Codeforces Beta Round #88

A. Elevator

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

And now the numerous qualifying tournaments for one of the most prestigious Russian contests *Russian Codec Cup* are over. All n participants who have made it to the finals found themselves in a huge m-floored 10^8-star hotel. Of course the first thought to come in a place like this is "How about checking out the elevator?".

The hotel's elevator moves between floors according to one never changing scheme. Initially (at the moment of time 0) the elevator is located on the 1-st floor, then it moves to the 2-nd floor, then — to the 3-rd floor and so on until it reaches the m-th floor. After that the elevator moves to floor m - 1, then to floor m - 2, and so on until it reaches the first floor. This process is repeated infinitely. We know that the elevator has infinite capacity; we also know that on every floor people get on the elevator immediately. Moving between the floors takes a unit of time.

For each of the n participant you are given s_i, which represents the floor where the i-th participant starts, f_i, which represents the floor the i-th participant wants to reach, and t_i, which represents the time when the i-th participant starts on the floor s_i.

For each participant print the minimum time of his/her arrival to the floor f_i.

If the elevator stops on the floor s_i at the time t_i, then the i-th participant can enter the elevator immediately. If the participant starts on the floor s_i and that's the floor he wanted to reach initially ($s_i = f_i$), then the time of arrival to the floor f_i for this participant is considered equal to t_i.

Input

The first line contains two space-separated integers n and m ($1 \le n \le 10^5, 2 \le m \le 10^8$).

Next n lines contain information about the participants in the form of three space-separated integers $s_i\ f_i\ t_i$ ($1 \le s_i, f_i \le m, 0 \le t_i \le 10^8$), described in the problem statement.

Output

Print n lines each containing one integer — the time of the arrival for each participant to the required floor.

Sample test(s)

input

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

output

```
9
1
0
7
10
7
5
```

input

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

output

```
12
10
10
8
7
```

Note

Let's consider the first sample. The first participant starts at floor $s = 2$ by the time equal to $t = 3$. To get to the floor $f = 4$, he has to wait until the time equals 7, that's the time when the elevator will go upwards for the second time. Then the first participant should get on the elevator and go two

floors up. In this case the first participant gets to the floor f at time equal to 9. The second participant starts at the time $t = 0$ on the floor $s = 1$, enters the elevator immediately, and arrives to the floor $f = 2$. The third participant doesn't wait for the elevator, because he needs to arrive to the same floor where he starts.

B. Very Interesting Game

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

In a very ancient country the following game was popular. Two people play the game. Initially first player writes a string s_1, consisting of exactly nine digits and representing a number that does not exceed a. After that second player looks at s_1 and writes a string s_2, consisting of exactly nine digits and representing a number that does not exceed b. Here a and b are some given constants, s_1 and s_2 are chosen by the players. The strings are allowed to contain leading zeroes.

If a number obtained by the concatenation (joining together) of strings s_1 and s_2 is divisible by mod, then the second player wins. Otherwise the first player wins. You are given numbers a, b, mod. Your task is to determine who wins if both players play in the optimal manner. If the first player wins, you are also required to find the lexicographically minimum winning move.

Input

The first line contains three integers a, b, mod ($0 \le a, b \le 10^9$, $1 \le mod \le 10^7$).

Output

If the first player wins, print "1" and the lexicographically minimum string s_1 he has to write to win. If the second player wins, print the single number "2".

Sample test(s)

input
1 10 7
output
2

input
4 0 9
output
1 000000001

Note

The lexical comparison of strings is performed by the < operator in modern programming languages. String x is lexicographically less than string y if exists such i ($1 \le i \le 9$), that $x_i < y_i$, and for any j ($1 \le j < i$) $x_j = y_j$. These strings always have length 9.

C. Cycle

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

A *tournament* is a directed graph without self-loops in which every pair of vertexes is connected by exactly one directed edge. That is, for any two vertexes u and v ($u \neq v$) exists either an edge going from u to v, or an edge from v to u.

You are given a tournament consisting of n vertexes. Your task is to find there a cycle of length three.

Input

The first line contains an integer n ($1 \leq n \leq 5000$). Next n lines contain the adjacency matrix A of the graph (without spaces). $A_{i,j} = 1$ if the graph has an edge going from vertex i to vertex j, otherwise $A_{i,j} = 0$. $A_{i,j}$ stands for the j-th character in the i-th line.

It is guaranteed that the given graph is a tournament, that is, $A_{i,i} = 0$, $A_{i,j} \neq A_{j,i}$ ($1 \leq i, j \leq n, i \neq j$).

Output

Print three distinct vertexes of the graph a_1, a_2, a_3 ($1 \leq a_i \leq n$), such that $A_{a_1,a_2} = A_{a_2,a_3} = A_{a_3,a_1} = 1$, or "-1", if a cycle whose length equals three does not exist.

If there are several solutions, print any of them.

