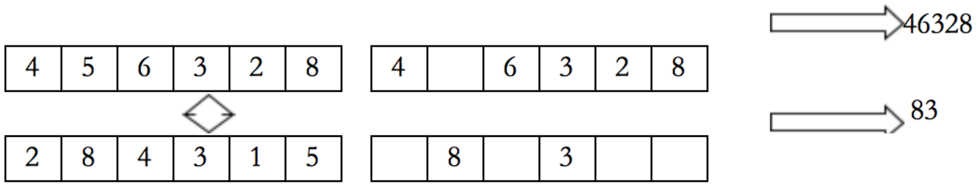
TASK	YODA	HAN	DEATHSTAR	CHEWBACCA	GALAKSIJA	ENDOR
input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	2 seconds	1 second	1 second	1 second	1 second
memory limit	64 MB	64 MB	64 MB	64 MB	64 MB	64 MB
score	**50**	**80**	**100**	**120**	**140**	**160**
	total 650					

A long, long time ago in a galaxy far, far away a big collision of integers is taking place right now. What happens when two integers **collide**? During collision, each digit of one number **compares itself** to the corresponding digit of the other number (the least significant digit with the other's least significant digit, and so on). The **smaller** digit "**falls out**" of the number containing it. Additionally, if the digits are the **same**, nothing happens. If a number **doesn't** consist of a corresponding digit, then we consider it to be **zero**. After all comparisons of corresponding digits, the leftover digits in the number **come closer and create** a new number. For example:



Write a programme that will, for two given integers, determine their **values after collision**. If it happens that all the digits of one number fell out, then for that number output the message "YODA".

INPUT

The first line of input contains the integer N ($1 \leqslant N \leqslant 10^9$), one of the integers from the task.
The second line of input contains the integer M ($1 \leqslant N \leqslant 10^9$), one of the integers from the task.

OUTPUT

The first line of output must contain the new value of the first given integer from the task.
The second line of output must contain the new value of the second given integer from the task.

SCORING

In test cases worth 30% points it will hold that N and M consist of three digits.

SAMPLE TESTS

input	**input**	**input**
300 500	65743 9651	2341 6785
output	**output**	**output**
0 500	673 95	YODA 6785

Han didn't want to study solo so he invited his friend Dominik to come over. After an eventful evening that will be remembered for a record number of solved tasks from the field of electronics, Dominik went home. To his surprise, the police stopped him thinking he was **drunk**. It is known that in these situations sobriety is proven by solving a series of carefully crafted tasks that test a man's cognitive abilities. If we can trust Dominik, the conversation went something like this:

Policeman: Something easy to begin with... What is the complexity of bubble sort?
Dominik: That is really easy, $O(n^2)$.
Policeman: Say the English alphabet in reverse.
Dominik: Trivial, zyxwvutsrqponmlkjihgfedcba
Policeman: You learned that by heart. Now imagine that all the letters of the English alphabet from 'a' to 'z' are respectively written clockwise in a circle. Begin with the letter 'a' and say the letters clockwise. After each spoken letter, I can tell you to continue saying the alphabet in reverse order or I can ask you how many times so far you've said a certain letter. Are you ready? 3, 2, 1, Go!
Dominik: Um... a, b, c...

Write a programme that solves Dominik's problem.

INPUT

The first line of input contains the integer Q ($1 \leqslant Q \leqslant 100\,000$), the number of policeman's orders. Each of the following Q lines contains one of the policeman's order in the form of "**SMJER n**" (Croatian for direction) or "**UPIT n x**" (Croatian for query). The order in the form "**SMJER n**" means that, after the nth spoken letter, Dominik must start saying the alphabet in reverse, whereas the order in the form "**UPIT n x**" means that Dominik must say how many times so far he's said the letter x in the first n spoken letters.

The policeman's order will be given chronologically in the input, or, the numbers n ($1 \leqslant n \leqslant 10^9$) from the orders will be strictly ascending. The character x from the order in the form of "**UPIT n x**" is a lowercase letter of the English alphabet.

OUTPUT

For each order in the form of "**UPIT n x**", output how many times Dominik has said the letter x in the first n spoken letters. The answer to each query needs to be written in a separate line, and the queries need to be answered in the order given in the input.

SCORING

In test cases worth 40% of total points, it will additionally hold: $N \leqslant 1000$.
In test cases worth 60% of total points, it will additionally hold: $N \leqslant 10^5$.

