Codeforces Round #539 (Div. 1)

A. Sasha and a Bit of Relax

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

Sasha likes programming. Once, during a very long contest, Sasha decided that he was a bit tired and needed to relax. So he did. But since Sasha isn't an ordinary guy, he prefers to relax unusually. During leisure time Sasha likes to upsolve unsolved problems because upsolving is very useful.

Therefore, Sasha decided to upsolve the following problem:

You have an array a with n integers. You need to count the number of *funny* pairs (l, r) $(l \leq r)$. To check if a pair (l, r) is a *funny* pair, take $mid = \frac{l+r-1}{2}$, then if $r - l + 1$ is an **even** number and $a_l \oplus a_{l+1} \oplus \ldots \oplus a_{mid} = a_{mid+1} \oplus a_{mid+2} \oplus \ldots \oplus a_r$, then the pair is *funny*. In other words, $\oplus$ of elements of the left half of the subarray from l to r should be equal to $\oplus$ of elements of the right half. Note that $\oplus$ denotes the bitwise XOR operation.

It is time to continue solving the contest, so Sasha asked you to solve this task.

Input

The first line contains one integer n ($2 \leq n \leq 3 \cdot 10^5$) — the size of the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i < 2^{20}$) — array itself.

Output

Print one integer — the number of *funny* pairs. You should consider only pairs where $r - l + 1$ is even number.

Examples

input
```
5
1 2 3 4 5
```
output
```
1
```

input
```
6
3 2 2 3 7 6
```
output
```
3
```

input
```
3
42 4 2
```
output
```
0
```

Note

Be as cool as Sasha, upsolve problems!

In the first example, the only *funny* pair is $(2, 5)$, as $2 \oplus 3 = 4 \oplus 5 = 1$.

In the second example, funny pairs are $(2, 3)$, $(1, 4)$, and $(3, 6)$.

In the third example, there are no *funny* pairs.

B. Sasha and One More Name

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

Reading books is one of Sasha's passions. Once while he was reading one book, he became acquainted with an unusual character.

The character told about himself like that: "Many are my names in many countries. Mithrandir among the Elves, Tharkûn to the Dwarves, Olórin I was in my youth in the West that is forgotten, in the South Incánus, in the North Gandalf; to the East I go not."

And at that moment Sasha thought, how would that character be called in the East? In the East all names are palindromes. A string is a palindrome if it reads the same backward as forward. For example, such strings as "`kazak`", "`oo`" and "`r`" are palindromes, but strings "`abb`" and "`ij`" are not.

Sasha believed that the hero would be named after one of the gods of the East. As long as there couldn't be two equal names, so in the East people did the following: they wrote the original name as a string on a piece of paper, then cut the paper minimum number of times k, so they got $k+1$ pieces of paper with substrings of the initial string, and then unite those pieces together to get a new string. Pieces **couldn't be turned over**, they could be shuffled.

In this way, it's possible to achive a string `abcdefg` from the string `f|de|abc|g` using 3 cuts (by swapping papers with substrings `f` and `abc`). The string `cbadefg` can't be received using the same cuts.

More formally, Sasha wants for the given **palindrome** s find such minimum k, that you can cut this string into $k+1$ parts, and then unite them in such a way that the final string will be a palindrome and it won't be equal to the initial string s. It there is no answer, then print "`Impossible`" (without quotes).

Input

The first line contains one string s $(1 \le |s| \le 5\,000)$ — the initial name, which consists only of lowercase Latin letters. It is guaranteed that s is a palindrome.

Output

Print one integer k — the minimum number of cuts needed to get a new name, or "`Impossible`" (without quotes).

Examples

input
```
nolon
```
output
```
2
```

input
```
otto
```
output
```
1
```

input
```
qqqq
```
output
```
Impossible
```

input
```
kinnikkinnik
```
output
```
1
```

Note

In the first example, you can cut the string in those positions: `no|l|on`, and then unite them as follows `on|l|no`. It can be shown that there is no solution with one cut.

In the second example, you can cut the string right in the middle, and swap peaces, so you get `toot`.

In the third example, you can't make a string, that won't be equal to the initial one.

In the fourth example, you can cut the suffix `nik` and add it to the beginning, so you get `nikkinnikkin`.

C. Sasha and a Patient Friend

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Fedya and Sasha are friends, that's why Sasha knows everything about Fedya.

