Manthan, Codefest 18 (rated, Div. 1 + Div. 2)

A. Packets

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

You have n coins, each of the same value of 1.

Distribute them into packets such that any amount x ($1 \le x \le n$) can be formed using some (possibly one or all) number of these packets.

Each packet may only be used entirely or not used at all. No packet may be used more than once in the formation of the single x, however it may be reused for the formation of other x's.

Find the minimum number of packets in such a distribution.

Input
The only line contains a single integer n ($1 \le n \le 10^9$) — the number of coins you have.

Output
Output a single integer — the minimum possible number of packets, satisfying the condition above.

Examples

input
6
output
3

input
2
output
2

Note
In the first example, three packets with 1, 2 and 3 coins can be made to get any amount x ($1 \le x \le 6$).

- To get 1 use the packet with 1 coin.
- To get 2 use the packet with 2 coins.
- To get 3 use the packet with 3 coins.
- To get 4 use packets with 1 and 3 coins.
- To get 5 use packets with 2 and 3 coins
- To get 6 use all packets.

In the second example, two packets with 1 and 1 coins can be made to get any amount x ($1 \le x \le 2$).

B. Reach Median

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

You are given an array a of n integers and an integer s. It is guaranteed that n is odd.

In one operation you can either increase or decrease any single element by one. Calculate the minimum number of operations required to make the median of the array being equal to s.

The median of the array with odd length is the value of the element which is located on the middle position after the array is sorted. For example, the median of the array $6, 5, 8$ is equal to 6, since if we sort this array we will get $5, 6, 8$, and 6 is located on the middle position.

Input
The first line contains two integers n and s ($1 \le n \le 2 \cdot 10^5 - 1$, $1 \le s \le 10^9$) — the length of the array and the required value of median.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array a.

It is guaranteed that n is odd.

Output
In a single line output the minimum number of operations to make the median being equal to s.

Examples

input
3 8 6 5 8
output
2

input
7 20 21 15 12 11 20 19 12
output
6

Note
In the first sample, 6 can be increased twice. The array will transform to $8, 5, 8$, which becomes $5, 8, 8$ after sorting, hence the median is equal to 8.

In the second sample, 19 can be increased once and 15 can be increased five times. The array will become equal to $21, 20, 12, 11, 20, 20, 12$. If we sort this array we get $11, 12, 12, 20, 20, 20, 21$, this way the median is 20.

C. Equalize

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

You are given two binary strings a and b of the same length. You can perform the following two operations on the string a:

- Swap any two bits at indices i and j respectively ($1 \le i, j \le n$), the cost of this operation is $|i - j|$, that is, the absolute difference between i and j.
- Select any arbitrary index i ($1 \le i \le n$) and flip (change 0 to 1 or 1 to 0) the bit at this index. The cost of this operation is 1.

Find the minimum cost to make the string a equal to b. It is not allowed to modify string b.

Input
The first line contains a single integer n ($1 \le n \le 10^6$) — the length of the strings a and b.

The second and third lines contain strings a and b respectively.

Both strings a and b have length n and contain only '0' and '1'.

Output
Output the minimum cost to make the string a equal to b.

Examples

input
3 100 001
output
2

input
4 0101 0011
output
1

Note
In the first example, one of the optimal solutions is to flip index 1 and index 3, the string a changes in the following way: "100" $\rightarrow$ "000" $\rightarrow$ "001". The cost is $1 + 1 = 2$.

The other optimal solution is to swap bits and indices 1 and 3, the string a changes then "100" $\rightarrow$ "001", the cost is also $|1 - 3| = 2$

.

In the second example, the optimal solution is to swap bits at indices 2 and 3, the string a changes as "0101" $\rightarrow$ "0011". The cost is $|2-3| = 1$.

D. Valid BFS?

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

The BFS algorithm is defined as follows.

1. Consider an undirected graph with vertices numbered from 1 to n. Initialize q as a new queue containing only vertex 1, mark the vertex 1 as used.
2. Extract a vertex v from the head of the queue q.
3. Print the index of vertex v.
4. Iterate in arbitrary order through all such vertices u that u is a neighbor of v and is not marked yet as used. Mark the vertex u as used and insert it into the tail of the queue q.
5. If the queue is not empty, continue from step 2.
6. Otherwise finish.

Since the order of choosing neighbors of each vertex can vary, it turns out that there may be multiple sequences which BFS can print.

In this problem you need to check whether a given sequence corresponds to some valid BFS traversal of the given tree **starting from vertex** 1. The tree is an undirected graph, such that there is exactly one simple path between any two vertices.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) which denotes the number of nodes in the tree.

