Educational Codeforces Round 53 (Rated for Div. 2)

A. Diverse Substring

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

You are given a string s, consisting of n lowercase Latin letters.

A substring of string s is a continuous segment of letters from s. For example, "defor" is a substring of "codeforces" and "fors" is not.

The length of the substring is the number of letters in it.

Let's call some string of length n *diverse* if and only if there is no letter to appear strictly more than $\frac{n}{2}$ times. For example, strings "abc" and "iltlml" are *diverse* and strings "aab" and "zz" are not.

Your task is to find **any** *diverse* substring of string s or report that there is none. Note that it is not required to maximize or minimize the length of the resulting substring.

Input
The first line contains a single integer n ($1 \le n \le 1000$) — the length of string s.

The second line is the string s, consisting of exactly n lowercase Latin letters.

Output
Print "NO" if there is no *diverse* substring in the string s.

Otherwise the first line should contain "YES". The second line should contain **any** *diverse* substring of string s.

Examples

input
10 codeforces
output
YES code

input
5 aaaaa
output
NO

Note
The first example has lots of correct answers.

Please, restrain yourself from asking if some specific answer is correct for some specific test or not, these questions always lead to "No comments" answer.

B. Vasya and Books

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

Vasya has got n books, numbered from 1 to n, arranged in a stack. The topmost book has number a_1, the next one — a_2, and so on. The book at the bottom of the stack has number a_n. **All numbers are distinct**.

Vasya wants to move all the books to his backpack in n steps. During i-th step he wants to move the book number b_i into his backpack. If the book with number b_i is in the stack, he takes this book and all the books **above** the book b_i, and puts them into the backpack; otherwise he does nothing and begins the next step. For example, if books are arranged in the order $[1, 2, 3]$ (book 1 is the topmost), and Vasya moves the books in the order $[2, 1, 3]$, then during the first step he will move two books (1 and 2), during the second step he will do nothing (since book 1 is already in the backpack), and during the third step — one book (the book number 3). **Note that $b_1, b_2, \ldots, b_n$ are distinct.**

Help Vasya! Tell him the number of books he will put into his backpack during each step.

Input
The first line contains one integer n $(1 \le n \le 2 \cdot 10^5)$ — the number of books in the stack.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$ denoting the stack of books.

The third line contains n integers $b_1, b_2, \ldots, b_n$ $(1 \le b_i \le n)$ denoting the steps Vasya is going to perform.

All numbers $a_1 \ldots a_n$ are distinct, the same goes for $b_1 \ldots b_n$.

Output
Print n integers. The i-th of them should be equal to the number of books Vasya moves to his backpack during the i-th step.

Examples

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

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

input
6 6 5 4 3 2 1 6 5 3 4 2 1
output
1 1 2 0 1 1

Note
The first example is described in the statement.

In the second example, during the first step Vasya will move the books $[3, 1, 4]$. After that only books 2 and 5 remain in the stack (2 is above 5). During the second step Vasya will take the books 2 and 5. After that the stack becomes empty, so during next steps Vasya won't move any books.

C. Vasya and Robot

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

Vasya has got a robot which is situated on an infinite Cartesian plane, initially in the cell $(0, 0)$. Robot can perform the following four kinds of operations:

- U — move from (x, y) to $(x, y + 1)$;
- D — move from (x, y) to $(x, y - 1)$;
- L — move from (x, y) to $(x - 1, y)$;
- R — move from (x, y) to $(x + 1, y)$.

Vasya also has got a sequence of n operations. Vasya wants to modify this sequence so after performing it the robot will end up in (x, y).

Vasya wants to change the sequence so the length of changed subsegment is minimum possible. This length can be calculated as follows: $maxID - minID + 1$, where $maxID$ is the maximum index of a changed operation, and $minID$ is the minimum index of a changed operation. For example, if Vasya changes RRRRRRR to RLRRLRL, then the operations with indices 2, 5 and 7 are changed, so the length of changed subsegment is $7 - 2 + 1 = 6$. Another example: if Vasya changes DDDD to DDRD, then the length of changed subsegment is 1.

If there are no changes, then the length of changed subsegment is 0. Changing an operation means replacing it with some operation (possibly the same); Vasya can't insert new operations into the sequence or remove them.

Help Vasya! Tell him the minimum length of subsegment that he needs to change so that the robot will go from $(0, 0)$ to (x, y), or tell him that it's impossible.

Input
The first line contains one integer number n $(1 \le n \le 2 \cdot 10^5)$ — the number of operations.

The second line contains the sequence of operations — a string of n characters. Each character is either U, D, L or R.