Fedya keeps his patience in an infinitely large bowl. But, unlike the bowl, Fedya's patience isn't infinite, that is why let v be the number of liters of Fedya's patience, and, as soon as v becomes **equal** to 0, the bowl will burst immediately. There is one tap in the bowl which pumps s liters of patience per second. Notice that s can be negative, in that case, the tap pumps out the patience. Sasha can do different things, so he is able to change the tap's speed. All actions that Sasha does can be represented as q queries. There are three types of queries:

1. `"1 t s"` — add a new event, means that starting from the t-th second the tap's speed will be equal to s.
2. `"2 t"` — delete the event which happens at the t-th second. It is guaranteed that such event exists.
3. `"3 l r v"` — Sasha wonders: if you take all the events for which $l \le t \le r$ and simulate changes of Fedya's patience from the very beginning of the l-th second till the very beginning of the r-th second inclusive (the initial volume of patience, at the beginning of the l-th second, equals to v liters) then when will be the moment when the bowl will burst. If that does not happen, then the answer will be -1.

Since Sasha does not want to check what will happen when Fedya's patience ends, and he has already come up with the queries, he is asking you to help him and find the answer for each query of the 3-rd type.

It is guaranteed that at any moment of time, there won't be two events which happen at the same second.

Input

The first line contans one integer q ($1 \le q \le 10^5$) — the number of queries.

Each of the next q lines have one of the following formats:

- `1 t s` ($1 \le t \le 10^9$, $-10^9 \le s \le 10^9$), means that a new event is added, which means that starting from the t-th second the tap's speed will be equal to s.
- `2 t` ($1 \le t \le 10^9$), means that the event which happens at the t-th second must be deleted. Guaranteed that such exists.
- `3 l r v` ($1 \le l \le r \le 10^9$, $0 \le v \le 10^9$), means that you should simulate the process from the very beginning of the l-th second till the very beginning of the r-th second inclusive, and to say when will the bowl burst.

It is guaranteed that t, s, l, r, v in all the queries are integers.

Also, it is guaranteed that there is at least one query of the 3-rd type, and there won't be a query of the 1-st type with such t, that there already exists an event which happens at that second t.

Output

For each query of the 3-rd type, print in a new line the moment when the bowl will burst or print -1 if it won't happen.

Your answer will be considered correct if it's absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

input

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

output

```
5
5.666667
6
-1
```

input

```
10
1 2 2
1 4 4
1 7 -10
3 2 4 1
3 5 6 0
3 1 15 1
2 4
3 1 15 1
1 8 1
3 1 15 1
```

output

```
-1
5
8.7
8.1
-1
```

input

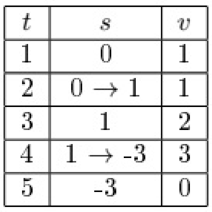
```
5
1 1000 9999999
1 2000 -9999
3 1000 2000 0
2 1000
3 1000 2002 1
```

output

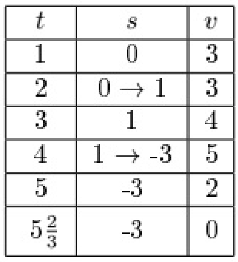
```
1000
2000.0001
```

Note

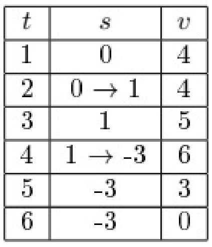
In the first example all the queries of the 3-rd type cover all the events, it's simulation is following:



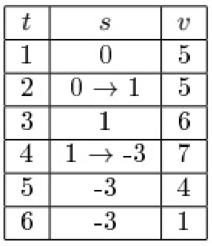
t	s	v
1	0	1
2	$0 \to 1$	1
3	1	2
4	$1 \to$ -3	3
5	-3	0



t	s	v
1	0	3
2	$0 \to 1$	3
3	1	4
4	$1 \to$ -3	5
5	-3	2
$5\frac{2}{3}$	-3	0



t	s	v
1	0	4
2	$0 \to 1$	4
3	1	5
4	$1 \to$ -3	6
5	-3	3
6	-3	0



t	s	v
1	0	5
2	$0 \to 1$	5
3	1	6
4	$1 \to$ -3	7
5	-3	4
6	-3	1

D. Sasha and Interesting Fact from Graph Theory

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Once, during a lesson, Sasha got bored and decided to talk with his friends. Suddenly, he saw Kefa. Since we can talk endlessly about Kefa, we won't even start doing that. The conversation turned to graphs. Kefa promised Sasha to tell him about one interesting fact from graph theory if Sasha helps Kefa to count the number of *beautiful trees*.

