and with uniform probability. The possible statements are: "first inhabitant is a knight", "first inhabitant is a liar", "second inhabitant is a knight", ..., "n-th inhabitant is a liar". After the protocol is complete, Vasya will search for a *solution*: for each of the inhabitants, he wants to determine whether this inhabitant is a knight or a liar, so that all statements of the protocol are correct.

Note that the number of possible solutions can be different for different protocols. In particular, there are protocols for which no solutions exist at all: for example, if a certain inhabitant says about himself that he is a liar, there are no solutions no matter what and about whom all the other inhabitants say. On the contrary, if in the protocol, each inhabitant says about himself that he is a knight, each of them can be either a knight or a liar independently of the others, so the number of possible solutions is 2^n.

Now Vasya asked you to estimate his chances to get a solution. Consider the moment of time when the protocol is already fixed, i. e. the missing statements are chosen uniformly and independently, and it is time to find a solution. Vasya is interested in two values. The first one is the expected number of solutions. The second is the probability that at least one solution exists.

Unfortunately, Vasya could have made a mistake while recording the statements on the meeting. So, the initial data could be contradictory and, as a result, it may happen that there are no solutions for any of the possible protocols. In this case, the two required values are zeroes.

Recall that the expected value of a random variable is a sum over all possible outcomes (in our case, these are protocols) of the probability of an outcome multiplied by the value of this variable (in our case, the number of solutions) for this outcome.

Input

The first line of input contains two integers n and k separated by a space: the number of inhabitants and the number of statements Vasya heard ($1 \le n \le 50$, $0 \le k \le n$). The next k lines describe the statements. Each of these lines has the form i j s where i is the number of inhabitant which

pronounced this statement ($1 \le i \le n$), j is the number of inhabitant this statement refers to ($1 \le j \le n$), and s is either "`knight`" or "`liar`" which is what inhabitant i said about inhabitant j.

It is guaranteed that all the statements are produced by different inhabitants. Remember that the data could be contradictory.

Output

On the first line of output, write two numbers separated by a space: the expected number of solutions and the probability that at least one solution exists. The answer is considered correct if it differs from the exact one by no more than 10^{-9}.

Example

knl.in	knl.out
3 2 1 2 liar 3 1 knight	1.0000000000 0.5000000000

Explanation

In this example, inhabitant 1 says that inhabitant 2 is a liar and inhabitant 3 says that inhabitant 1 is a knight. There are six possible protocols. They all differ only by what inhabitant 2 said. Each of them has a probability of $\frac{1}{6}$. The number of possible solutions is shown in brackets. The solutions themselves are coded by strings of three characters describing inhabitants in natural order (`K` for knight, `L` for liar).

1. Inhabitant 2 says that inhabitant 1 is a knight (0 solutions).
2. Inhabitant 2 says that inhabitant 1 is a liar (2 solutions: `KLK` and `LKL`).
3. Inhabitant 2 says that inhabitant 2 is a knight (2 solutions: `KLK` and `LKL`).
4. Inhabitant 2 says that inhabitant 2 is a liar (0 solutions).
5. Inhabitant 2 says that inhabitant 3 is a knight (0 solutions).
6. Inhabitant 2 says that inhabitant 3 is a liar (2 solutions: `KLK` and `LKL`).

The expected number of solutions is $\frac{1}{6} \cdot 0 + \frac{1}{6} \cdot 2 + \frac{1}{6} \cdot 2 + \frac{1}{6} \cdot 0 + \frac{1}{6} \cdot 0 + \frac{1}{6} \cdot 2 = 1$. The probability that at least one solution exists is $0 + \frac{1}{6} + \frac{1}{6} + 0 + 0 + \frac{1}{6} = \frac{1}{2}$.

Problem F. Longest Common Subsequence (Division 1 Only!)

Developer:	Anton Maydell
Input file:	`lcs.in`
Output file:	`lcs.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Finding longest common subsequence is a well-known problem, isn't it?

Research Institute of Given Strings (RIGS) needs a program to search for the greatest common subsequence of two arrays of a special kind. The length of each array is $2N$. Both arrays consist of integers from 1 to N, furthermore, each of these integers appears in each of the arrays exactly twice.

Vasya claimed he can write this program but failed. Help him!

Input

The input consists of one or more test cases.

Each test case consists of three lines. The first line contains an integer N ($1 \le N \le 50\,000$). The second line contains $2N$ integers separated by spaces: the contents of the first array. The third line contains $2N$ integers separated by spaces: the contents of the second array. It is guaranteed that each of the numbers 1, 2, ..., N occurs in each of the arrays exactly twice.

The total number of test cases is no more than 1000. The sum of values of N in the input does not exceed 100 000.

Input is terminated by a line containing a single integer 0.

Output

For each test case, output two lines. On the first of these lines, output M, the length of the longest common subsequence. On the second line, output M integers separated by spaces: the subsequence itself. If there are several optimal answers, output any of them.

Example

lcs.in	lcs.out
2 1 2 1 2 2 1 2 1 3 1 1 2 2 3 3 3 3 2 2 1 1 0	3 1 2 1 2 1 1

Problem G. The Game of "Life" (Division 1 Only!)