The following $n-1$ lines describe the edges of the tree. Each of them contains two integers x and y ($1 \le x, y \le n$) — the endpoints of the corresponding edge of the tree. It is guaranteed that the given graph is a tree.

The last line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the sequence to check.

Output

Print "Yes" (quotes for clarity) if the sequence corresponds to some valid BFS traversal of the given tree and "No" (quotes for clarity) otherwise.

You can print each letter in any case (upper or lower).

Examples

input
4 1 2 1 3 2 4 1 2 3 4
output
Yes

input
4 1 2 1 3 2 4 1 2 4 3
output
No

Note

Both sample tests have the same tree in them.

In this tree, there are two valid BFS orderings:

- $1, 2, 3, 4,$
- $1, 3, 2, 4.$

The ordering $1, 2, 4, 3$ doesn't correspond to any valid BFS order.

E. Trips

time limit per test: 2 seconds

memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n persons who initially don't know each other. On each morning, two of them, who were not friends before, become friends.

We want to plan a trip for every evening of m days. On each trip, you have to select a group of people that will go on the trip. For every person, one of the following should hold:

- Either this person does not go on the trip,
- Or at least k of his friends also go on the trip.

Note that the friendship is not transitive. That is, if a and b are friends and b and c are friends, it does not necessarily imply that a and c are friends.

For each day, find the maximum number of people that can go on the trip on that day.

Input

The first line contains three integers n, m, and k ($2 \leq n \leq 2 \cdot 10^5, 1 \leq m \leq 2 \cdot 10^5$, $1 \leq k < n$) — the number of people, the number of days and the number of friends each person on the trip should have in the group.

The i-th ($1 \leq i \leq m$) of the next m lines contains two integers x and y ($1 \leq x, y \leq n$, $x \neq y$), meaning that persons x and y become friends on the morning of day i. It is guaranteed that x and y were not friends before.

Output

Print exactly m lines, where the i-th of them ($1 \leq i \leq m$) contains the maximum number of people that can go on the trip on the evening of the day i.

Examples

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

input
5 8 2 2 1 4 2 5 4 5 2 4 3 5 1 4 1 3 2
output
0 0 0 3 3 4 4 5

input
5 7 2 1 5 3 2 2 5 3 4 1 2 5 3 1 3
output
0 0 0 0 3 4 4

Note
In the first example,

- $1, 2, 3$ can go on day 3 and 4.

In the second example,

- $2, 4, 5$ can go on day 4 and 5.
- $1, 2, 4, 5$ can go on day 6 and 7.
- $1, 2, 3, 4, 5$ can go on day 8.

In the third example,

- $1, 2, 5$ can go on day 5.
- $1, 2, 3, 5$ can go on day 6 and 7.

F. Maximum Reduction

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

Given an array a of n integers and an integer k ($2 \le k \le n$), where each element of the array is denoted by a_i ($0 \le i < n$). Perform the operation z given below on a and print the value of $z(a, k)$ modulo $10^9 + 7$.

```
function z(array a, integer k):
    if length(a) < k:
        return 0
    else:
        b = empty array
        ans = 0
        for i = 0 .. (length(a) - k):
            temp = a[i]
            for j = i .. (i + k - 1):
                temp = max(temp, a[j])
            append temp to the end of b
            ans = ans + temp
        return ans + z(b, k)
```

Input
The first line of input contains two integers n and k ($2 \le k \le n \le 10^6$) — the length of the initial array a and the parameter k.

The second line of input contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($1 \le a_i \le 10^9$) — the elements of the array a.

Output
Output the only integer, the value of $z(a, k)$ modulo $10^9 + 7$.

Examples

input
3 2 9 1 10
output
29

input
5 3 5 8 7 1 9
output
34

Note
In the first example:

- for $a = (9, 1, 10)$, $ans = 19$ and $b = (9, 10)$,
- for $a = (9, 10)$, $ans = 10$ and $b = (10)$,
- for $a = (10)$, $ans = 0$.

So the returned value is $19 + 10 + 0 = 29$.

In the second example:

- for $a = (5, 8, 7, 1, 9)$, $ans = 25$ and $b = (8, 8, 9)$,
- for $a = (8, 8, 9)$, $ans = 9$ and $b = (9)$,
- for $a = (9)$, $ans = 0$.

So the returned value is $25 + 9 + 0 = 34$.

G. A Game on Strings

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice and Bob are playing a game on strings.

