Codeforces Round #557 (Div. 2) [based on Forethought Future Cup - Final Round]

A. Zoning Restrictions Again

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

You are planning to build housing on a street. There are n spots available on the street on which you can build a house. The spots are labeled from 1 to n from left to right. In each spot, you can build a house with an integer height between 0 and h.

In each spot, if a house has height a, you will gain a^2 dollars from it.

The city has m zoning restrictions. The i-th restriction says that the tallest house from spots l_i to r_i (inclusive) must be at most x_i.

You would like to build houses to maximize your profit. Determine the maximum profit possible.

Input

The first line contains three integers n, h, and m ($1 \le n, h, m \le 50$) — the number of spots, the maximum height, and the number of restrictions.

Each of the next m lines contains three integers l_i, r_i, and x_i ($1 \le l_i \le r_i \le n$, $0 \le x_i \le h$) — left and right limits (inclusive) of the i-th restriction and the maximum possible height in that range.

Output

Print a single integer, the maximum profit you can make.

Examples

input

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

output

```
14
```

input

```
4 10 2
2 3 8
3 4 7
```

output

```
262
```

Note

In the first example, there are 3 houses, the maximum height of a house is 3, and there are 3 restrictions. The first restriction says the tallest house between 1 and 1 must be at most 1. The second restriction says the tallest house between 2 and 2 must be at most 3. The third restriction says the tallest house between 3 and 3 must be at most 2.

In this case, it is optimal to build houses with heights $[1, 3, 2]$. This fits within all the restrictions. The total profit in this case is $1^2 + 3^2 + 2^2 = 14$.

In the second example, there are 4 houses, the maximum height of a house is 10, and there are 2 restrictions. The first restriction says the tallest house from 2 to 3 must be at most 8. The second restriction says the tallest house from 3 to 4 must be at most 7.

In this case, it's optimal to build houses with heights $[10, 8, 7, 7]$. We get a profit of $10^2 + 8^2 + 7^2 + 7^2 = 262$. Note that there are two restrictions on house 3 and both of them must be satisfied. Also, note that even though there isn't any explicit restrictions on house 1, we must still limit its height to be at most 10 ($h = 10$).

B. Double Matrix

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

You are given **two** $n \times m$ matrices containing integers. A sequence of integers is strictly increasing if each next number is greater than the previous one. A row is strictly increasing if all numbers from left to right are strictly increasing. A column is strictly increasing if all numbers from top to bottom are strictly increasing. A matrix is increasing if all rows are strictly increasing **and** all columns are strictly increasing.

For example, the matrix $\begin{bmatrix} 9 & 10 & 11 \\ 11 & 12 & 14 \end{bmatrix}$ is increasing because each individual row and column is strictly increasing. On the other hand, the matrix $\begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}$ is not increasing because the first row is not strictly increasing.

Let a position in the i-th row (from top) and j-th column (from left) in a matrix be denoted as (i, j).

In one operation, you can choose any two numbers i and j and swap the number located in (i, j) in the first matrix with the number in (i, j) in the second matrix. In other words, you can swap two numbers in different matrices if they are located in the corresponding positions.

You would like to make both matrices increasing by performing some number of operations (possibly none). Determine if it is possible to do this. If it is, print "`Possible`", otherwise, print "`Impossible`".

Input

The first line contains two integers n and m ($1 \leq n, m \leq 50$) — the dimensions of each matrix.

Each of the next n lines contains m integers $a_{i1}, a_{i2}, \ldots, a_{im}$ ($1 \leq a_{ij} \leq 10^9$) — the number located in position (i, j) in the first matrix.

Each of the next n lines contains m integers $b_{i1}, b_{i2}, \ldots, b_{im}$ ($1 \leq b_{ij} \leq 10^9$) — the number located in position (i, j) in the second matrix.

Output

Print a string "`Impossible`" or "`Possible`".

Examples

input

```
2 2
2 10
11 5
9 4
3 12
```

output

```
Possible
```

input

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

output

```
Possible
```

input

```
3 2
1 3
2 4
5 10
3 1
3 6
4 8
```

output

```
Impossible
```

Note

The first example, we can do an operation on the top left and bottom right cells of the matrices. The resulting matrices will be $\begin{bmatrix} 9 & 10 \\ 11 & 12 \end{bmatrix}$ and $\begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}$.

In the second example, we don't need to do any operations.

In the third example, no matter what we swap, we can't fix the first row to be strictly increasing in both matrices.

C. Hide and Seek

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

Alice and Bob are playing a game on a line with n cells. There are n cells labeled from 1 through n. For each i from 1 to $n-1$, cells i and $i+1$ are adjacent.

Alice initially has a token on some cell on the line, and Bob tries to guess where it is.

Bob guesses a sequence of line cell numbers $x_1, x_2, \ldots, x_k$ in order. In the i-th question, Bob asks Alice if her token is currently on cell x_i. That is, Alice can answer either "YES" or "NO" to each Bob's question.

At most one time in this process, before or after answering a question, Alice is allowed to move her token from her current cell to some **adjacent** cell. Alice acted in such a way that she was able to answer "NO" to **all** of Bob's questions.

Note that Alice can even move her token before answering the first question or after answering the last question. Alice can also choose to not move at all.

You are given n and Bob's questions $x_1, \ldots, x_k$. You would like to count the number of scenarios that let Alice answer "NO" to all of Bob's questions.

Let (a,b) denote a scenario where Alice starts at cell a and ends at cell b. Two scenarios (a_i, b_i) and (a_j, b_j) are different if $a_i \neq a_j$ or $b_i \neq b_j$.

Input

The first line contains two integers n and k ($1 \leq n, k \leq 10^5$) — the number of cells and the number of questions Bob asked.

The second line contains k integers $x_1, x_2, \ldots, x_k$ ($1 \leq x_i \leq n$) — Bob's questions.

Output

Print a single integer, the number of scenarios that let Alice answer "NO" to all of Bob's questions.

Examples

input

```
5 3
5 1 4
```

output

```
9
```

input

```
4 8
1 2 3 4 4 3 2 1
```

output

```
0
```

input

```
100000 1
42
```

output

```
299997
```

Note

The notation (i,j) denotes a scenario where Alice starts at cell i and ends at cell j.

In the first example, the valid scenarios are $(1,2), (2,1), (2,2), (2,3), (3,2), (3,3), (3,4), (4,3), (4,5)$. For example, $(3,4)$ is valid since Alice can start at cell 3, stay there for the first three questions, then move to cell 4 after the last question.

$(4,5)$ is valid since Alice can start at cell 4, stay there for the first question, the move to cell 5 for the next two questions. Note that $(4,5)$ is only counted once, even though there are different questions that Alice can choose to do the move, but remember, we only count each pair of starting and ending positions once.

In the second example, Alice has no valid scenarios.

In the last example, all (i,j) where $|i-j| \leq 1$ except for $(42, 42)$ are valid scenarios.

D. Chladni Figure

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

Inaka has a disc, the circumference of which is n units. The circumference is equally divided by n points numbered clockwise from 1 to n, such that points i and $i+1$ ($1 \leq i < n$) are adjacent, and so are points n and 1.

There are m straight segments on the disc, the endpoints of which are all among the aforementioned n points.

Inaka wants to know if her image is *rotationally symmetrical*, i.e. if there is an integer k ($1 \leq k < n$), such that if all segments are rotated clockwise around the center of the circle by k units, the new image will be the same as the original one.

Input

The first line contains two space-separated integers n and m ($2 \le n \le 100\,000$, $1 \le m \le 200\,000$) — the number of points and the number of segments, respectively.

The i-th of the following m lines contains two space-separated integers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) that describe a segment connecting points a_i and b_i.

It is guaranteed that no segments coincide.

Output

Output one line — "`Yes`" if the image is rotationally symmetrical, and "`No`" otherwise (both excluding quotation marks).

