Codeforces Round #504 (rated, Div. 1 + Div. 2, based on VK Cup 2018 Final)

A. Single Wildcard Pattern Matching

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two strings s and t. The string s consists of lowercase Latin letters and **at most one** wildcard character '*', the string t consists only of lowercase Latin letters. The length of the string s equals n, the length of the string t equals m.

The wildcard character '*' in the string s (if any) can be replaced with an arbitrary sequence (possibly empty) of lowercase Latin letters. No other character of s can be replaced with anything. If it is possible to replace a wildcard character '*' in s to obtain a string t, then the string t matches the pattern s.

For example, if s ="aba*aba" then the following strings match it "abaaba", "abacaba" and "abazzzaba", but the following strings do not match: "ababa", "abcaaba", "codeforces", "aba1aba", "aba?aba".

If the given string t matches the given string s, print "YES", otherwise print "NO".

Input

The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the length of the string s and the length of the string t, respectively.

The second line contains string s of length n, which consists of lowercase Latin letters and **at most one** wildcard character '*'.

The third line contains string t of length m, which consists only of lowercase Latin letters.

Output

Print "YES" (without quotes), if you can obtain the string t from the string s. Otherwise print "NO" (without quotes).

Examples

input
6 10 code*s codeforces
output
YES

input
6 5 vk*cup vkcup
output
YES

input
1 1 v k
output
NO

input
9 6 gfgf*gfgf gfgfgf
output
NO

Note

In the first example a wildcard character '*' can be replaced with a string "force". So the string s after this replacement is "codeforces" and the answer is "YES".

In the second example a wildcard character '*' can be replaced with an empty string. So the string s after this replacement is "vkcup" and the answer is "YES".

There is no wildcard character '*' in the third example and the strings "v" and "k" are different so the answer is "NO".

In the fourth example there is no such replacement of a wildcard character '*' that you can obtain the string t so the answer is "NO".

B. Pair of Toys

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Tanechka is shopping in the toy shop. There are exactly n toys in the shop for sale, the cost of the i-th toy is i burles. She wants to choose two toys in such a way that their total cost is k burles. How many ways to do that does she have?

Each toy appears in the shop exactly once. Pairs (a, b) and (b, a) are considered equal. Pairs (a, b), where $a = b$, are not allowed.

Input

The first line of the input contains two integers n, k ($1 \le n, k \le 10^{14}$) — the number of toys and the expected total cost of the pair of toys.

Output

Print the number of ways to choose the pair of toys satisfying the condition above. Print `0`, if Tanechka can choose no pair of toys in such a way that their total cost is k burles.

Examples

input
8 5
output
2

input
8 15
output
1

input
7 20
output
0

input
1000000000000 1000000000001
output
500000000000

Note

In the first example Tanechka can choose the pair of toys $(1, 4)$ or the pair of toys $(2, 3)$.

In the second example Tanechka can choose only the pair of toys $(7, 8)$.

In the third example choosing any pair of toys will lead to the total cost less than 20. So the answer is `0`.

In the fourth example she can choose the following pairs: $(1, 1000000000000)$, $(2, 999999999999)$, $(3, 999999999998)$, ..., $(500000000000, 500000000001)$. The number of such pairs is exactly 500000000000.

C. Bracket Subsequence

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

A bracket sequence is a string containing only characters "(" and ")". A regular bracket sequence is a bracket sequence that can be transformed into a correct arithmetic expression by inserting characters "1" and "+" between the original characters of the sequence. For example, bracket sequences "()()" and "(())" are regular (the resulting expressions are: "(1)+(1)" and "((1+1)+1)"), and ")(", "(" and ")" are not.

Subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

You are given a regular bracket sequence s and an integer number k. Your task is to find a regular bracket sequence of length

exactly k such that it is also a subsequence of s.

It is guaranteed that such sequence always exists.