In this task, a *tree* is a weighted connected graph, consisting of n vertices and $n-1$ edges, and weights of edges are integers from 1 to m. Kefa determines the beauty of a tree as follows: he finds in the tree his two favorite vertices — vertices with numbers a and b, and counts the distance between them. The distance between two vertices x and y is the sum of weights of edges on the simple path from x to y. If the distance between two vertices a and b is **equal** to m, then the tree is *beautiful*.

Sasha likes graph theory, and even more, Sasha likes interesting facts, that's why he agreed to help Kefa. Luckily, Sasha is familiar with you `the best programmer in Byteland`. Help Sasha to count the number of *beautiful* trees for Kefa. Two trees are considered to be distinct if there is an edge that occurs in one of them and doesn't occur in the other one. Edge's **weight matters**.

Kefa warned Sasha, that there can be too many beautiful trees, so it will be enough to count the number modulo 10^9+7.

Input

The first line contains four integers n, m, a, b ($2 \le n \le 10^6$, $1 \le m \le 10^6$, $1 \le a, b \le n$, $a \ne b$) — the number of vertices in the tree, the maximum weight of an edge and two Kefa's favorite vertices.

Output

Print one integer — the number of *beautiful trees* modulo 10^9+7.

Examples

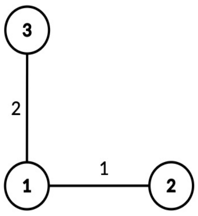
input
```
3 2 1 3
```
output
```
5
```

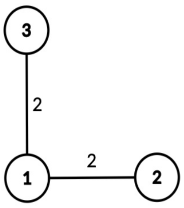
input
```
3 1 1 2
```
output
```
2
```

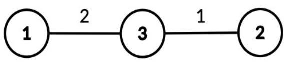
input
```
5 15 1 5
```
output
```
345444
```

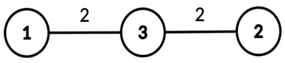
Note

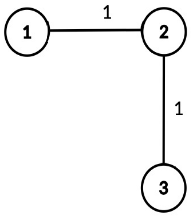
There are 5 *beautiful trees* in the first example:



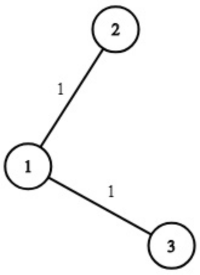


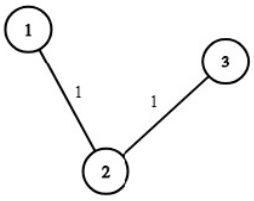






In the second example the following trees are *beautiful*:





E. Sasha and a Very Easy Test

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Egor likes math, and not so long ago he got the highest degree of recognition in the math community — Egor became a *red* mathematician. In this regard, Sasha decided to congratulate Egor and give him a math test as a present. This test contains an array a of integers of length n and exactly q queries. Queries were of three types:

1. "`1 l r x`" — multiply each number on the range from l to r by x.
2. "`2 p x`" — divide the number at the position p by x (divisibility guaranteed).
3. "`3 l r`" — find the sum of all elements on the range from l to r.

The sum can be big, so Sasha asked Egor to calculate the sum modulo some integer mod.

But since Egor is a *red* mathematician, he doesn't have enough time to solve such easy tasks, at the same time he doesn't want to anger Sasha, that's why he asked you to help and to find answers for all queries of the 3-rd type.

Input

The first line contains two integers n and mod ($1 \le n \le 10^5$, $2 \le mod \le 10^9 + 9$) — the size of the array and the number mod.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^5$) — the array itself.

The third line contains one integer $q (1 \le q \le 10^5)$ — the number of queries.

Next q lines satisfy one of the following formats:

- `1 l r x` ($1 \le l \le r \le n$, $1 \le x \le 10^5$), means that you must multiply each number on the range from l to r by x.
- `2 p x` ($1 \le p \le n$, $1 \le x \le 10^5$), means that you must divide number at the position p by x (divisibility guaranteed).
- `3 l r` ($1 \le l \le r \le n$), means that you must find the sum of elements on the range from l to r.

It is guaranteed that there is at least one query of the 3-rd type.

Output

For each query of the 3-rd type print the answer on a new line modulo mod.