The third line contains two integers x, y $(-10^9 \le x, y \le 10^9)$ — the coordinates of the cell where the robot should end its path.

Output
Print one integer — the minimum possible length of subsegment that can be changed so the resulting sequence of operations moves the robot from $(0, 0)$ to (x, y). If this change is impossible, print -1.

Examples

input
5 RURUU -2 3
output
3

input
4 RULR 1 1
output
0

input
3 UUU 100 100
output
-1

Note
In the first example the sequence can be changed to LULUU. So the length of the changed subsegment is $3 - 1 + 1 = 3$.

In the second example the given sequence already leads the robot to (x, y), so the length of the changed subsegment is 0.

In the third example the robot can't end his path in the cell (x, y).

D. Berland Fair

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

XXI Berland Annual Fair is coming really soon! Traditionally fair consists of n booths, arranged in a circle. The booths are numbered 1 through n clockwise with n being adjacent to 1. The i-th booths sells some candies for the price of a_i burles per item. Each booth has an unlimited supply of candies.

Polycarp has decided to spend at most T burles at the fair. However, he has some plan in mind for his path across the booths:

- at first, he visits booth number 1;
- if he has enough burles to buy **exactly one** candy from the current booth, then he buys it immediately;
- then he proceeds to the next booth in the clockwise order (regardless of if he bought a candy or not).

Polycarp's money is finite, thus the process will end once he can no longer buy candy at any booth.

Calculate the number of candies Polycarp will buy.

Input
The first line contains two integers n and T $(1 \le n \le 2 \cdot 10^5, 1 \le T \le 10^{18})$ — the number of booths at the fair and the initial amount of burles Polycarp has.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — the price of the single candy at booth number i.

Output
Print a single integer — the total number of candies Polycarp will buy.

Examples

input
3 38

5 2 5

output
10

input
5 21 2 4 100 2 6

output
6

Note

Let's consider the first example. Here are Polycarp's moves until he runs out of money:

1. Booth 1, buys candy for 5, $T = 33$;
2. Booth 2, buys candy for 2, $T = 31$;
3. Booth 3, buys candy for 5, $T = 26$;
4. Booth 1, buys candy for 5, $T = 21$;
5. Booth 2, buys candy for 2, $T = 19$;
6. Booth 3, buys candy for 5, $T = 14$;
7. Booth 1, buys candy for 5, $T = 9$;
8. Booth 2, buys candy for 2, $T = 7$;
9. Booth 3, buys candy for 5, $T = 2$;
10. Booth 1, buys no candy, not enough money;
11. Booth 2, buys candy for 2, $T = 0$.

No candy can be bought later. The total number of candies bought is 10.

In the second example he has 1 burle left at the end of his path, no candy can be bought with this amount.

E. Segment Sum

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

You are given two integers l and r ($l \le r$). Your task is to calculate the sum of numbers from l to r (including l and r) such that each number contains **at most** k different digits, and print this sum modulo 998244353.

For example, if $k = 1$ then you have to calculate all numbers from l to r such that each number is formed using only one digit. For $l = 10, r = 50$ the answer is $11 + 22 + 33 + 44 = 110$.

Input

The only line of the input contains three integers l, r and k ($1 \le l \le r < 10^{18}, 1 \le k \le 10$) — the borders of the segment and the maximum number of different digits.

Output

Print one integer — the sum of numbers from l to r such that each number contains at most k different digits, modulo 998244353.

Examples

input
10 50 2

output
1230

input
1 2345 10

output
2750685

input
101 154 2

output
2189

Note

For the first example the answer is just the sum of numbers from l to r which equals to $\frac{50 \cdot 51}{2} - \frac{9 \cdot 10}{2} = 1230$. This example also explained in the problem statement but for $k = 1$.

For the second example the answer is just the sum of numbers from l to r which equals to $\frac{2345 \cdot 2346}{2} = 2750685$.

For the third example the answer is
$101 + 110 + 111 + 112 + 113 + 114 + 115 + 116 + 117 + 118 + 119 + 121 + 122 + 131 + 133 + 141 + 144 + 151 = 2189$.

F. Choosing Two Paths

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

You are given an undirected unweighted tree consisting of n vertices.

An undirected tree is a connected undirected graph with $n - 1$ edges.

Your task is to choose two pairs of vertices of this tree (all the chosen vertices **should be distinct**) (x_1, y_1) and (x_2, y_2) in such a way that neither x_1 nor y_1 belong to the simple path from x_2 to y_2 and vice versa (neither x_2 nor y_2 should not belong to the simple path from x_1 to y_1).