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

Examples

input

```
12 6
1 3
3 7
5 7
7 11
9 11
11 3
```

output

```
Yes
```

input

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

output

```
Yes
```

input

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

output

```
No
```

input

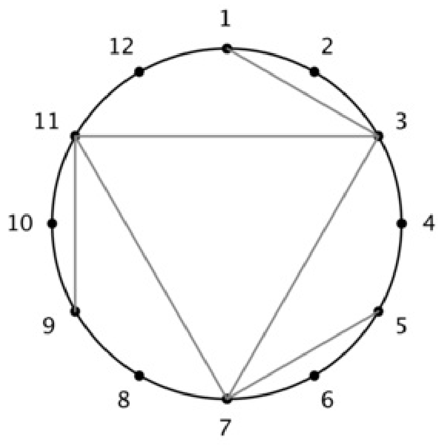
```
10 2
1 6
2 7
```

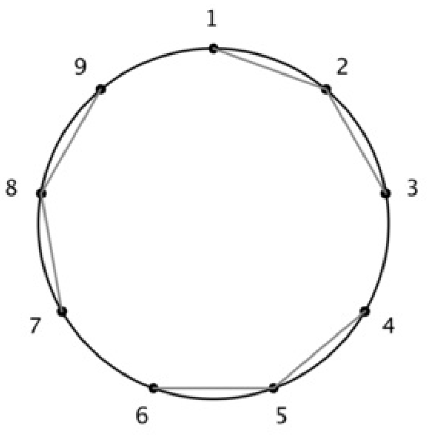
output

```
Yes
```

Note

The first two examples are illustrated below. Both images become the same as their respective original ones after a clockwise rotation of 120 degrees around the center.





E. Thanos Nim

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

Alice and Bob are playing a game with n piles of stones. It is guaranteed that n is an even number. The i-th pile has a_i stones.

Alice and Bob will play a game alternating turns with Alice going first.

On a player's turn, they must choose **exactly** $\frac{n}{2}$ nonempty piles and independently remove a positive number of stones from each of the chosen piles. They **can** remove a **different** number of stones from the piles in a single turn. The first player unable to make a move loses (when there are less than $\frac{n}{2}$ nonempty piles).

Given the starting configuration, determine who will win the game.

Input

The first line contains one integer n $(2 \le n \le 50)$ — the number of piles. It is guaranteed that n is an even number.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 50)$ — the number of stones in the piles.

Output

Print a single string "Alice" if Alice wins; otherwise, print "Bob" (without double quotes).

Examples

input

```
2
8 8
```

output

```
Bob
```

input

```
4
3 1 4 1
```

output

```
Alice
```

Note

In the first example, each player can only remove stones from one pile ($\frac{2}{2} = 1$). Alice loses, since Bob can copy whatever Alice does on the other pile, so Alice will run out of moves first.

In the second example, Alice can remove 2 stones from the first pile and 3 stones from the third pile on her first move to guarantee a win.

F. Palindrome XOR

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 characters "1", "0", and "?". The first character of s is guaranteed to be "1". Let m be the number of characters in s.

Count the number of ways we can choose a pair of integers a, b that satisfies the following:

- $1 \le a < b < 2^m$
- When written without leading zeros, the base-2 representations of a and b are both palindromes.
- The base-2 representation of bitwise XOR of a and b matches the pattern s. We say that t matches s if the lengths of t and s are the same and for every i, the i-th character of t is equal to the i-th character of s, or the i-th character of s is "?".

Compute this count modulo 998244353.

Input

The first line contains a single string s $(1 \le |s| \le 1\,000)$. s consists only of characters "1", "0" and "?". It is guaranteed that the first character of s is a "1".

Output

Print a single integer, the count of pairs that satisfy the conditions modulo 998244353.

Examples

input

```
10110
```

output

```
3
```

input

```
1?0???10
```

output

```
44
```

input

```
1?????????????????????????????????????
```

output

```
519569202
```

input

```
1
```

output

```
0
```

Note

For the first example, the pairs in base-2 are $(111, 10001), (11, 10101), (1001, 11111)$.