Developer:	Sergey Kopeliovich
Input file:	`life.in`
Output file:	`life.out`
Time limit:	7 seconds (*8 seconds for Java*)
Memory limit:	256 mebibytes

> Play is connected with no material interest, and no profit can be gained from it
>
> Homo Ludens

Given is a grid $W \times H$. Every cell is initially either black or white. Two cells are said to be neighbours if and only if they share a side.

The following procedure is repeated:

- A white cell with at least two black neighbours is found. If there's no such cell, the process stops.
- The cell that was found is painted in black.

Write a program to calculate the number of different final paintings. As the answer can be huge, find its remainder modulo 10^{18}.

Input

The first line of input contains the grid size: two integers W and H ($1 \le W, H \le 13$). The following H lines contain W characters each and describe the grid:

- Character «`B`» means that the cell was initially painted black.
- Character «`.`» means that the initial color of the cell is unknown, i. e. it could be either black or white.

Output

Write the only integer: the remainder of the number of different final paintings modulo 10^{18}.

Example

life.in	life.out
3 4 B.B .B.	3

Explanation

The example allows three possible paintings (character «`W`» denotes white cells):

```
BBB   BBB   BBB
BBB   BBB   BBB
WWW   BBB   BBB
WWW   WWW   BBB
```

Problem H. Memcached (Division 1 Only!)

Developer:	Anton Maydell
Input file:	`memcached.in`
Output file:	`memcached.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Tiggers do not like chipmunks.

From an unpublished book about
Winnie-the-Pooh

Research Institute of Given Strings (RIGS) asked Vasya to implement a subset of *memcached* protocol. This subset contains only two commands: `SET` and `GET`. The `SET` command is used to assign a variable a certain value, and the `GET` command to output the value of a variable or to determine that the variable is not yet initialized.

Help Vasya! The details of the command syntax and the format of answers are shown below.

Input

The input consists of no more than 100 lines, each line consists of no more than 100 characters, and each of these characters has ASCII-code from 32 to 126. A *variable name* is a string consisting of non-whitespace characters which has length from 1 to 50 characters. Names of variables which differ in letter case are considered different. A variable is *initialized* if command `SET` is called at least once for this variable.

The `SET` command occupies two lines of input:

```
SET <variable name> 0 0 <value length>
<variable value>
```

The length of the second line is always equal to the value length given on the first line.

The `GET` command occupies one line of input:

```
GET <variable name>
```

The first lines of `GET` and `SET` commands do not contain two or more consecutive spaces. The names of commands always consist of capital letters. It is guaranteed that all commands in the input is correct and there are no empty lines between consecutive commands.

Output

For each `SET` command, output "`STORED`" on a separate line and memorize the value of the variable.

For each `GET` command for an initialized variable, output three lines:

```
VALUE <variable name> 0 <value length>
<variable value>
END
```

For each GET command for an uninitialized variable, output "END" on a separate line.

Example

memcached.in	memcached.out
GET empty SET puzzle 0 0 25 Do kittens eat chipmunks? GET puzzle SET puzzle 0 0 33 Shhhhhhhhhh, I'm trying thinking. GET puzzle	END STORED VALUE puzzle 0 25 Do kittens eat chipmunks? END STORED VALUE puzzle 0 33 Shhhhhhhhhh, I'm trying thinking. END

Problem I. Find the Path (Division 1 Only!)

Developer:	Natalya Ginzburg
Input file:	`pathfind.in`
Output file:	`pathfind.out`
Time limit:	3 seconds (*5 seconds for Java*)
Memory limit:	256 mebibytes

Consider an undirected weighted graph without loops and multiple edges. Given is a sequence of numbers. Find and output a path in the graph such that lengths of edges in it form the given sequence. If there are multiple such paths, output the lexicographically smallest one. In case there are no such paths, output "`No solution`" (without quotes).

Note that a number in the sequence may match no edge length in the graph.

A path in the graph is defined by a sequence of vertex numbers. Any neighboring vertices in the path should be connected by an edge in the graph.

A path comes earlier lexicographically than another path of the same length if and only if it has a smaller vertex number in the first position at which they differ.

Input

The first line of input contains three integers n, m and k separated by single spaces: the number of vertices, the number of edges of the graph and the length of the path ($2 \le n \le 500$, $1 \le m \le \frac{n(n-1)}{2}$, $1 \le k \le 500$).

Next m lines describe edges of the graph. Each row contains three integers u, v and l separated by single spaces: numbers of two adjacent vertices and the length of an edge connecting them ($1 \le u, v \le n$, $1 \le l \le 10^9$).

The next line contains k integers separated by single spaces: the given sequence. These numbers also are in the range from 1 to 10^9.

Output

In case there are no paths with given edge lengths, output "`No solution`". Othewise, on the first line output the path: sequence of $k+1$ integers containing the numbers of the first, second, ..., $k+1$-th vertex of the path. Vertex numbers are 1-based.

The path should be the lexicographically smallest one. Numbers should be separated by spaces.

Examples

pathfind.in	pathfind.out
3 3 3 1 2 10 2 3 20 1 3 30 30 10 20	3 1 2 3
3 3 3 1 2 10 2 3 20 1 3 30 30 10 30	No solution
3 3 3 1 2 10 2 3 20 1 3 30 30 10 40	No solution
4 5 6 1 2 10 2 3 10 1 3 10 1 4 10 2 4 10 10 10 10 10 10 10	1 2 1 2 1 2 1