## Codeforces Round #180 (Div. 1)

# A. 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, $1010 \rightarrow 10100$.
- Remove the first character of $a$. For example, $1001 \rightarrow 001$. 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 $b$ ($1 \le |a|, |b| \le 1000$). 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<br>0110 |
| **output** |
| YES |

| input |
| --- |
| 0011<br>1110 |
| **output** |
| NO |

### Note

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

# B. 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<br>2 2 2<br>1 1 3 |
| **output** |
| YES |

| input |
| --- |
| 4 7 9<br>5 2 7 3<br>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.

# C. 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<br>12 5 8 3 11 9 |
| **output** |
| YES<br>6 2 6 0 2 4<br>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.

# D. Color the Carpet

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

Even polar bears feel cold when lying on the ice. Therefore, a polar bear Alice is going to make a carpet. The carpet can be viewed as a grid with height $h$ and width $w$. Then the grid is divided into $h \times w$ squares. Alice is going to assign one of $k$ different colors to each square. The colors are numbered from 1 to $k$. She may choose not to use all of the colors.

However, there are some restrictions. For every two adjacent squares (squares that shares an edge) $x$ and $y$, there is a *color constraint* in one of the forms:

- $color(x) = color(y)$, or
- $color(x) \neq color(y)$.

Example of the color constraints:

Ideally, Alice wants to satisfy all color constraints. But again, life in the Arctic is hard. It is not always possible to satisfy all color constraints. Fortunately, she will still be happy if at least $\frac{3}{4}$ of the color constraints are satisfied.

If she has $4$ colors she can color the carpet in the following way:

And she is happy because $\frac{13}{17}$ of the color constraints are satisfied, and $\frac{13}{17} \geq \frac{3}{4}$. Your task is to help her color the carpet.

## Input

The first line contains three integers $h, w, k$ ($2 \leq h, w \leq 1000, 1 \leq k \leq w \cdot h$).

The next $2h - 1$ lines describe the color constraints from top to bottom, left to right. They contain $w - 1, w, w - 1, w, \ldots, w - 1$ characters respectively. Each color constraint is represented by a character "E" or "N", where "E" means " $=$ " and "N" means " $\neq$ ".

The color constraints listed in the order they are depicted on the picture.

## Output

If there is a coloring that satisfies at least $\frac{3}{4}$ of the color constraints, print "YES" (without quotes) in the first line. In each of the next $h$ lines, print $w$ integers describing the coloring.

Otherwise, print "NO" (without quotes).

### Examples

| input |
| --- |
| 3 4 4<br>ENE<br>NNEE<br>NEE<br>ENEN<br>ENN |
| **output** |
| YES<br>1 1 2 2<br>3 4 1 1<br>3 3 2 4 |

# E. Mystic Carvings

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

The polar bears have discovered a gigantic circular piece of floating ice with some mystic carvings on it. There are $n$ lines carved on the ice. Each line connects two points on the boundary of the ice (we call these points *endpoints*). The endpoints are numbered $1, 2, \ldots, 2n$ counter-clockwise along the circumference. No two lines share an endpoint.

Now a group of 6 polar bears (Alice, Bob, Carol, Dave, Eve, Frank) are going to build caves on the endpoints. Each polar bear would build a cave and live in it. No two polar bears can build a cave on the same endpoints. Alice and Bob is a pair of superstitious lovers. They believe the lines are carved by aliens (or humans, which are pretty much the same thing to polar bears), and have certain spiritual power. Therefore they want to build their caves on two endpoints which are connected by a line. The same for Carol and Dave, Eve and Frank.

The *distance* between two caves X and Y is defined as one plus minimum number of other caves one need to pass through in order to travel from X to Y along the boundary of the ice (endpoints without caves are not counted).

To ensure fairness, the distances between the three pairs of lovers have to be the same (that is, the distance between Alice and Bob, the distance between Carol and Dave, and the distance between Eve and Frank are the same).

The figures below show two different configurations, where the dots on the circle are the endpoints. The configuration on the left is not valid. Although each pair of lovers (A and B, C and D, E and F) is connected a line, the distance requirement is not satisfied. The distance between A and B is 2 (one can go from A to B in the clockwise direction passing through F). The distance between E and F is also 2. However, the distance between C and D is 1 (one can go from C to D in the counter-clockwise direction without passing through any other caves). The configuration on the right is valid. All three pairs have the same distance 1.



Count the number of ways to build the caves under the requirements. Two configurations are considered the same if the same set of 6 endpoints are used.

## Input

The first line contains integer $n (3 \le n \le 10^5)$ — the number of lines.

Each of the following $n$ lines contains two integers $a_i, b_i$ $(1 \le a_i, b_i \le 2n)$, which means that there is a line carved on the ice connecting the $a_i$-th and $b_i$-th endpoint.

It's guaranteed that each endpoints touches exactly one line.

## Output

Print the number of ways to build the caves.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

### Examples

| input |
| --- |
| 4<br>5 4<br>1 2<br>6 7<br>8 3 |
| **output** |
| 2 |

| input |
| --- |
| 8<br>1 7<br>2 4<br>3 9<br>5 11<br>6 8<br>10 16<br>13 15<br>14 12 |
| **output** |
| 6 |

## Note

The second sample corresponds to the figure in the problem statement.