CODEFORCES β
Sponsored by Telegram

Codeforces Round #454 (Div. 2, based on Technocup 2018 Elimination Round 4)

A. Masha and Bears

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A family consisting of father bear, mother bear and son bear owns three cars. Father bear can climb into the largest car and he likes it. Also, mother bear can climb into the middle car and she likes it. Moreover, son bear can climb into the smallest car and he likes it. It's known that the largest car is strictly larger than the middle car, and the middle car is strictly larger than the smallest car.

Masha came to test these cars. She could climb into all cars, but she liked only the smallest car.

It's known that a character with size a can climb into some car with size b if and only if $a \leq b$, he or she likes it if and only if he can climb into this car and $2a \geq b$.

You are given sizes of bears and Masha. Find out some possible integer non-negative sizes of cars.

Input

You are given four integers V_1, V_2, V_3, $V_m (1 \leq V_i \leq 100)$ — sizes of father bear, mother bear, son bear and Masha, respectively. It's guaranteed that $V_1 > V_2 > V_3$.

Output

Output three integers — sizes of father bear's car, mother bear's car and son bear's car, respectively.

If there are multiple possible solutions, print any.

If there is no solution, print "`-1`" (without quotes).

Examples

input

```
50 30 10 10
```

output

```
50
30
10
```

input

```
100 50 10 21
```

output

```
-1
```

Note

In first test case all conditions for cars' sizes are satisfied.

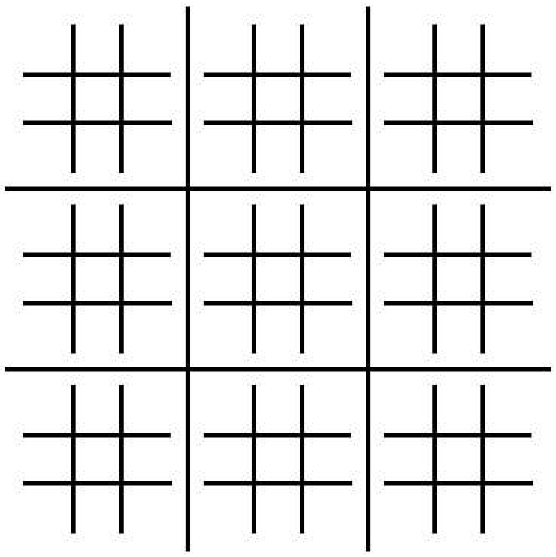
In second test case there is no answer, because Masha should be able to climb into smallest car (so size of smallest car in not less than 21), but son bear should like it, so maximum possible size of it is 20.

B. Tic-Tac-Toe

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Two bears are playing tic-tac-toe via mail. It's boring for them to play usual tic-tac-toe game, so they are a playing modified version of this game. Here are its rules.

The game is played on the following field.



Players are making moves by turns. At first move a player can put his chip in any cell of any small field. For following moves, there are some restrictions: if during last move the opposite player put his chip to cell with coordinates (x_l, y_l) in some small field, the next move should be done in one of the cells of the small field with coordinates (x_l, y_l). For example, if in the first move a player puts his chip to lower left cell of central field, then the second player on his next move should put his chip into some cell of lower left field (pay attention to the first test case). If there are no free cells in the required field, the player can put his chip to any empty cell on any field.

You are given current state of the game and coordinates of cell in which the last move was done. You should find all cells in which the current player can put his chip.

A hare works as a postman in the forest, he likes to foul bears. Sometimes he changes the game field a bit, so the current state of the game could be unreachable. However, after his changes the cell where the last move was done is not empty. You don't need to find if the state is unreachable or not, just output possible next moves according to the rules.

Input

First 11 lines contains descriptions of table with 9 rows and 9 columns which are divided into 9 small fields by spaces and empty lines. Each small field is described by 9 characters without spaces and empty lines. character "x" (ASCII-code 120) means that the cell is occupied with chip of the first player, character "o" (ASCII-code 111) denotes a field occupied with chip of the second player, character "." (ASCII-code 46) describes empty cell.

The line after the table contains two integers x and y $(1 \le x, y \le 9)$. They describe coordinates of the cell in table where the last move was done. Rows in the table are numbered from up to down and columns are numbered from left to right.

It's guaranteed that cell where the last move was done is filled with "x" or "o". Also, it's guaranteed that there is at least one empty cell. It's not guaranteed that current state of game is reachable.

Output

Output the field in same format with characters "!" (ASCII-code 33) on positions where the current player can put his chip. All other cells should not be modified.

Examples

input

```
... ... ...
... ... ...
... ... ...

... ... ...
... ... ...
... x.. ...

... ... ...
... ... ...
... ... ...
6 4
```

output

```
... ... ...
... ... ...
... ... ...

... ... ...
... ... ...
... x.. ...

!!! ... ...
!!! ... ...
!!! ... ...
```

input Copy

```
xoo x.. x..
ooo ... ...
ooo ... ...

x.. x.. x..
... ... ...
... ... ...

x.. x.. x..
... ... ...
... ... ...
7 4
```

output

```
xoo x!! x!!
ooo !!! !!!
ooo !!! !!!

x!! x!! x!!
!!! !!! !!!
!!! !!! !!!

x!! x!! x!!
!!! !!! !!!
!!! !!! !!!
```

input Copy

```
o.. ... ...
... ... ...
... ... ...

... xxx ...
... xox ...
... ooo ...

... ... ...
... ... ...
... ... ...
5 5
```

output

```
o!! !!! !!!
!!! !!! !!!
!!! !!! !!!

!!! xxx !!!
!!! xox !!!
!!! ooo !!!

!!! !!! !!!
!!! !!! !!!
!!! !!! !!!
```

Note

In the first test case the first player made a move to lower left cell of central field, so the second player can put a chip only to cells of lower left field.

In the second test case the last move was done to upper left cell of lower central field, however all cells in upper left field are occupied, so the second player can put his chip to any empty cell.

In the third test case the last move was done to central cell of central field, so current player can put his chip to any cell of central field, which is already occupied, so he can move anywhere. Pay attention that this state of the game is unreachable.

C. Shockers

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Valentin participates in a show called "Shockers". The rules are quite easy: jury selects one letter which Valentin doesn't know. He should make a small speech, but every time he pronounces a word that contains the selected letter, he receives an electric shock. He can make guesses which letter is selected, but for each incorrect guess he receives an electric shock too. The show ends when Valentin guesses the selected letter correctly.

Valentin can't keep in mind everything, so he could guess the selected letter much later than it can be uniquely determined and get excessive electric shocks. Excessive electric shocks are those which Valentin got after the moment the selected letter can be uniquely determined. You should find out the number of excessive electric shocks.

Input

The first line contains a single integer n $(1 \leq n \leq 10^5)$ — the number of actions Valentin did.

The next n lines contain descriptions of his actions, each line contains description of one action. Each action can be of one of three types:

1. Valentin pronounced some word and didn't get an electric shock. This action is described by the string "`. w`" (without quotes), in which "`.`" is a dot (ASCII-code 46), and w is the word that Valentin said.
2. Valentin pronounced some word and got an electric shock. This action is described by the string "`! w`" (without quotes), in which "`!`" is an exclamation mark (ASCII-code 33), and w is the word that Valentin said.
3. Valentin made a guess about the selected letter. This action is described by the string "`? s`" (without quotes), in which "`?`" is a question mark (ASCII-code 63), and s is the guess — a lowercase English letter.

All words consist only of lowercase English letters. The total length of all words does not exceed 10^5.

It is guaranteed that last action is a guess about the selected letter. Also, it is guaranteed that Valentin didn't make correct guesses about the selected letter before the last action. Moreover, it's guaranteed that if Valentin got an electric shock after pronouncing some word, then it contains the selected letter; and also if Valentin didn't get an electric shock after pronouncing some word, then it does not contain the selected letter.

Output

Output a single integer — the number of electric shocks that Valentin could have avoided if he had told the selected letter just after it became uniquely determined.

Examples

input Copy
```
5
! abc
. ad
. b
! cd
? c
```
output
```
1
```

input Copy
```
8
! hello
! codeforces
? c
. o
? d
? h
. l
? e
```
output
```
2
```

input Copy
```
7
! ababahalamaha
? a
? b
? a
? b
? a
? h
```
output
```
0
```

Note

In the first test case after the first action it becomes clear that the selected letter is one of the following: a, b, c. After the second action we can note that the selected letter is not a. Valentin tells word "b" and doesn't get a shock. After that it is clear that the selected letter is c, but Valentin pronounces the word cd and gets an excessive electric shock.

In the second test case after the first two electric shocks we understand that the selected letter is e or o. Valentin tries some words consisting of these letters and after the second word it's clear that the selected letter is e, but Valentin makes 3 more actions before he makes a correct hypothesis.

In the third example the selected letter can be uniquely determined only when Valentin guesses it, so he didn't get excessive electric shocks.

D. Seating of Students

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Students went into a class to write a test and sat in some way. The teacher thought: "Probably they sat in this order to copy works of each other. I need to rearrange them in such a way that students that were neighbors are not neighbors in a new seating."

The class can be represented as a matrix with n rows and m columns with a student in each cell. Two students are neighbors if cells in which they sit have a common side.

Let's enumerate students from 1 to $n \cdot m$ in order of rows. So a student who initially sits in the cell in row i and column j has a number $(i - 1) \cdot m + j$. You have to find a matrix with n rows and m columns in which all numbers from 1 to $n \cdot m$ appear exactly once and adjacent numbers in the original matrix are not adjacent in it, or determine that there is no such matrix.

Input

The only line contains two integers n and m $(1 \le n, m \le 10^5; n \cdot m \le 10^5)$ — the number of rows and the number of columns in the required matrix.

Output

If there is no such matrix, output "NO" (without quotes).

Otherwise in the first line output "YES" (without quotes), and in the next n lines output m integers which form the required matrix.

Examples

input Copy
```
2 4
```
output
```
YES
5 4 7 2
3 6 1 8
```

input Copy
```
2 1
```
output
```
NO
```

Note

In the first test case the matrix initially looks like this:

```
1 2 3 4
5 6 7 8
```

It's easy to see that there are no two students that are adjacent in both matrices.

In the second test case there are only two possible seatings and in both of them students with numbers 1 and 2 are neighbors.

E. Party

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Arseny likes to organize parties and invite people to it. However, not only friends come to his parties, but friends of his friends, friends of friends of his friends and so on. That's why some of Arseny's guests can be unknown to him. He decided to fix this issue using the following procedure.

At each step he selects one of his guests A, who pairwise introduces all of his friends to each other. After this action any two friends of A become friends. This process is run until all pairs of guests are friends.

Arseny doesn't want to spend much time doing it, so he wants to finish this process using the minimum number of steps. Help Arseny to do it.

Input

The first line contains two integers n and m ($1 \le n \le 22$; $0 \le m \le \frac{n \cdot (n-1)}{2}$) — the number of guests at the party (including Arseny) and the number of pairs of people which are friends.

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$; $u \ne v$), which means that people with numbers u and v are friends initially. It's guaranteed that each pair of friends is described not more than once and the graph of friendship is connected.

Output

In the first line print the minimum number of steps required to make all pairs of guests friends.

In the second line print the ids of guests, who are selected at each step.

If there are multiple solutions, you can output any of them.

Examples

input

```
5 6
1 2
1 3
2 3
2 5
3 4
4 5
```

output

```
2
2 3
```

input

```
4 4
1 2
1 3
1 4
3 4
```

output

```
1
1
```

Note

In the first test case there is no guest who is friend of all other guests, so at least two steps are required to perform the task. After second guest pairwise introduces all his friends, only pairs of guests $(4, 1)$ and $(4, 2)$ are not friends. Guest 3 or 5 can introduce them.

In the second test case guest number 1 is a friend of all guests, so he can pairwise introduce all guests in one step.

F. Power Tower

time limit per test: 4.5 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Priests of the Quetzalcoatl cult want to build a tower to represent a power of their god. Tower is usually made of power-charged rocks. It is built with the help of rare magic by levitating the current top of tower and adding rocks at its bottom. If top, which is built from $k - 1$ rocks, possesses power p and we want to add the rock charged with power w_k then value of power of a new tower will be $\{w_k\}^p$.

Rocks are added from the last to the first. That is for sequence $w_1, ..., w_m$ value of power will be

$$w_1^{\left(w_2^{\left(w_3^{\cdots^{w_m}}\right)}\right)}$$

After tower is built, its power may be extremely large. But still priests want to get some information about it, namely they want to know a number called cumulative power which is the true value of power taken modulo m. Priests have n rocks numbered from 1 to n. They ask you to calculate which value of cumulative power will the tower possess if they will build it from rocks numbered $l, l + 1, ..., r$.

Input

First line of input contains two integers n $(1 \le n \le 10^5)$ and m $(1 \le m \le 10^9)$.

Second line of input contains n integers w_k $(1 \le w_k \le 10^9)$ which is the power of rocks that priests have.

Third line of input contains single integer q $(1 \le q \le 10^5)$ which is amount of queries from priests to you.

k^{th} of next q lines contains two integers l_k and r_k $(1 \le l_k \le r_k \le n)$.

Output

Output q integers. k-th of them must be the amount of cumulative power the tower will have if is built from rocks $l_k, l_k + 1, ..., r_k$.

Example

input

```
6 1000000000
1 2 2 3 3 3
8
1 1
1 6
2 2
2 3
2 4
4 4
4 5
4 6
```

output

```
1
1
2
4
256
3
27
597484987
```

Note

$3^{27} = 7625597484987$