Input
The first line contains two integers n and k ($2 \le k \le n \le 2 \cdot 10^5$, both n and k are even) — the length of s and the length of the sequence you are asked to find.

The second line is a string s — regular bracket sequence of length n.

Output
Print a single string — a regular bracket sequence of length exactly k such that it is also a subsequence of s.

It is guaranteed that such sequence always exists.

Examples

input
6 4 ()(())
output
()()

input
8 8 (()(()))
output
(()(()))

D. Array Restoration

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Initially there was an array a consisting of n integers. Positions in it are numbered from 1 to n.

Exactly q queries were performed on the array. During the i-th query some segment (l_i, r_i) $(1 \le l_i \le r_i \le n)$ was selected and values of elements on positions from l_i to r_i inclusive got changed to i. The order of the queries couldn't be changed and all q queries were applied. It is also known that every position from 1 to n got covered by at least one segment.

We could have offered you the problem about checking if some given array (consisting of n integers with values from 1 to q) can be obtained by the aforementioned queries. However, we decided that it will come too easy for you.

So the enhancement we introduced to it is the following. Some set of positions (possibly empty) in this array is selected and values of elements on these positions are set to 0.

Your task is to check if this array can be obtained by the aforementioned queries. Also if it can be obtained then restore this array.

If there are multiple possible arrays then print any of them.

Input
The first line contains two integers n and q ($1 \le n, q \le 2 \cdot 10^5$) — the number of elements of the array and the number of queries perfomed on it.

The second line contains n integer numbers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le q$) — the resulting array. If element at some position j is equal to 0 then the value of element at this position can be any integer from 1 to q.

Output
Print "YES" if the array a can be obtained by performing q queries. Segments (l_i, r_i) $(1 \le l_i \le r_i \le n)$ are chosen separately for each query. Every position from 1 to n should be covered by at least one segment.

Otherwise print "NO".

If some array can be obtained then print n integers on the second line — the i-th number should be equal to the i-th element of the resulting array and should have value from 1 to q. This array should be obtainable by performing exactly q queries.

If there are multiple possible arrays then print any of them.

Examples

input
4 3 1 0 2 3
output

YES
1 2 2 3

input
3 10 10 10 10
output
YES 10 10 10

input
5 6 6 5 6 2 2
output
NO

input
3 5 0 0 0
output
YES 5 4 2

Note

In the first example you can also replace 0 with 1 but not with 3.

In the second example it doesn't really matter what segments to choose until query 10 when the segment is $(1, 3)$.

The third example showcases the fact that the order of queries can't be changed, you can't firstly set $(1, 3)$ to 6 and after that change $(2, 2)$ to 5. The segment of 5 should be applied before segment of 6.

There is a lot of correct resulting arrays for the fourth example.

E. Down or Right

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem.

Bob lives in a square grid of size $n \times n$, with rows numbered 1 through n from top to bottom, and columns numbered 1 through n from left to right. Every cell is either allowed or blocked, but you don't know the exact description of the grid. You are given only an integer n.

Bob can move through allowed cells but only in some limited directions. When Bob is in an allowed cell in the grid, he can move **down or right** to an adjacent cell, if it is allowed.

You can ask at most $4 \cdot n$ queries of form "? r_1 c_1 r_2 c_2" ($1 \le r_1 \le r_2 \le n$, $1 \le c_1 \le c_2 \le n$). The answer will be "YES" if Bob can get from a cell (r_1, c_1) to a cell (r_2, c_2), and "NO" otherwise. In particular, if one of the two cells (or both) is a blocked cell then the answer is "NO" for sure. Since Bob doesn't like short trips, you can only ask queries with the manhattan distance between the two cells at least $n - 1$, i.e. the following condition must be satisfied: $(r_2 - r_1) + (c_2 - c_1) \ge n - 1$.

