Problem 1. Labyrinth

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
Memory limit:	256 megabyte
Feedback:	

Alex is seeing yet another dream — this time got stuck in a giant labyrinth.

Alex have spent some time running in the labyrinth, got tired. Now he wants to just lean on an labyrinth's wall to take a rest.

Labyrinth is a field of square cells. The length of a side of a cell is 1 kilometer. Also, lets assume we have we have X-axis and Y-axis.

However, this space is just similar, in fact it is non-Euclidian: For an infinetely small time δt Alex can move by a vector $(\delta x, \delta y)$, where

$$\max(|\delta x|, |\delta y|) \le \delta t,$$

(measuring distance in meters and time in seconds)

For a given position P Alex could reach a wall for the minimal time $T_{min}(P)$. However, we don't know Alexes position. He could be anywhere inside the labyrinth. Let's consider the probability of his position is evenly distrubuted across all the empty cells. We need to find out expected value of T_{min}.

Input

The first line of input file contains two integers R and C — number of rows and number of columns of the labyrinth ($1 \le R, C \le 10^3$). Folloing R lines describes the labyrinth. Each line contains exactly C symbols. '`.`' denotes an empty cell, and '`#`' — a rock-filled cell.

There is at least one empty cell. All the cells around the provided map are wall.

Output

Output the mean time of running in seconds. Mean absolute and relative errors should not exceed 10^{-9}.

Examples

input.txt	output.txt
3 4 #### #.## ####	166.6666666666666666
3 3 #.# ... #..	305.5555555555555555

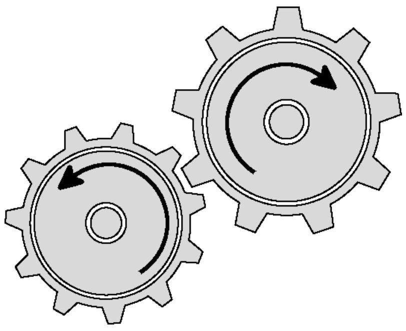
Problem 2. Gears

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes
Feedback:	

An engineer received the task to repair broken experimental device. After opening the device, he found a mechanism, made of interconnected gears. According to the user manual, the device is driven by two laboratory assistants, who should rotate The Main Wheel clockwise (The Main Wheel is also a gear).

Quick review of the device showed that there are no cycles. Any pair of gears are connected by not more than one pathway of immediate connections.

To fix the broken device, the engineer needs to know the rotational direction of terminal gears. A gear is terminal if it is adjacent to not more than one other gear.



Input

The first line of the input file contains two integers N and G, separated by single space. N is the number of gears and G is the index of The Main Wheel ($1 \le N \le 10^5$, $1 \le G \le N$). All the indices are run from 1 to N.

The following N lines describe how gears are adjacent to each other.

Gears are listed in the order of its' indices. The very first number in the line is g_i – the number of adjacent gears for a i-th gear ($1 \le i \le N, 0 \le g_i < N$). Next g_i numbers in the line is the indices of adjacent gears.

Output

The output file should list rotational directions of terminal wheel in the order of indices. Clockwise direction should be denoted with word "`CW`", counterclockwise with "`CCW`". If the gear is not moving, it shall be denoted with single character '`-`' (ASCII 45).

Example

input.txt	output.txt

6 2
3 6 4 5
1 5
0
1 1
2 2 1
1 1
	CW
	-
	CCW
	CCW

Problem 3. Dirtree

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes
Feedback:	

Any filesystem's folder with its contents could be presented in a tree form. The root of that tree would represent the forementioned folder. The files would be presented as leaves, and the inner nodes of that tree would denote the subfolders. For a folder, its immediate content would be presented as children nodes and/or leaves.

One long-forgotten OS used a convenient presentation of tree structure: A tree is presented as a sequence of lines, each line made of a folder/file name, prefixed with a few indentation chars "`| `" (ASCII: 124, 32) and a flag. The number of indentation chars denote a level of folder nesting. The flag is either '`-`' (ASCII: 45) or '`.`' (ASCII: 46). '`-`' denotes folder and '`.`' denotes a file.

The first line contains the name of the root folder, no indentaion, folder flag is present. The following lines contain all the folder contents, with its indentation and flags. Subfolders and files, contained in a folder are unordered, except that all folders go ahead of all files.

A folder may not be empty. The names of all the files and folders are unique.

Though the OS is long-forgotten, it has some use, so you are to add some functionality to it.

You are to implement a printer-friendly presentation of the dirtree. However, there is a catch. One page may contain no more than W lines. Since full tree could easily make more than W lines, you need to split the tree into pages, while trying to minimize the number of pages.

To fit a dirtree onto one page, you may now «close» some subfolders. If a subfolder is closed, its contents is not printed immediately. Closed folders denoted by a new flag '`+`' (ASCII: 43).