It is guaranteed that it is possible to choose such pairs for the given tree.

Among all possible ways to choose such pairs you have to choose one with the **maximum number of common vertices** between paths from x_1 to y_1 and from x_2 to y_2. And among all such pairs you have to choose one with the **maximum total length** of these two paths.

It is guaranteed that the answer with at least two common vertices exists for the given tree.

The length of the path is the number of edges in it.

The simple path is the path that visits each vertex at most once.

Input

The first line contains an integer n — the number of vertices in the tree ($6 \le n \le 2 \cdot 10^5$).

Each of the next $n - 1$ lines describes the edges of the tree.

Edge i is denoted by two integers u_i and v_i, the labels of vertices it connects ($1 \le u_i, v_i \le n$, $u_i \ne v_i$).

It is guaranteed that the given edges form a tree.

It is guaranteed that the answer with at least two common vertices exists for the given tree.

Output

Print **any** two pairs of vertices satisfying the conditions described in the problem statement.

It is guaranteed that it is possible to choose such pairs for the given tree.

Examples

input
7 1 4 1 5 1 6 2 3 2 4 4 7
output
3 6 7 5

input
9 9 3 3 5 1 2 4 3 4 7 1 7 4 6 3 8
output
2 9 6 8

input
10 6 8

```
10 3
3 7
5 8
1 7
7 2
2 9
2 8
1 4
```

output

```
10 6
4 5
```

input

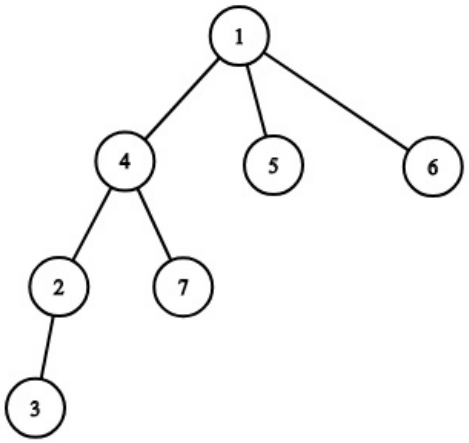
```
11
1 2
2 3
3 4
1 5
1 6
6 7
5 8
5 9
4 10
4 11
```

output

```
9 11
8 10
```

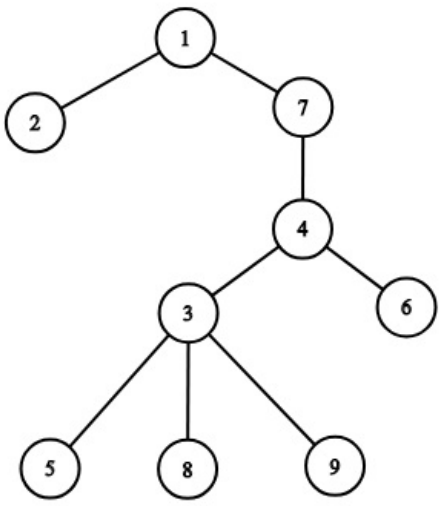
Note

The picture corresponding to the first example:



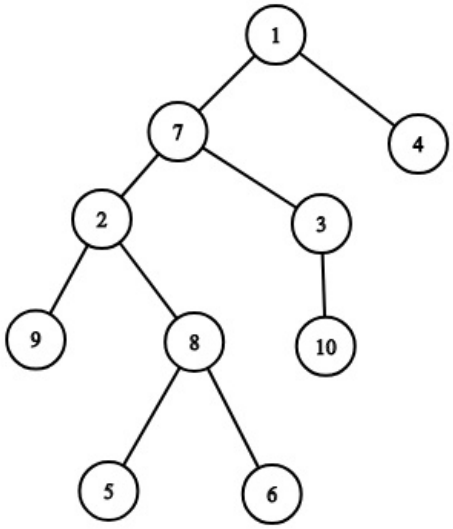
The intersection of two paths is 2 (vertices 1 and 4) and the total length is $4 + 3 = 7$.