It's guaranteed that Bob can get from the top-left corner $(1, 1)$ to the bottom-right corner (n, n) and your task is to find a way to do it. You should print the answer in form "! S" where S is a string of length $2 \cdot n - 2$ consisting of characters 'D' and 'R', denoting moves down and right respectively. The down move increases the first coordinate by 1, the right move increases the second coordinate by 1. If there are multiple solutions, any of them will be accepted. You should terminate immediately after printing the solution.

Input

The only line of the input contains an integer n ($2 \le n \le 500$) — the size of the grid.

Output

When you are ready to print the answer, print a single line containing "! S" where where S is a string of length $2 \cdot n - 2$ consisting of characters 'D' and 'R', denoting moves down and right respectively. The path should be a valid path going from the cell $(1, 1)$ to the cell (n, n) passing only through allowed cells.

Interaction

You can ask at most $4 \cdot n$ queries. To ask a query, print a line containing "? r_1 c_1 r_2 c_2" ($1 \le r_1 \le r_2 \le n$, $1 \le c_1 \le c_2 \le n$). After that you should read a single line containing "YES" or "NO" depending on the answer of the query. "YES" means Bob can go from the cell (r_1, c_1) to the cell (r_2, c_2), "NO" means the opposite.

Note that the grid is fixed before the start of your solution and does not depend on your queries.

After printing a query do not forget to output end of line and flush the output. Otherwise you will get `Idleness limit exceeded` or other negative verdict. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

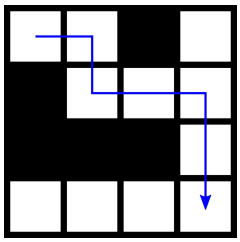
Answer "BAD" instead of "YES" or "NO" means that you made an invalid query. Exit immediately after receiving "BAD" and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

Example

input
4 YES NO YES YES
output
? 1 1 4 4 ? 1 2 4 3 ? 4 1 4 4 ? 1 4 4 4 ! RDRRDD

Note

The first example is shown on the picture below.



To hack, use the following input format:

The first line should contain a single integer n ($2 \le n \le 500$) — the size of the grid.

Each of the next n lines should contain a string of n characters '#' or '.', where '#' denotes a blocked cell, and '.' denotes an allowed cell.

For example, the following text encodes the example shown above:

```
4
..#.
#...
###.
....
```

F. Mobile Phone Network

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are managing a mobile phone network, and want to offer competitive prices to connect a network.

The network has n nodes.

Your competitor has already offered some connections between some nodes, with some fixed prices. These connections are

bidirectional. There are initially m connections the competitor is offering. The i-th connection your competitor is offering will connect nodes fa_i and fb_i and costs fw_i.

You have a list of k connections that you want to offer. It is guaranteed that this set of connection does not form any cycle. The j-th of these connections will connect nodes ga_j and gb_j. These connections are also bidirectional. The cost of these connections have not been decided yet.

You can set the prices of these connections to any arbitrary integer value. These prices are set independently for each connection. After setting the prices, the customer will choose such $n-1$ connections that all nodes are connected in a single network and the total cost of chosen connections is minimum possible. If there are multiple ways to choose such networks, the customer will choose an arbitrary one that also maximizes the number of your connections in it.

You want to set prices in such a way such that **all** your k connections are chosen by the customer, and the sum of prices of your connections is maximized.

Print the maximum profit you can achieve, or -1 if it is unbounded.

Input

The first line of input will contain three integers n, k and m ($1 \leq n, k, m \leq 5 \cdot 10^5, k \leq n-1$), the number of nodes, the number of your connections, and the number of competitor connections, respectively.

The next k lines contain two integers ga_i and gb_i ($1 \leq ga_i, gb_i \leq n$, $ga_i \neq gb_i$), representing one of your connections between nodes ga_i and gb_i. Your set of connections is guaranteed to be acyclic.

