A. Snow Footprints

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

There is a straight snowy road, divided into n blocks. The blocks are numbered from 1 to n from left to right. If one moves from the i-th block to the $(i+1)$-th block, he will leave a right footprint on the i-th block. Similarly, if one moves from the i-th block to the $(i - 1)$-th block, he will leave a left footprint on the i-th block. If there already is a footprint on the i-th block, the new footprint will cover the old one.

At the beginning, there were no footprints. Then polar bear Alice starts from the s-th block, makes a sequence of moves and ends in the t-th block. It is known that Alice never moves outside of the road.

You are given the description of Alice's footprints. Your task is to find a pair of possible values of s, t by looking at the footprints.

Input

The first line of the input contains integer n $(3 \le n \le 1000)$.

The second line contains the description of the road — the string that consists of n characters. Each character will be either "`.`" (a block without footprint), or "L" (a block with a left footprint), "R" (a block with a right footprint).

It's guaranteed that the given string contains at least one character not equal to "`.`". Also, the first and the last character will always be "`.`". It's guaranteed that a solution exists.

Output

Print two space-separated integers — the values of s and t. If there are several possible solutions you can print any of them.

Examples

input

```
9
..RRLL...
```

output

```
3 4
```

input

```
11
.RRRLLLLL..
```

output

```
7 5
```

Note

The first test sample is the one in the picture.

B. Sail

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

The polar bears are going fishing. They plan to sail from (s_x, s_y) to (e_x, e_y). However, the boat can only sail by wind. At each second, the wind blows in one of these directions: east, south, west or north. Assume the boat is currently at (x, y).

- If the wind blows to the east, the boat will move to $(x + 1, y)$.
- If the wind blows to the south, the boat will move to $(x, y - 1)$.
- If the wind blows to the west, the boat will move to $(x - 1, y)$.
- If the wind blows to the north, the boat will move to $(x, y + 1)$.

Alternatively, they can hold the boat by the anchor. In this case, the boat stays at (x, y). Given the wind direction for t seconds, what is the earliest time they sail to (e_x, e_y)?

Input

The first line contains five integers t, s_x, s_y, e_x, e_y $(1 \le t \le 10^5, -10^9 \le s_x, s_y, e_x, e_y \le 10^9)$. The starting location and the ending location will be different.

The second line contains t characters, the i-th character is the wind blowing direction at the i-th second. It will be one of the four possibilities: "E" (east), "S" (south), "W" (west) and "N" (north).

Output

If they can reach (e_x, e_y) within t seconds, print the earliest time they can achieve it. Otherwise, print "-1" (without quotes).

Examples

input

```
5 0 0 1 1
SESNW
```

output

```
4
```

input

```
10 5 3 3 6
NENSWESNEE
```

output

```
-1
```

Note

In the first sample, they can stay at seconds 1, 3, and move at seconds 2, 4.

In the second sample, they cannot sail to the destination.

C. Parity Game

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

You are fishing with polar bears Alice and Bob. While waiting for the fish to bite, the polar bears get bored. They come up with a game. First Alice and Bob each writes a 01-string (strings that only contain character "0" and "1") a and b. Then you try to turn a into b using two types of operations:

- Write $parity(a)$ to the end of a. For example, $1010 \to 10100$.
- Remove the first character of a. For example, $1001 \to 001$. You cannot perform this operation if a is empty.

You can use as many operations as you want. The problem is, is it possible to turn a into b?

The $parity$ of a 01-string is 1 if there is an odd number of "1"s in the string, and 0 otherwise.

Input

The first line contains the string a and the second line contains the string b $(1 \leq |a|, |b| \leq 1000)$. Both strings contain only the characters "0" and "1". Here $|x|$ denotes the length of the string x.

Output

Print "YES" (without quotes) if it is possible to turn a into b, and "NO" (without quotes) otherwise.

Examples

input

```
01011
0110
```

output

```
YES
```

input

```
0011
1110
```

output

```
NO
```

Note

In the first sample, the steps are as follows: $01011 \to 1011 \to 011 \to 0110$

D. Fish Weight

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

It is known that there are k fish species in the polar ocean, numbered from 1 to k. They are sorted by non-decreasing order of their weight, which is a positive number. Let the weight of the i-th type of fish be w_i, then $0 < w_1 \leq w_2 \leq \ldots \leq w_k$ holds.

Polar bears Alice and Bob each have caught some fish, and they are guessing who has the larger sum of weight of the fish he/she's caught. Given the type of the fish they've caught, determine whether it is possible that the fish caught by Alice has a **strictly larger** total weight than Bob's. In other words, does there exist a sequence of weights w_i (not necessary integers), such that the fish caught by Alice has a strictly larger total weight?

Input

The first line contains three integers n, m, k $(1 \leq n, m \leq 10^5, 1 \leq k \leq 10^9)$ — the number of fish caught by Alice and Bob respectively, and the number of fish species.

The second line contains n integers each from 1 to k, the list of fish type caught by Alice. The third line contains m integers each from 1 to k, the list of fish type caught by Bob.

Note that one may have caught more than one fish for a same species.

Output

Output "YES" (without quotes) if it is possible, and "NO" (without quotes) otherwise.

Examples

input

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

output

```
YES
```

input

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

output

```
NO
```

Note

In the first sample, if $w_1 = 1, w_2 = 2, w_3 = 2.5$, then Alice has a total of $2 + 2 + 2 = 6$ weight units, while Bob only has $1 + 1 + 2.5 = 4.5$.

In the second sample, the fish that Alice caught is a subset of Bob's. Therefore, the total weight of Bob’s fish is always not less than the total weight of Alice’s fish.

E. Splitting the Uniqueness

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

Polar bears like *unique arrays* — that is, arrays without repeated elements.

You have got a unique array s with length n containing non-negative integers. Since you are good friends with Alice and Bob, you decide to split the array in two. Precisely, you need to construct two arrays a and b that are also of length n, with the following conditions for all i $(1 \le i \le n)$:

- a_i, b_i are non-negative integers;
- $s_i = a_i + b_i$.

Ideally, a and b should also be unique arrays. However, life in the Arctic is hard and this is not always possible. Fortunately, Alice and Bob are still happy if their arrays are *almost unique*. We define an array of length n to be almost unique, if and only if it can be turned into a unique array by removing no more than $\lceil \frac{n}{3} \rceil$ entries.

For example, the array $[1, 2, 1, 3, 2]$ is almost unique because after removing the first two entries, it becomes $[1, 3, 2]$. The array $[1, 2, 1, 3, 1, 2]$ is not almost unique because we need to remove at least 3 entries to turn it into a unique array.

So, your task is to split the given unique array s into two almost unique arrays a and b.

Input

The first line of the input contains integer n $(1 \le n \le 10^5)$.

The second line contains n distinct integers $s_1, s_2, \ldots s_n$ $(0 \le s_i \le 10^9)$.

Output

If it is possible to make Alice and Bob happy (if you can split the given array), print "YES" (without quotes) in the first line. In the second line, print the array a. In the third line, print the array b. There may be more than one solution. Any of them will be accepted.

If it is impossible to split s into almost unique arrays a and b, print "NO" (without quotes) in the first line.

Examples

input

```
6
12 5 8 3 11 9
```

output

```
YES
6 2 6 0 2 4
6 3 2 3 9 5
```

Note

In the sample, we can remove the first two entries from a and the second entry from b to make them both unique.