# A. Integer Sequence Dividing

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

You are given an integer sequence $1, 2, \dots, n$. You have to divide it into two sets $A$ and $B$ in such a way that each element belongs to **exactly one** set and $|sum(A) - sum(B)|$ is minimum possible.

The value $|x|$ is the absolute value of $x$ and $sum(S)$ is the sum of elements of the set $S$.

### Input

The first line of the input contains one integer $n$ ($1 \le n \le 2 \cdot 10^9$).

### Output

Print one integer — the minimum possible value of $|sum(A) - sum(B)|$ if you divide the initial sequence $1, 2, \dots, n$ into two sets $A$ and $B$.

#### Examples

**input**
```
3
```
**output**
```
0
```

**input**
```
5
```
**output**
```
1
```

**input**
```
6
```
**output**
```
1
```

### Note

Some (not all) possible answers to examples:

In the first example you can divide the initial sequence into sets $A = \{1, 2\}$ and $B = \{3\}$ so the answer is $0$.

In the second example you can divide the initial sequence into sets $A = \{1, 3, 4\}$ and $B = \{2, 5\}$ so the answer is $1$.

In the third example you can divide the initial sequence into sets $A = \{1, 4, 5\}$ and $B = \{2, 3, 6\}$ so the answer is $1$.

# B. Array K-Coloring

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

You are given an array $a$ consisting of $n$ integer numbers.

You have to color this array in $k$ colors in such a way that:

- Each element of the array should be colored in some color;
- For each $i$ from $1$ to $k$ there should be **at least one** element colored in the $i$-th color in the array;
- For each $i$ from $1$ to $k$ all elements colored in the $i$-th color should be **distinct**.

Obviously, such coloring might be impossible. In this case, print "NO". Otherwise print "YES" and **any** coloring (i.e. numbers $c_1, c_2, \dots c_n$, where $1 \le c_i \le k$ and $c_i$ is the color of the $i$-th element of the given array) satisfying the conditions above. If there are multiple answers, you can print **any**.

### Input

The first line of the input contains two integers $n$ and $k$ ($1 \le k \le n \le 5000$) — the length of the array $a$ and the number of colors, respectively.

The second line of the input contains $n$ integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 5000$) — elements of the array $a$.

### Output

If there is no answer, print "NO". Otherwise print "YES" and **any** coloring (i.e. numbers $c_1, c_2, \dots c_n$, where $1 \le c_i \le k$ and $c_i$ is the color of the $i$-th element of the given array) satisfying the conditions described in the problem statement. If there are multiple answers, you can print **any**.

#### Examples

**input**

```
4 2
1 2 2 3
```

**output**

```
YES
1 1 2 2
```

**input**

```
5 2
3 2 1 2 3
```

**output**

```
YES
2 1 1 2 1
```

**input**

```
5 2
2 1 1 2 1
```

**output**

```
NO
```

### Note

In the first example the answer $2\ 1\ 2\ 1$ is also acceptable.

In the second example the answer $1\ 1\ 1\ 2\ 2$ is also acceptable.

There exist other acceptable answers for both examples.

## C. Doors Breaking and Repairing

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are policeman and you are playing a game with Slavik. The game is turn-based and each turn consists of two phases. During the first phase you make your move and during the second phase Slavik makes his move.

There are $n$ doors, the $i$-th door initially has durability equal to $a_i$.

During your move you can try to break one of the doors. If you choose door $i$ and its current durability is $b_i$ then you reduce its durability to $max(0, b_i - x)$ (the value $x$ is given).

During Slavik's move he tries to repair one of the doors. If he chooses door $i$ and its current durability is $b_i$ then he increases its durability to $b_i + y$ (the value $y$ is given). **Slavik cannot repair doors with current durability equal to** $0$.

The game lasts $10^{100}$ turns. If some player cannot make his move then he has to skip it.

Your goal is to maximize the number of doors with durability equal to $0$ at the end of the game. You can assume that Slavik **wants to minimize** the number of such doors. What is the number of such doors in the end if you both play optimally?

### Input

The first line of the input contains three integers $n$, $x$ and $y$ ($1 \le n \le 100$, $1 \le x, y \le 10^5$) — the number of doors, value $x$ and value $y$, respectively.

The second line of the input contains $n$ integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^5$), where $a_i$ is the initial durability of the $i$-th door.

### Output

Print one integer — the number of doors with durability equal to $0$ at the end of the game, if you and Slavik both play optimally.

#### Examples

**input**

```
6 3 2
```

```
2 3 1 3 4 2
```

**output**

```
6
```

**input**

```
5 3 3
1 2 4 2 3
```

**output**

```
2
```

**input**

```
5 5 6
1 2 6 10 3
```

**output**

```
2
```

## Note

Clarifications about the optimal strategy will be ignored.