The next m lines contain three integers each, fa_i, fb_i and fw_i ($1 \leq fa_i, fb_i \leq n$, $fa_i \neq fb_i$, $1 \leq fw_i \leq 10^9$), denoting one of your competitor's connections between nodes fa_i and fb_i with cost fw_i. None of these connections connects a node to itself, and no pair of these connections connect the same pair of nodes. In addition, these connections are given by non-decreasing order of cost (that is, $fw_{i-1} \leq fw_i$ for all valid i).

Note that there may be some connections that appear in both your set and your competitor's set (though no connection will appear twice in one of this sets).

It is guaranteed that the union of all of your connections and your competitor's connections form a connected network.

Output

Print a single integer, the maximum possible profit you can achieve if you set the prices on your connections appropriately. If the profit is unbounded, print -1.

Examples

input
4 3 6 1 2 3 4 1 3 2 3 3 3 1 4 1 2 4 4 2 8 4 3 8 4 1 10
output
14

input
3 2 1 1 2 2 3 1 2 30
output
-1

input
4 3 3 1 2 1 3 1 4 4 1 1000000000 4 2 1000000000 4 3 1000000000
output
3000000000

Note

In the first sample, it's optimal to give connection $1-3$ cost 3, connection $1-2$ cost 3, and connection $3-4$ cost 8. In this case, the cheapest connected network has cost 14, and the customer will choose one that chooses all of your connections.

In the second sample, as long as your first connection costs 30 or less, the customer chooses both your connections no matter what is the cost of the second connection, so you can get unbounded profit in this case.

G. Pisces

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A group of researchers are studying fish population in a natural system of lakes and rivers. The system contains n lakes connected by $n-1$ rivers. Each river has integer length (in kilometers) and can be traversed in both directions. It is possible to travel between any pair of lakes by traversing the rivers (that is, the network of lakes and rivers form a tree).

There is an unknown number of indistinguishable fish living in the lakes. On day 1, fish can be at arbitrary lakes. Fish can travel between lakes by swimming the rivers. Each fish can swim a river l kilometers long in any direction in l days. Further, each fish can stay any number of days in any particular lake it visits. No fish ever appear or disappear from the lake system. Each lake can accomodate any number of fish at any time.

The researchers made several observations. The j-th of these observations is "on day d_j there were at least f_j distinct fish in the lake p_j". Help the researchers in determining the smallest possible total number of fish living in the lake system that doesn't contradict the observations.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of lakes in the system.

The next $n-1$ lines describe the rivers. The i-th of these lines contains three integers u_i, v_i, l_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$, $1 \le l_i \le 10^3$) — 1-based indices of lakes connected by the i-th river, and the length of this river.

The next line contains a single integer k ($1 \le k \le 10^5$) — the number of observations.

The next k lines describe the observations. The j-th of these lines contains three integers d_j, f_j, p_j ($1 \le d_j \le 10^8$, $1 \le f_j \le 10^4$, $1 \le p_j \le n$) — the day, the number of fish, and the lake index of the j-th observation. No two observations happen on the same day at the same lake simultaneously.

Output

Print one integer — the smallest total number of fish not contradicting the observations.

Examples

input
4 1 2 1 1 3 1 1 4 1 5 1 1 2 1 1 3 2 2 1 3 1 4 3 1 2
output
2

input
5 1 2 1 1 3 1 1 4 1 1 5 1 4 1 1 2 2 1 3 3 1 4 4 1 5
output
2

input
5 2 5 1 5 1 1 2 4 1 5 3 3 6 5 2 4 2 1 1 2 1 3 2 2 4

4 7 5 4 1 2
output
10

Note

In the first example, there could be one fish swimming through lakes 2, 1, and 4, and the second fish swimming through lakes 3, 1, and 2.

In the second example a single fish can not possibly be part of all observations simultaneously, but two fish swimming $2 \to 1 \to 4$ and $3 \to 1 \to 5$ can.

In the third example one fish can move from lake 1 to lake 5, others staying in their lakes during all time: two fish in lake 4, six fish in lake 5, one fish in lake 3. The system of lakes is shown on the picture.