Since closed folders may contain some important stuff, you have to put them on the separate pages.

On that extra page the now-opened folder is presented as a root folder with all the contents. Some of the subfolders could again be «closed», though.

Each page may contain only one root folder. You may not put contents of two closed folders on the same page, even if those together take less than W lines. If folder's contents could not fit into W lines, such presentation is impossible.

Your task is to minimize the number of pages and output the printer-friendly presentation

Input

The first line of the input file contains one integer W — number of lines per output page.

The remaining lines represent dirtree.

The names of files and folders contain only lowercase letters of latin alphabet. The length of names is in bounds between 1 and 50.

Total number of lines in the presentation of root folder is between 2 and 10 000. Total number of symbols in the entire presentation would not exceed 10^6, not counting newline symbols.

Output

If it is impossible to print out the entire tree, output just one word: "`IMPOSSIBLE`".

Otherwise, there should one integer number K — the minimal number of pages. On the following lines, the contents of all K pages should be put. Every page should be finished with 20 '`=`' symbols (ASCII: 61).

If there are more than one optimal solution, output one of them. The tree on every page should be printed according to forementioned rules. Do not put extra spaces or newlines — this may be considered as an wrong output.

Examples

input.txt

```
5
-root
| -data
| | -documents
| | | -docs
| | | | .readme
| | | | .reference
| | -games
| | | -adom
| | | | .adomexe
| | | .sudoku
| | | .tetris
| | -programs
| | | -cpp
| | | | .helloworld
| | | .testlib
| .trash
| .deleteme
```

output.txt

```
5
-root
| +data
| .trash
| .deleteme
====================
-documents
| -docs
| | .readme
| | .reference
====================
-data
| +documents
| +games
| +programs
====================
-games
| -adom
| | .adomexe
| .tetris
| .sudoku
====================
-programs
| -cpp
| | .helloworld
| .testlib
====================
```

Problem 4. Crafty Isomorphism

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
Memory limit:	256 megabytes
Feedback:	

In this year Vasya went up to the first year of Mechanics and Mathematics Department. During first days of the studying he understood the severity of his choice. Perhaps, the most complicated definitions, which he found out in this autumn, are algebraic systems and their isomorphisms. Tonight Vasya as always sits at the studing room and tried again to sort out with these complicated mathematical definitions.

Using lecture notes he tried to write down the definitions in the special case when an algebraic system has exactly one unary operation and, that's what he has obtained:

- An unary operation on a set A is defined as arbitrary function acting from A to the set A itself.
- Algebraic system with exactly one unary operation is defined as a pair $< A, * >$, where A is some set and $\star$ is an unary operation on A.
- Let there are two algebraic systems $M =< A, * >$ and $N =< B, \circ >$ with unary operations $*$ and $\circ$, moreover, A and B have equal cardinalities. Then a bijection map ϕ from A to B is called an isomorphism of algebraic systems M and N, if $\phi(*(a)) = \circ(\phi(a))$ for every $a \in A$.

For better understanding how such isomorphism is organized, Vasya decided to write down all different algebraic systems of the form $< A, * >$, where A is an n-element set, $*$ is an unary operation on A, writing only those algebraic systems, which are not isomorphic to those, he already has written down. Help to Vasya to count total number of such algebraic systems up to isomorphsim, because today he have to sort out with the Turing machines too...

Input

The first line of the input file contains integer n and prime p separated by a singe space ($1 \le n \le 50$, $53 \le p \le 10^9$).

Output

The output file must contain one integer which is equal to the number of algebraic systems up to isomorphism made upon of an n-element set with one unary operation modulo p.

Examples

input.txt	output.txt
2 53	3
3 83	7

Problem 5. Alien Socks

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes
Feedback:	

A wife of an alien has washed his socks. Now you are to help her to find out if there is a "pair" for every sock.

Input

The first line of the input file contains two integers N and L, separated by a single space. N is the number of the socks and L is the number of the alien's legs. ($1 \le N \le 10^6, 1 \le L \le 10^4$).

The following N lines describe the socks. Each line contains one positive integer denoting the type of the corresponding sock.The integers are less or equal to 10^5.

Output

The output file must contain the word "`OK`" if all the socks can be divided into the groups of size L, so that each group would only contain socks of the same type.

Otherwise, the output file must contain the word "`ARGH!!1`".

Examples

input.txt	output.txt

4 2
1
2
1
2
	OK

6 3
10
15
7
15
10
15
	ARGH!!1

Problem 6. Brackets

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 megabytes
Feedback:	

As you may know, the very fundamental part of LISP-like languages are brackets. Round, square, and curly — lots of brackets. Sure, the are some other symbols, but most of all symbols are brackets.

In the dungeons of Noble Smartasses University the new programming language was invented — Uncommon LISP. It took the very essence of all the functional languages. It has no symbols other than brackets.

The first version (v1) of Uncommon LISP defines that the syntactically correct program would be a correct sequence of brackets of four types: "`()`", "`{}`", "`[]`" and "`<>`".

For the sequence to be correct, the following rules must be complied:

- empty sequence is a correct sequence;
- if we begin with correct sequence, and put an opening bracket of a type before the sequence, and the closing bracket of the same type after the sequence, this will make a correct sequence;
- two correct sequences concatenated make a correct sequence.

Unluckily, the abovementioned rules were too complicated, so in the next generation of the Uncommon LISP (v2) the definition of correct sequence has been changed. v2 language standard defines what you could make a correct v2-sequence out of v1-sequence by repeatedly switching pairs of adjacent brackets if both of them are either closing or opening. For a given sequence you need to determine, if it is a correct v2-sequence.

Input

The first line of input file contains single integer number N — the number of tests in the following lines. Each of the following lines contains one test sequence.

The total length of all sequences in one test file will not exceed 1 000 000.

Output

For each sequence in the input file, output one line with the single letter "`T`" if the sequence is a valid v2-sequence, or "`NIL`" if it is not.

Examples

input.txt	output.txt
2 ([)])([]	T NIL

Problem 7. James Bond

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes
Feedback:	

Agent 007 likes to play games with his enemies. And they have reasons to suspect so. Some time ago an enemy agency stole few encrypted letters from James Bond.

Trying to decipher these letters Agency staff has assumed the following:

James Bond encrypts his letters in quite straightforward way: for every symbol c from the letter he substitutes it with a codeword A_c.

Every symbol has exactly one corresponding codeword. Each codeword contains only latin letters (uppercase and lowercase) and numbers. All codewords are different.

The agency was able to get the table of symbols with respective codewords. However, there is still a chance what the given table could produce more than one text for some encrypted message. It could be that James Bond uses this ambiguity to trick the agency.

The agency hired you to determine if this table has such ambiguity. In other words, for a given table, could you construct a pair of texts (C and D), which, upon ciphering would produce similar code. $C = c_1, c_2, \ldots, c_t$ and $D = d_1, d_2, \ldots, d_s$, that $A_{c_1} A_{c_2} \ldots A_{c_t} = A_{d_1} A_{d_2} \ldots A_{d_s}$.

Input

The first line contains positive integer number $n \leq 100$ — number of symbols. Each of next n lines contains one codeword for each of n symbols.

Total length of all codewords will not exceed $5 \cdot 10^4$.

Output

You must output "`YES`", if the provided codewords can produce ambiguous messages, or "`NO`" otherwise.

Examples

input.txt	output.txt
4 a aB B5 5	YES
3 aB c abc	NO

Problem 8. Sorting the photos

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes
Feedback:	

One day Sergey decided to clean up his room. First, he looked under his couch. There he discovered a box full of old photos. Some of the photos were black-and-white, some were colored. Sergey took out all the photos out of the box and put them onto the table, splitting them to several stacks.

Then he decided to to put the photos back to the box rearranging them, so what the top part of the stack would be of colored photos, and the bottom would be of black-and-whites.

Sergey was well aware of his sorting urges, so he decided to spend as less time as possible. Help him to sort the photos using minimal number of moves.

For one move Sergey can pick up few photos from the top of one stack and put them on top of another stack, no flipping or rearranging allowed. Following rules should be obeyed:

- you shall not make new stacks;
- if you have a stack run out empty, you con continue using its place for stacking;
- intially, the box is empty;
- photos shall not be taken out of the box;
- the photos, taken from the source stack, shall not change its' order in the target stack.

Input

The first line of input file contains one integer N — the number of stacks ($1 \le N \le 1000$). In the following N lines stacks are described (upside down). Each line begins with integer n_i — total number of photos in the i-th stack ($1 \le i \le N$, $1 \le n_i \le 1000$). Each photo denoted as either "`bw`", for black-and-whites, or "`c`", for colored.

Output

The output file must contain one integer number — the minimal number of moves needed to rearrange the photos in the forementioned way.

If it is impossible to rearrange the photos, just one number −1 (minus one) must be put in the output file.