Initially, they have some string t. In one move the first player selects the character c present in t and erases all it's occurrences in t, thus splitting t into many smaller strings. The game then goes independently with each of the strings — to make the move player selects one of the strings and one of the characters there, deletes all occurrences and adds the remaining string back to the game.

Alice always starts the game, and then Alice and Bob take turns making moves. The player who is unable to make a move (because there is no string left) loses.

Alice and Bob used to always start with a string s, but recently they found out that this became too boring. Now before each game they choose two integers l and r such that $1 \le l \le r \le |s|$ and play the game with the string $s_l s_{l+1} s_{l+2} \ldots s_r$ instead.

Given the string s and integers l, r for each game. Find who is going to win each game assuming they are smart and are playing optimally.

Input

The first line contains the string s ($1 \le |s| \le 10^5$) consisting of lowercase English letters. This is the string Alice and Bob used to start with.

The second line contains a single integer m ($1 \le m \le 10^5$) — the number of games to analyze.

Each of the next m lines contains two integers l and r ($1 \le l \le r \le |s|$) — the bounds of the starting substring in the string s.

Output

For each game output a single line containing the name of the winner — "Alice" or "Bob" respectively.

Examples

input
aaab 2 1 2 1 4
output
Alice Bob

input
aaccbdb 2 5 7 1 7
output
Alice Alice

Note

In the first example,

1. In the first game the string "aa" is selected. Alice deletes character 'a' and Bob is unable to move.
2. In the second game the string "aaab" is selected. No matter what character Alice will delete, Bob deletes the other one and Alice is unable to move.

In the second example Alice wins both game "bdb" and "aaccbdb".

To win game "bdb" Alice can erase symbol 'd', the game then goes independently on strings "b" and "b". Bob deletes one of this strings and the Alice deletes the other one and Bob is unable to move.

To win game "aaccbdb" Alice can erase symbol 'd', the game then goes independently on strings "aaccb" and "b". It is possible to show, that no matter what are the moves, the remaining game can only finish in exactly 4 moves, so the Bob will be unable to move after that.

H. Security

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Some programming website is establishing a secure communication protocol. For security reasons, they want to choose several more or less random strings.

Initially, they have a string s consisting of lowercase English letters. Now they want to choose q strings using the following steps, and you are to help them.

1. A string x consisting of lowercase English letters and integers l and r ($1 \le l \le r \le |s|$) are chosen.
2. Consider all non-empty distinct substrings of the $s_l s_{l+1} \ldots s_r$, that is all distinct strings $s_i s_{i+1} \ldots s_j$ where $l \le i \le j \le r$. Among all of them choose all strings that are lexicographically greater than x.
3. If there are no such strings, you should print -1. Otherwise print the lexicographically smallest among them.

String a is lexicographically less than string b, if either a is a prefix of b and $a \neq b$, or there exists such a position i ($1 \le i \le min(|a|, |b|)$), such that $a_i < b_i$ and for all j ($1 \le j < i$) $a_j = b_j$. Here $|a|$ denotes the length of the string a.

Input

The first line of input contains a non-empty string s ($1 \le |s| \le 10^5$) consisting of lowercase English letters.

The second line contains an integer q ($1 \le q \le 2 \cdot 10^5$) — the number of strings to select.

Each of the next q lines contains two integers l, r ($1 \le l \le r \le |s|$) and a non-empty string x consisting of lowercase English letters. The total length of strings x for all queries does not exceed $2 \cdot 10^5$.

Output

Output q lines, each of them should contain the desired string or -1, if there is no such string.

Examples

input
baa 5 1 2 ba 2 3 a 1 2 b 2 3 aa 1 3 b
output
-1 aa ba -1 ba

input
bacb 4 1 2 ba 2 3 ac 1 3 ac 3 4 c
output
-1 c b cb

input
bba 1 1 1 b
output
-1

Note

Consider the first example.

The string s is "baa". The queries are as follows.

1. We consider the substring $s_1 s_2$ that is "ba". It has substrings "b", "a" and "ba", since none of them is greater than the query string "ba", the answer is -1.
2. We consider substring "aa". Among its substrings only "aa" is greater than the query string "a". So the answer is "aa".
3. We consider substring "ba". Out of "b", "a" and "ba" only "ba" is greater than the query string "b", so the answer is "ba".

4. We consider substring "aa". No substring of "aa" is greater than the query string "aa" so the answer is -1.
5. We consider substring "baa" and it has (among others) substrings "ba", "baa" which are greater than the query string "b". Since "ba" is lexicographically smaller than "baa", the answer is "ba".