Sample test(s)

input
5 00100 10000 01001 11101 11000
output
1 3 2

input
5 01111 00000 01000 01100 01110
output
-1

D. Not Quick Transformation

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let a be an array consisting of n numbers. The array's elements are numbered from 1 to n, $even$ is an array consisting of the numerals whose numbers are even in a ($even_i = a_{2i}$, $1 \le 2i \le n$), odd is an array consisting of the numberals whose numbers are odd in a ($odd_i = a_{2i-1}$, $1 \le 2i - 1 \le n$). Then let's define the transformation of array $F(a)$ in the following manner:

- if $n > 1$, $F(a) = F(odd) + F(even)$, where operation " + " stands for the arrays' concatenation (joining together)
- if $n = 1$, $F(a) = a$

Let a be an array consisting of n numbers $1, 2, 3, ..., n$. Then b is the result of applying the transformation to the array a (so $b = F(a)$). You are given m queries (l, r, u, v). Your task is to find for each query the sum of numbers b_i, such that $l \le i \le r$ and $u \le b_i \le v$. You should print the query results modulo mod.

Input

The first line contains three integers n, m, mod ($1 \le n \le 10^{18}$, $1 \le m \le 10^5$, $1 \le mod \le 10^9$). Next m lines describe the queries. Each query is defined by four integers l, r, u, v ($1 \le l \le r \le n$, $1 \le u \le v \le 10^{18}$).

Please do not use the `%lld` specificator to read or write 64-bit integers in C++. Use `%I64d` specificator.

Output

Print m lines each containing an integer — remainder modulo mod of the query result.

Sample test(s)

input

```
4 5 10000
2 3 4 5
2 4 1 3
1 2 2 4
2 3 3 5
1 3 3 4
```

output

```
0
5
3
3
3
```

input

```
2 5 10000
1 2 2 2
1 1 4 5
1 1 2 5
1 1 1 3
1 2 5 5
```

output

```
2
0
0
1
0
```

Note

Let's consider the first example. First let's construct an array $b = F(a) = F([1, 2, 3, 4])$.

- Step 1. $F([1, 2, 3, 4]) = F([1, 3]) + F([2, 4])$
- Step 2. $F([1, 3]) = F([1]) + F([3]) = [1] + [3] = [1, 3]$
- Step 3. $F([2, 4]) = F([2]) + F([4]) = [2] + [4] = [2, 4]$
- Step 4. $b = F([1, 2, 3, 4]) = F([1, 3]) + F([2, 4]) = [1, 3] + [2, 4] = [1, 3, 2, 4]$

Thus $b = [1, 3, 2, 4]$. Let's consider the first query $l = 2$, $r = 3$, $u = 4$, $v = 5$. The second and third positions in the array b do not have numbers in the range $[4, 5]$, so the sum obviously equals zero. Let's consider the second query $l = 2$, $r = 4$, $u = 1$, $v = 3$. The second and third positions have two numbers that belong to the range $[1, 3]$, their sum equals 5.

E. Tree or not Tree

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected connected graph G consisting of n vertexes and n edges. G contains no self-loops or multiple edges. Let each edge has two states: on and off. Initially all edges are switched off.

You are also given m queries represented as (v, u) — change the state of all edges on the shortest path from vertex v to vertex u in graph G. If there are several such paths, the lexicographically minimal one is chosen. More formally, let us consider all shortest paths from vertex v to vertex u as the sequences of vertexes $v, v_1, v_2, ..., u$. Among such sequences we choose the lexicographically minimal one.

After each query you should tell how many connected components has the graph whose vertexes coincide with the vertexes of graph G and edges coincide with the switched on edges of graph G.

Input

The first line contains two integers n and m ($3 \le n \le 10^5$, $1 \le m \le 10^5$). Then n lines describe the graph edges as $a\ b$ ($1 \le a, b \le n$). Next m lines contain the queries as $v\ u$ ($1 \le v, u \le n$).

It is guaranteed that the graph is connected, does not have any self-loops or multiple edges.

Output

Print m lines, each containing one integer — the query results.

Sample test(s)

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

output
```
3
3
```

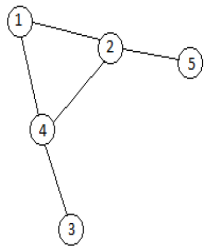
input
```
6 2
4 6
4 3
1 2
6 5
1 5
1 4
2 5
2 6
```

output
```
4
3
```

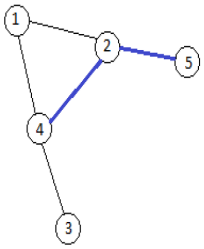
Note

Let's consider the first sample. We'll highlight the switched on edges blue on the image.

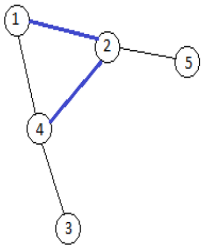
- The graph before applying any operations. No graph edges are switched on, that's why there initially are 5 connected components.



- The graph after query $v = 5, u = 4$. We can see that the graph has three components if we only consider the switched on edges.



- The graph after query $v = 1, u = 5$. We can see that the graph has three components if we only consider the switched on edges.



Lexicographical comparison of two sequences of equal length of k numbers should be done as follows. Sequence x is lexicographically less than sequence y if exists such i ($1 \le i \le k$), so that $x_i < y_i$, and for any j ($1 \le j < i$) $x_j = y_j$.