The picture corresponding to the second example:



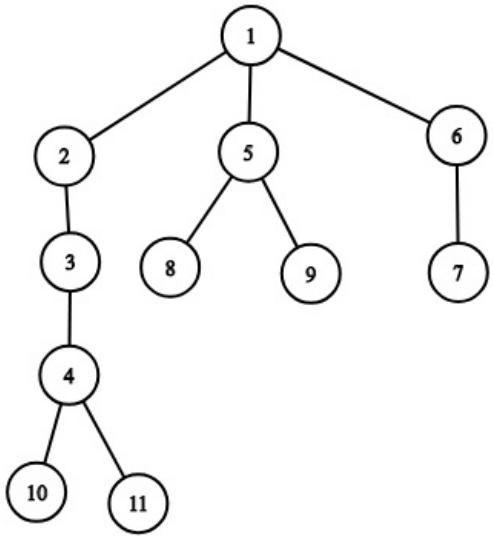
The intersection of two paths is 2 (vertices 3 and 4) and the total length is $5 + 3 = 8$.

The picture corresponding to the third example:



The intersection of two paths is 3 (vertices 2, 7 and 8) and the total length is $5 + 5 = 10$.

The picture corresponding to the fourth example:



The intersection of two paths is 5 (vertices 1, 2, 3, 4 and 5) and the total length is $6 + 6 = 12$.

G. Yet Another LCP Problem

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

Let $\mathrm{LCP}(s, t)$ be the length of the longest common prefix of strings s and t. Also let $s[x \dots y]$ be the substring of s from index x to index y (inclusive). For example, if $s =$ "abcde", then $s[1 \dots 3] =$ "abc", $s[2 \dots 5] =$ "bcde".

You are given a string s of length n and q queries. Each query is a pair of integer sets $a_1, a_2, \dots, a_k$ and $b_1, b_2, \dots, b_l$. Calculate $\sum_{i=1}^{i=k} \sum_{j=1}^{j=l} \mathrm{LCP}(s[a_i \dots n], s[b_j \dots n])$ for each query.

Input

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

The second line contains a string s consisting of lowercase Latin letters ($|s| = n$).

Next $3q$ lines contains descriptions of queries — three lines per query. The first line of each query contains two integers k_i and l_i ($1 \le k_i, l_i \le n$) — sizes of sets a and b respectively.

The second line of each query contains k_i integers $a_1, a_2, \dots a_{k_i}$ ($1 \le a_1 < a_2 < \dots < a_{k_i} \le n$) — set a.

The third line of each query contains l_i integers $b_1, b_2, \dots b_{l_i}$ ($1 \le b_1 < b_2 < \dots < b_{l_i} \le n$) — set b.

It is guaranteed that $\sum_{i=1}^{i=q} k_i \le 2 \cdot 10^5$ and $\sum_{i=1}^{i=q} l_i \le 2 \cdot 10^5$.

Output

Print q integers — answers for the queries in the same order queries are given in the input.

Example

input
7 4 abacaba 2 2 1 2 1 2 3 1 1 2 3 7 1 7 1 1 2 3 4 5 6 7 2 2 1 5 1 5

output
13 2 12 16

Note

Description of queries:

1. In the first query $s[1 \dots 7] = \text{abacaba}$ and $s[2 \dots 7] = \text{bacaba}$ are considered. The answer for the query is $\text{LCP}(\text{abacaba}, \text{abacaba}) + \text{LCP}(\text{abacaba}, \text{bacaba}) + \text{LCP}(\text{bacaba}, \text{abacaba}) + \text{LCP}(\text{bacaba}, \text{bacaba}) = 7 + 0 + 0 + 6 = 13$.
2. In the second query $s[1 \dots 7] = \text{abacaba}$, $s[2 \dots 7] = \text{bacaba}$, $s[3 \dots 7] = \text{acaba}$ and $s[7 \dots 7] = \text{a}$ are considered. The answer for the query is $\text{LCP}(\text{abacaba}, \text{a}) + \text{LCP}(\text{bacaba}, \text{a}) + \text{LCP}(\text{acaba}, \text{a}) = 1 + 0 + 1 = 2$.
3. In the third query $s[1 \dots 7] = \text{abacaba}$ are compared with all suffixes. The answer is the sum of non-zero values: $\text{LCP}(\text{abacaba}, \text{abacaba}) + \text{LCP}(\text{abacaba}, \text{acaba}) + \text{LCP}(\text{abacaba}, \text{aba}) + \text{LCP}(\text{abacaba}, \text{a}) = 7 + 1 + 3 + 1 = 12$.
4. In the fourth query $s[1 \dots 7] = \text{abacaba}$ and $s[5 \dots 7] = \text{aba}$ are considered. The answer for the query is $\text{LCP}(\text{abacaba}, \text{abacaba}) + \text{LCP}(\text{abacaba}, \text{aba}) + \text{LCP}(\text{aba}, \text{abacaba}) + \text{LCP}(\text{aba}, \text{aba}) = 7 + 3 + 3 + 3 = 16$.