SAMPLE TESTS

input

```
5
UPIT 1 b
UPIT 3 b
SMJER 4
UPIT 7 a
UPIT 10 z
```

output

```
0
1
2
1
```

input

```
5
SMJER 1
SMJER 2
SMJER 3
UPIT 5 a
UPIT 7 w
```

output

```
2
1
```

input

```
4
UPIT 100 a
UPIT 200 c
UPIT 300 a
UPIT 400 b
```

output

```
4
8
12
16
```

Clarification of the first example: Dominik says the following letters: a, b, c, d, c, b, a, z, y, x.

Clarification of the second example: Dominik says the following letters: a, z, a, z, y, x, w.

Young jedi Ivan has infiltrated in The Death Star and his task is to destroy it. In order to destroy The Death Star, he needs an array of non-negative integers a_i of length N that represents the code for initiating the self-destruction of The Death Star. Ivan doesn't have the array, but he has a piece of paper with requirements for that array, given to him by his good old friend Darth Vader.

On the paper, a square matrix of the size N is written down. In that matrix m in the ith row and jth column there is a number that is equal to **bitwise and** between numbers a_i and a_j. Unfortunately, a lightsaber has destroyed all the fields on the matrix's main diagonal and Ivan cannot read what is on these fields. Help Ivan to reconstruct an array for the self-destruction of The Death Star that meets the requirements of the matrix.

The solution doesn't need to be unique, but will always **exist**.

INPUT

The first line of input contains the integer N ($1 \leqslant N \leqslant 1\,000$), size of the matrix.
Each of the following N lines contains N numbers m_{ij} ($1 \leqslant m_{ij} \leqslant 10^9$), the elements of the matrix.

OUTPUT

The first and only line of output must contain any array of N non-negative integers less than 10^9 that meet the requirements from the task.

SAMPLE TESTS

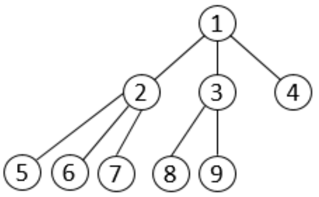
input	**input**

3
0 1 1
1 0 1
1 1 0
	5
	0 0 1 1 1
	0 0 2 0 2
	1 2 0 1 3
	1 0 1 0 1
	1 2 3 1 0

output	**output**
1 1 1	1 2 3 1 11

Clarification of the first example: It is clear that one of the arrays that meets the requirements from the matrix is [1 1 1]. Notice that this is not the only possible solution.

You are given a tree of order K with N nodes or, in other words, each node can have at most K children. The tree is constructed so it's of the "lowest energy": the nodes are placed in a new depth of the tree only when all the places (from left to right) in the previous depth have been filled. This is also the order of enumerating the nodes, starting with 1. Image depicts an example of a tree of order 3 with 9 nodes.



You need to output answers to Q queries in the form of x y, where the answer is the minimal number of steps needed to get from node x to node y.

INPUT

The first line of input contains the integers N ($1 \leqslant N \leqslant 10^{15}$), K ($1 \leqslant K \leqslant 1\,000$) i Q ($1 \leqslant Q \leqslant 100\,000$) from the task.
Each of the following Q lines contains pairs xy ($1 \leqslant x, y \leqslant N, x \neq y$) from the task.

OUTPUT

Output Q lines, each line containing the answer to a query from the task.

SCORING

In test cases worth 20% of total points, it will hold $1 \leqslant N, Q \leqslant 1\,000$.
In test cases worth 50% of total points, it will hold $1 \leqslant N, Q \leqslant 100\,000$.

SAMPLE TESTS

input 7 2 3 1 2 2 1 4 7 **output** 1 1 4	**input** 9 3 3 8 9 5 7 8 4 **output** 2 2 3

Clarification of the first example: You are given a complete binary tree. Node 2 is a direct child of node 1 so the distance between them is exactly 1. Nodes 4 and 7 are on complete opposite sides of the tree, so the distance between them is 4: 4 → 2 → 1 → 3 → 7.
Clarification of the second example: This example corresponds to the image.

A long time ago in a galaxy far, far away, there were N planets. There were also $N - 1$ interplanetary paths that connected all the planets (directly or indirectly). In other words, the network of planets and paths formed a tree. Additionally, each path was enumerated with an integer that denoted the **curiosity** of the path.

A pair of planets A, B is boring if the following holds:

- A and B are different planets
- travelling between planet A and B is possible using one or more interplanetary paths
- binary **XOR** of the curiosity of all the paths in that travel is equal to 0

Alas, the times have changed and an evil emperor is ruling the galaxy. He decided to use the Force to destroy all the interplanetary paths in a **certain order**.
Determine the number of boring pairs of planets before the emperor started the destruction and after each destruction.