Examples

input

```
5 100
4 1 2 3 5
5
3 1 5
1 2 3 6
3 1 2
1 1 5 1
3 2 4
```

output

```
15
10
21
```

input

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

output

```
1
0
1
0
```

input

```
5 2100
1 2 3 4 5
10
1 1 3 12
1 1 5 10
2 5 50
3 2 4
1 4 4 28
2 4 7
3 1 2
3 3 4
2 3 3
3 1 5
```

output

```
640
360
520
641
```

Note

The first example:

Inital array is $[4, 1, 2, 3, 5]$

- In the first query, you must calculate the sum of the whole array, it's equal to $(4 + 1 + 2 + 3 + 5) \bmod 100 = 15 \bmod 100 = 15$
- In the second query, you must multiply each number on the range from 2 to 3 by 6. The resulting array will be $[4, 6, 12, 3, 5]$
- In the third query, you must calculate the sum on the range from 1 to 2, it's equal to $(4 + 6) \bmod 100 = 10 \bmod 100 = 10$
- In the fourth query, you must multiply each number on the range from 1 to 5 by 1. Multiplication by 1 doesn't affect the array.
- In the fifth query, you must calculate the sum on the range from 2 to 4, it's equal to $(6 + 12 + 3) \bmod 100 = 21 \bmod 100 = 21$

The second example:

Inital array is $[4, 1, 2, 3, 5]$

- In the first query, you must calculate the sum of the whole array, it's equal to $(4 + 1 + 2 + 3 + 5) \bmod 2 = 15 \bmod 2 = 1$
- In the second query, you must multiply each number on the range from 2 to 3 by 6. The resulting array will be $[4, 6, 12, 3, 5]$
- In the third query, you must calculate the sum on the range from 1 to 2, it's equal to $(4 + 6) \bmod 2 = 10 \bmod 2 = 0$
- In the fourth query, you must multiply each number on the range from 1 to 5 by 1. Multiplication by 1 doesn't affect the array.
- In the fifth query, you must calculate the sum on the range from 2 to 4, it's equal to $(6 + 12 + 3) \bmod 2 = 21 \bmod 2 = 1$
- In the sixth query, you must divide number at the position 3 by 4. $\frac{12}{4} = 3$, so the array will be $[4, 6, 3, 3, 5]$.
- In the seventh, query you must calculate the sum on the range form 3 to 4, it's equal to $(3 + 3) \bmod 2 = 6 \bmod 2 = 0$

F. Sasha and Algorithm of Silence's Sounds

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One fine day Sasha went to the park for a walk. In the park, he saw that his favorite bench is occupied, and he had to sit down on the neighboring one. He sat down and began to listen to the silence. Suddenly, he got a question: what if in different parts of the park, the silence sounds in different ways? So it was. Let's divide the park into 1×1 meter squares and call them *cells*, and numerate rows from 1 to n from up to down, and columns from 1 to m from left to right. And now, every cell can be described with a pair of two integers (x, y), where x — the number of the row, and y — the number of the column. Sasha knows that the level of silence in the cell (i, j) equals to $f_{i,j}$, and all $f_{i,j}$ form a permutation of numbers from 1 to $n \cdot m$. Sasha decided to count, how many are there *pleasant* segments of silence?

Let's take some segment $[l \dots r]$. Denote S as the set of cells (i, j) that $l \le f_{i,j} \le r$. Then, the segment of silence $[l \dots r]$ is *pleasant* if there is only **one simple** path between every pair of cells from S (path can't contain cells, which are not in S). In other words, set S should look like a tree on a plain. Sasha has done this task pretty quickly, and called the algorithm — "algorithm of silence's sounds".

Time passed, and the only thing left from the algorithm is a legend. To prove the truthfulness of this story, you have to help Sasha and to find the number of **different** *pleasant* segments of silence. Two segments $[l_1 \dots r_1]$, $[l_2 \dots r_2]$ are different, if $l_1 \neq l_2$ or $r_1 \neq r_2$ or both at the same time.

Input

The first line contains two integers n and m ($1 \le n, m \le 1000$, $1 \le n \cdot m \le 2 \cdot 10^5$) — the size of the park.

Each from next n lines contains m integers $f_{i,j}$ ($1 \le f_{i,j} \le n \cdot m$) — the level of silence in the cell with number (i, j).

It is guaranteed, that all $f_{i,j}$ are different.

Output

Print one integer — the number of *pleasant* segments of silence.

Examples

input

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

output

```
15
```

input

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

output

```
15
```

input

```
4 4
4 3 2 16
1 13 14 15
5 7 8 12
6 11 9 10
```

output

```
50
```

Note

In the first example, all segments of silence are *pleasant*.

In the second example, *pleasant* segments of silence are the following: