Intel Code Challenge Final Round (Div. 1 + Div. 2, Combined)

A. Checking the Calendar

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

You are given names of two days of the week.

Please, determine whether it is possible that during some **non-leap year** the first day of some month was equal to the first day of the week you are given, while the first day of **the next month** was equal to the second day of the week you are given. **Both months should belong to one year**.

In this problem, we consider the Gregorian calendar to be used. The number of months in this calendar is equal to 12. The number of days in months during any non-leap year is: $31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31$.

Names of the days of the week are given with lowercase English letters: "monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday".

Input

The input consists of two lines, each of them containing the name of exactly one day of the week. It's guaranteed that each string in the input is from the set "monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday".

Output

Print "YES" (without quotes) if such situation is possible during some non-leap year. Otherwise, print "NO" (without quotes).

Examples

input
```
monday
tuesday
```
output
```
NO
```

input
```
sunday
sunday
```
output
```
YES
```

input
```
saturday
tuesday
```
output
```
YES
```

Note

In the second sample, one can consider February 1 and March 1 of year 2015. Both these days were Sundays.

In the third sample, one can consider July 1 and August 1 of year 2017. First of these two days is Saturday, while the second one is Tuesday.

B. Batch Sort

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

You are given a table consisting of n rows and m columns.

Numbers in each row form a permutation of integers from 1 to m.

You are allowed to pick two elements in one row and swap them, but **no more than once** for each row. Also, **no more than once** you are allowed to pick two columns and swap them. Thus, you are allowed to perform from 0 to $n+1$ actions in total. **Operations can be performed in any order**.

You have to check whether it's possible to obtain the identity permutation $1, 2, ..., m$ in each row. In other words, check if one can perform some of the operation following the given rules and make each row sorted in increasing order.

Input

The first line of the input contains two integers n and m $(1 \le n, m \le 20)$ — the number of rows and the number of columns in the given table.

Each of next n lines contains m integers — elements of the table. It's guaranteed that numbers in each line form a permutation of integers from 1 to m.

Output

If there is a way to obtain the identity permutation in each row by following the given rules, print "`YES`" (without quotes) in the only line of the output. Otherwise, print "`NO`" (without quotes).

Examples

input

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

output

```
YES
```

input

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

output

```
NO
```

input

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

output

```
YES
```

Note

In the first sample, one can act in the following way:

1. Swap second and third columns. Now the table is
$$\begin{matrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{matrix}$$
2. In the second row, swap the second and the fourth elements. Now the table is
$$\begin{matrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{matrix}$$

C. Ray Tracing

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

There are k sensors located in the rectangular room of size $n \times m$ meters. The i-th sensor is located at point (x_i, y_i). All sensors are located at distinct points strictly inside the rectangle.

Opposite corners of the room are located at points $(0, 0)$ and (n, m). Walls of the room are parallel to coordinate axes.

At the moment 0, from the point $(0, 0)$ the laser ray is released in the direction of point $(1, 1)$. The ray travels with a speed of meters per second. Thus, the ray will reach the point $(1, 1)$ in exactly one second after the start.

When the ray meets the wall it's reflected by the rule that the angle of incidence is equal to the angle of reflection. If the ray reaches any of the four corners, it immediately stops.

For each sensor you have to determine the first moment of time when the ray will pass through the point where this sensor is located. If the ray will never pass through this point, print -1 for such sensors.

Input

The first line of the input contains three integers n, m and k ($2 \le n, m \le 100\,000$, $1 \le k \le 100\,000$) — lengths of the room's walls and the number of sensors.

Each of the following k lines contains two integers x_i and y_i ($1 \le x_i \le n - 1$, $1 \le y_i \le m - 1$) — coordinates of the sensors. It's guaranteed that no two sensors are located at the same point.

Output

Print k integers. The i-th of them should be equal to the number of seconds when the ray first passes through the point where the i-th sensor is located, or -1 if this will never happen.

Examples

input

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

output

```
1
-1
-1
2
```

input

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

output

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

input

```
7 4 5
1 3
2 2
5 1
5 3
4 3
```

output

```
13
2
9
5
-1
```

Note

In the first sample, the ray will consequently pass through the points $(0, 0)$, $(1, 1)$, $(2, 2)$, $(3, 3)$. Thus, it will stop at the point $(3, 3)$ after 3 seconds.

In the second sample, the ray will consequently pass through the following points: $(0, 0)$, $(1, 1)$, $(2, 2)$, $(3, 3)$, $(2, 4)$, $(1, 3)$, $(0, 2)$, $(1, 1)$, $(2, 0)$, $(3, 1)$, $(2, 2)$, $(1, 3)$, $(0, 4)$. The ray will stop at the point $(0, 4)$ after 12 seconds. It will reflect at the points $(3, 3)$, $(2, 4)$, $(0, 2)$, $(2, 0)$ and $(3, 1)$.

D. Dense Subsequence

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 lowercase English letters, and the integer m.

One should choose some symbols from the given string so that any contiguous subsegment of length m has at least one selected symbol. Note that here we choose positions of symbols, not the symbols themselves.

Then one uses the chosen symbols to form **a new string**. All symbols from the chosen position should be used, but we are allowed to rearrange them in any order.

Formally, we choose a subsequence of indices $1 \le i_1 < i_2 < ... < i_t \le |s|$. The selected sequence must meet the following condition: for every j such that $1 \le j \le |s| - m + 1$, there must be at least one selected index that belongs to the segment $[j, \ j + m - 1]$, i.e. there should exist a k from 1 to t, such that $j \le i_k \le j + m - 1$.