INPUT

The first line of input contains the integer N ($1 \leqslant N \leqslant 100\,000$).
Each of the following $N - 1$ line contains three integers A_i, B_i, Z_i ($1 \leqslant A_i, B_i \leqslant N$, $0 \leqslant Z_i \leqslant 1\,000\,000\,000$) that denote that planets A_i and B_i are directly connected with a path of curiosity Z_i.
The following line of input contains the permutation of the first $N - 1$ integers that denote the order in which the emperor is destroying the paths. If the i^{th} element of the permutation is j, then the emperor destroyed the path between planets A_j and B_j in the i^{th} step.

OUTPUT

The output must contain N lines, the k^{th} line containing the number of boring pairs A, B from the task after the emperor destroyed exactly $k - 1$ paths.

SCORING

In test cases worth 20% of total points, it will hold $N \leqslant 1\,000$.
In test cases worth at least 30% of total points, every path's curiosity will be equal to 0.

SAMPLE TESTS

input	input	input

2
1 2 0
1
	3
	1 2 4
	2 3 4
	1 2
		4
		1 2 0
		2 3 0
		2 4 0
		3 1 2

output	**output**	**output**
1 0	1 0 0	6 3 1 0

Clarification of the first example: Before the destruction, the path between planets 1 and 2 is boring. After destruction, the path between them doesn't exist anymore.
Clarification of the second example: Before the destruction, pair of planets (1, 3) is boring. Travel between 1 and 3 is no longer possible after the first and after the second destruction, and none of the remaining pairs of planets is boring.
Clarification of the third example: Notice that in this example each pair of planets with a possible path between them is boring because all paths have the curiosity 0.

On the forest-filled moon of Endor there is, if we are to believe the Guinness Book of Records, the longest stick in the whole galaxy. On that stick of L meters in length there are N cheerful chameleons. Each chameleon is moving along the stick with constant speed of $1\,m/s$ in one of two possible directions (left or right) and is colored in one of the possible K colors.

It is known that the chameleons of Endor worship the ancient **ant** laws that dictate that the walk along the stick must be continued until the end of the stick is reached (which means getting off it), and when a collision with another chameleon takes place, you must turn **180 degrees** and continue the walk in the opposite direction. Additionally, after a chameleon colored in a moving to the left collides with a chameleon colored in b moving to the right, the chameleon moving to the left before the collision takes the color of the chameleon moving to the right before the collision (so, color b), while the chameleon moving to the right before the collision takes a new color $(a + b) \bmod K$.

If you are given the initial positions, colors and directions of movement of all the chameleons, for each color determine the total trip taken by the chameleons in that color before getting off the stick.

INPUT

The first line of input contains the integers N, K and L ($1 \leqslant N \leqslant 100\,000, 1 \leqslant K \leqslant 40, 1 \leqslant L \leqslant 1\,000\,000$) from the task. The i^{th} of the following N lines contains the integer d_i ($0 \leqslant d_i \leqslant L$) that denotes the distance between the i^{th} chameleon and the left end of the stick, then the integer b_i ($0 \leqslant b_i \leqslant K - 1$) that denotes the color of the i^{th} chameleon and the character 'L' (left) or 'D' (right) that denote the direction of movement of the i^{th} chameleon. All integers d_i will be unique and given in strictly ascending order.

OUTPUT

The output must contain K lines, the i^{th} line containing the total trip taken by the chameleons in color i.

SCORING

In test cases worth 50% of total points, it will additionally hold ($1 \leqslant N \leqslant 3\,000$).

SAMPLE TESTS

input
`2 3 10`
`0 0 D`
`10 1 L`
	input
	`4 3 7`
	`1 0 D`
	`3 0 D`
	`4 1 L`
	`6 2 D`
		input
		`4 4 5`
		`1 1 D`
		`3 3 L`
		`4 2 D`
		`5 0 L`

output
`10.0`
`10.0`
`0.0`
	output
	`10.0`
	`4.0`
	`1.0`
		output
		`2.5`
		`4.0`
		`2.5`
		`4.0`

Clarification of the first example: The chameleons collide after 5 travelled meters in the middle of the stick. After that, both chameleons change their direction of movement. The chameleon moving to the right after collision is colored in 0, whereas the chameleon moving to the left after collision is colored in 1.