# D. Balanced Ternary String

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

You are given a string $s$ consisting of exactly $n$ characters, and each character is either '0', '1' or '2'. Such strings are called **ternary strings**.

Your task is to **replace minimum number of characters** in this string with other characters to obtain a *balanced* ternary string (*balanced* ternary string is a ternary string such that the number of characters '0' in this string is equal to the number of characters '1', and the number of characters '1' (and '0' obviously) is equal to the number of characters '2').

Among all possible *balanced* ternary strings you have to obtain the lexicographically (alphabetically) smallest.

Note that you can neither remove characters from the string nor add characters to the string. Also note that you can replace the given characters only with characters '0', '1' and '2'.

It is **guaranteed** that the answer exists.

## Input

The first line of the input contains one integer $n$ ($3 \le n \le 3 \cdot 10^5$, $n$ is **divisible by** $3$) — the number of characters in $s$.

The second line contains the string $s$ consisting of exactly $n$ characters '0', '1' and '2'.

## Output

Print one string — the lexicographically (alphabetically) smallest *balanced* ternary string which can be obtained from the given one with **minimum** number of replacements.

Because $n$ is **divisible by** $3$ it is obvious that the answer exists. And it is obvious that there is only one possible answer.

### Examples

**input**

```
3
121
```

**output**

```
021
```

**input**

```
6
000000
```

**output**

```
001122
```

**input**

```
6
211200
```

**output**

```
211200
```

**input**

```
6
120110
```

**output**

```
120120
```

## E. Monotonic Renumeration

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

You are given an array $a$ consisting of $n$ integers. Let's denote *monotonic renumeration* of array $a$ as an array $b$ consisting of $n$ integers such that all of the following conditions are met:

- $b_1 = 0$;
- for every pair of indices $i$ and $j$ such that $1 \le i, j \le n$, if $a_i = a_j$, then $b_i = b_j$ (note that if $a_i \ne a_j$, it is still possible that $b_i = b_j$);
- for every index $i \in [1, n-1]$ either $b_i = b_{i+1}$ or $b_i + 1 = b_{i+1}$.

For example, if $a = [1, 2, 1, 2, 3]$, then two possible monotonic renumerations of $a$ are $b = [0, 0, 0, 0, 0]$ and $b = [0, 0, 0, 0, 1]$.

Your task is to calculate the number of different monotonic renumerations of $a$. The answer may be large, so print it modulo $998244353$.

### Input

The first line contains one integer $n$ ($2 \le n \le 2 \cdot 10^5$) — the number of elements in $a$.

The second line contains $n$ integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$).

### Output

Print one integer — the number of different monotonic renumerations of $a$, taken modulo $998244353$.

#### Examples

**input**

```
5
1 2 1 2 3
```

**output**

```
2
```

**input**

```
2
100 1
```

**output**

```
2
```

**input**

```
4
1 3 3 7
```

**output**

```
4
```

## F. Elongated Matrix

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

You are given a matrix $a$, consisting of $n$ rows and $m$ columns. Each cell contains an integer in it.

You can change the order of rows arbitrarily (including leaving the initial order), but you can't change the order of cells in a row. After you pick some order of rows, you traverse the whole matrix the following way: firstly visit all cells of the first column from the top row to the bottom one, then the same for the second column and so on. During the traversal you write down the sequence of the numbers on the cells in the same order you visited them. Let that sequence be $s_1, s_2, \dots, s_{nm}$.

The traversal is $k$-acceptable if for all $i$ ($1 \le i \le nm - 1$) $|s_i - s_{i+1}| \ge k$.

Find the maximum integer $k$ such that there exists some order of rows of matrix $a$ that it produces a $k$-acceptable traversal.

### Input

The first line contains two integers $n$ and $m$ ($1 \le n \le 16$, $1 \le m \le 10^4$, $2 \le nm$) — the number of rows and the number of columns, respectively.

Each of the next $n$ lines contains $m$ integers ($1 \le a_{i,j} \le 10^9$) — the description of the matrix.

### Output

Print a single integer $k$ — the maximum number such that there exists some order of rows of matrix $a$ that it produces an $k$-acceptable traversal.

#### Examples

**input**

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

**output**

```
5
```

**input**

```
2 4
1 2 3 4
10 3 7 3
```

**output**

```
0
```

**input**

```
6 1
3
6
2
5
1
4
```

**output**

```
3
```

### Note

In the first example you can rearrange rows as following to get the $5$-acceptable traversal:

```
5 3
10 8
4 3
9 9
```

Then the sequence $s$ will be $[5, 10, 4, 9, 3, 8, 3, 9]$. Each pair of neighbouring elements have at least $k = 5$ difference between them.

In the second example the maximum $k = 0$, any order is $0$-acceptable.

In the third example the given order is already $3$-acceptable, you can leave it as it is.