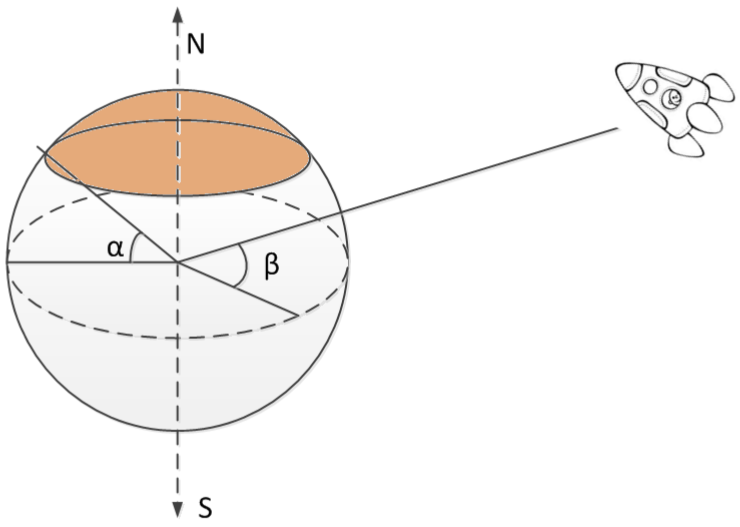
Examples

input.txt	output.txt
2 2 bw bw 1 c	2
3 1 c 2 bw c 1 c	4

Problem 9. Deep Field

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes
Feedback:	

A starship crew have discovered a very distant and interesting planet. A big round spot is present at the north pole of this planet. The spot fills all the surface from north pole to latitude α. The starship positioned somewhere over latitude β. The distance between the starship and the planet is so big, we would assume it is infinite.



You are to help astronauts to calculate which portion of the area of the visible disk the spot takes.

Input

The only line contains two integer numbers α and β ($0 \le \alpha \le 89, -90 \le \beta \le 90$). The lattitude is measured in degres, where -90 in south pole and 90 is north pole.

Output

Output file should contain one real number — the portion of visible disk, taken by the spot. The relative mean error should not exceed 10^{-8}.

Example

input.txt	output.txt
`60 0`	`0.02883444`

Problem 10. Lords of Hazelbright

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 megabytes
Feedback:	

The lords of Hazelbright have started a civil war, again! The war is going so serious, even The Emperor is considering an involvement. A Noble Lord Forwin decided to earn Emperor's trust by showing how Forwin aspires to peace and progress. To show that, Forwin wants to set up the farming in the middle of the province's capital.

The capital was build with a convenience of defense in mind. It consists mostly of towers and walls. So, there is n watchtowers, which are marked as dots on the map. Some pairs of watchtowers are connected by straight wall, the map shows them as segments. Walls never intersect. However, some watchtowers may be adjacent to more than one wall. A watchtower could not be located in the middle of a wall, so there always exactly two watchtowers on one wall.

The farmers need a protected territory inside of the city. To be protected, territory needs to be completely surrounded by wall. The wall should be a non-degenerate polygon without self-intersections. Each tower may not be included more than once.

Unfortunately, Forwin's enemies are not considering to stop the war. So, walls should be guarded by soldiers. Soldiers should be put on every selected wall and every selected watchtower. Forwin made explicit markings on the map, denoting, for every watchtower and evey wall, how many soldiers should be there to guard.

For the sake of peaceful look, Forwin wants to minimize density of military presence. Help Forwin to select a polygon with minimal ratio of soldiers to guarded area.

Input

The first line of the input file contains integer numbers n and m, where n — count of watchtowers, m — count of walls ($2 \leq n \leq 500$).

The following n lines describe watchtowers, one per line. Each tower is decribed with three integer numbers x_i, y_i, and c_i, where x_i, y_i – coordinates of the tower, and c_i — the number of soldiers, needed to guard this tower. ($|x_i|, |y_i| \leq 500$, $1 \leq c_i \leq 500$). Each tower also get an index assigned, from 1 in n, in the order of the lines.

Next m lines contain description of walls, one per line, again. Each wall is described by three integers a_j, b_j, and d_j, where a_j, b_j — indices of towers to which the wall is adjacent ,and d_j — number of soldiers, needed to guard to wall. ($1 \leq a_j < b_j \leq n$, $0 \leq d_j \leq 500$).

Output

For the selected polygon, output the number of watchtowers in the first line of output file. The second line should contain the indices of watchtowers in the counterclockwise order.

In the unluckily case, when it is impossible to select a polygon, you should output just one line: "`Barrayar in danger`".

Examples

input.txt	output.txt

10 13
2 2 5
5 1 3
7 3 6
4 3 7
1 5 2
5 5 5
7 8 50
3 7 3
8 1 1
3 3 50
1 2 3
2 3 7
2 4 20
3 9 3
3 4 5
1 4 27
4 6 0
4 8 3
6 8 0
3 7 7
1 5 2
5 8 3
7 8 1
	7
	1 2 3 4 6 8 5

5 3
2 4 1
1 1 1
4 1 1
3 3 1
4 3 1
1 2 0
2 3 0
4 5 0
	Barrayar in danger

Note

In the first example, the optimal area is 17, for which defense we need 51 soldiers. $51/17 = 3$.

The biggest area (polygon “`1 2 3 7 8 5`”) gives as ratio $92/30 = 3\frac{1}{15}$.

The area “`4 3 7 8 6`” gives $84/13 = 6\frac{6}{13}$.