Then we take any permutation p of the selected indices and form a new string $s_{i_{p_1}} s_{i_{p_2}} ... \ s_{i_{p_t}}$.

Find the lexicographically smallest string, that can be obtained using this procedure.

Input

The first line of the input contains a single integer m ($1 \le m \le 100\,000$).

The second line contains the string s consisting of lowercase English letters. It is guaranteed that this string is non-empty and its length doesn't exceed $100\,000$. It is also guaranteed that the number m doesn't exceed the length of the string s.

Output

Print the single line containing the lexicographically smallest string, that can be obtained using the procedure described above.

Examples

input

```
3
cbabc
```

output

```
a
```

input

```
2
abcab
```

output

```
aab
```

input

```
3
bcabcbaccba
```

output

```
aaabb
```

Note

In the first sample, one can choose the subsequence $\{3\}$ and form a string "`a`".

In the second sample, one can choose the subsequence $\{1, 2, 4\}$ (symbols on this positions are '`a`', '`b`' and '`a`') and rearrange the chosen symbols to form a string "`aab`".

E. Goods transportation

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

There are n cities located along the one-way road. Cities are numbered from 1 to n in the direction of the road.

The i-th city had produced p_i units of goods. No more than s_i units of goods can be sold in the i-th city.

For each pair of cities i and j such that $\mathbf{1 \le i < j \le n}$ you can **no more than once** transport **no more than** c units of goods from the city i to the city j. Note that goods can only be transported from a city with a lesser index to the city with a larger index. **You can transport goods between cities in any order.**

Determine the maximum number of produced goods that can be sold in total in all the cities after a sequence of transportations.

Input

The first line of the input contains two integers n and c ($1 \le n \le 10\,000$, $0 \le c \le 10^9$) — the number of cities and the maximum amount of goods for a single transportation.

The second line contains n integers p_i ($0 \le p_i \le 10^9$) — the number of units of goods that were produced in each city.

The third line of input contains n integers s_i ($0 \le s_i \le 10^9$) — the number of units of goods that can be sold in each city.

Output

Print the maximum total number of produced goods that can be sold in all cities after a sequence of transportations.

Examples

input

```
3 0
1 2 3
3 2 1
```

output

```
4
```

input

```
5 1
7 4 2 1 0
1 2 3 4 5
```

output

```
12
```

input

```
4 3
13 10 7 4
4 7 10 13
```

output

```
34
```

F. Uniformly Branched Trees

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

A tree is a connected graph without cycles.

Two trees, consisting of n vertices each, are called *isomorphic* if there exists a permutation $p\colon \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that the edge (u, v) is present in the first tree if and only if the edge (p_u, p_v) is present in the second tree.

Vertex of the tree is called internal if its degree is greater than or equal to two.

Count the number of different non-isomorphic trees, consisting of n vertices, such that the degree of each internal vertex is **exactly** d. Print the answer over the given prime modulo mod.

Input

The single line of the input contains three integers n, d and mod $(1 \le n \le 1000,\ 2 \le d \le 10,\ 10^8 \le mod \le 10^9)$ — the number of vertices in the tree, the degree of internal vertices and the prime modulo.

Output

Print the number of trees over the modulo mod.

Examples

input
```
5 2 433416647
```
output
```
1
```

input
```
10 3 409693891
```
output
```
2
```

input
```
65 4 177545087
```
output
```
910726
```

G. Xor-matic Number of the Graph

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

You are given an undirected graph, consisting of n vertices and m edges. Each edge of the graph has some non-negative integer written on it.

Let's call a triple (u, v, s) **interesting**, if $1 \le u < v \le n$ and there is a path (**possibly non-simple**, i.e. it can visit the same vertices and edges multiple times) between vertices u and v such that xor of all numbers written on the edges of this path is equal to s. **When we compute the value s for some path, each edge is counted in xor as many times, as it appear on this path.** It's not hard to prove that there are finite number of such triples.

Calculate the sum over modulo $10^9 + 7$ of the values of s over all **interesting** triples.

Input

The first line of the input contains two integers n and m ($1 \le n \le 100\,000$, $0 \le m \le 200\,000$) — numbers of vertices and edges in the given graph.

The follow m lines contain three integers u_i, v_i and t_i ($1 \le u_i, v_i \le n$, $0 \le t_i \le 10^{18}$, $u_i \ne v_i$) — vertices connected by the edge and integer written on it. It is guaranteed that graph doesn't contain self-loops and multiple edges.

Output

Print the single integer, equal to the described sum over modulo $10^9 + 7$.

Examples

input

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

output

```
12
```

input

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

output

```
90
```

input

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

output

```
62
```

Note

In the first example the are 6 interesting triples:

1. $(1, 2, 1)$
2. $(1, 3, 2)$
3. $(1, 4, 3)$
4. $(2, 3, 3)$
5. $(2, 4, 2)$
6. $(3, 4, 1)$

The sum is equal to $1 + 2 + 3 + 3 + 2 + 1 = 12$.
In the second example the are 12 interesting triples:

1. $(1, 2, 1)$
2. $(2, 3, 2)$
3. $(1, 3, 3)$
4. $(3, 4, 4)$
5. $(2, 4, 6)$

6. $(1, 4, 7)$
7. $(1, 4, 8)$
8. $(2, 4, 9)$
9. $(3, 4, 11)$
10. $(1, 3, 12)$
11. $(2, 3, 13)$
12. $(1, 2, 14)$

The sum is equal to $1 + 2 + 3 + 4 + 6 + 7 + 8 + 9 + 11 + 12 + 13 + 14 = 90$.