TASK	LOZINKA	OBILAZAK	EKSPLOZIJA	DOMINE	TROKUTI	LADICE
source code	lozinka.pas lozinka.c lozinka.cpp	obilazak.pas obilazak.c obilazak.cpp	eksplozija.pas eksplozija.c eksplozija.cpp	domine.pas domine.c domine.cpp	trokuti.pas trokuti.c trokuti.cpp	ladice.pas ladice.c ladice.cpp
input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	1 second	1 second	1 second	1 second	1 second
memory limit	32 MB	32 MB	32 MB	64 MB	32 MB	32 MB
point value	**50**	**80**	**100**	**120**	**140**	**160**
	650					

Problems translated from Croatian by: **Paula Gombar**

Mirko is an evil plotting genius and has gotten hold of a list of all possible passwords for a certain user account. The first thing he noticed was all the passwords are of **odd length**. Mirko assumes that the correct password is the one which can be found in **both the original and reverse order** in the list. For example, if the word "tulipan" would be the correct password, the word "napilut" has to also appear in the list. Given that both words are correct passwords, Mirko will try to use both, one at a time.

Help Mirko discover what the correct password is and output its length and central character.

INPUT

The first line of input contains the integer **N** (1 ≤ **N** ≤ 100), the number of possible passwords.

Each of the following **N** lines contains a single word, its length being an **odd number greater than 2 and lesser than 14**. All characters are lowercase letters of the English alphabet.

OUTPUT

The first and only line of output must contain the length of the correct password and its central letter. **The solution will be unique**.

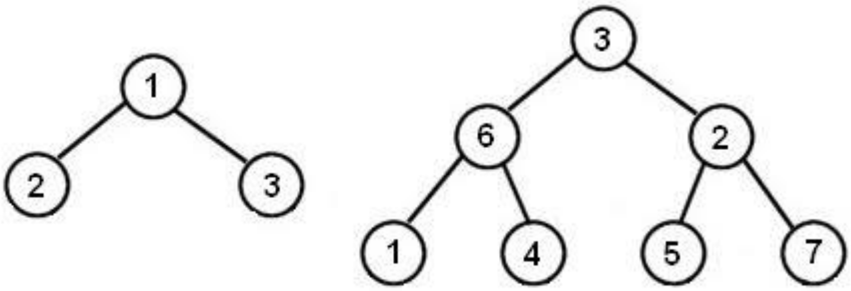
SAMPLE TESTS

input 4 las god psala sal **output** 3 a	**input** 4 kisik ptq tttrp tulipan **output** 5 s

Clarification of the first example: The required pair of words is "las" and "sal". Their length is 3 letters and the central character is 'a'.

Clarification of the second example: The word "kisik" can be found in both the original and reverse order on the list (the word is a palindrome), so it is a valid correct password.

Little Mirko has paid a touristic visit to a village nearby **Donji Andrijevci**, a town in Slavonia. As it happens, the arrangement of streets in the village looks awfully familiar to the shape of a perfect binary tree of the order **K**. A perfect binary tree of order **K** consists of 2^{K} - 1 nodes arranged in **K** levels (just like in the image). Each node contains a building labeled with a house number. Moreover, all buildings but the ones in the last level have a left and right child (see the image again).



Perfect binary tree of orders 2 and 3

Mirko has visited all the buildings in a village and noted down the exact entrance order. Now he wants to describe to you how the village looks like, but he can't quite remember. Luckily, he remembers the way in which he visited the buildings:

1. in the beginning, he was standing in front of the only building in the first level
2. if the building which he is currently standing in front of has a left child which he hasn't visited yet, he will move in front of the left child
3. if the building doesn't have a left child or he has already visited it, he will enter the current building and write its house number on his paper
4. if he has already visited the current building and the building has a right child, he will move in front of the right child
5. if he has visited the current building and its left and right child, he will return to the parent of the current building

After visiting the villages in the pictures above, the paper would look like this: 2-1-3 for the first village and 1-6-4-3-5-2-7 for the second village. Write a programme to help Mirko reconstruct the order of house numbers on each level.

INPUT

The first line of input contains the integer **K** (1 ≤ **K** ≤ 10), the number of levels of the village Mirko just visited.

The second line of input contains 2^{K} integers, the sequence of house numbers on Mirko's paper. The house numbers will be unique and from the interval [1, 2^{K}].

OUTPUT

The output must consist of **K** lines. The **i**th line must contain the sequence of house numbers in the **i**th level of the village.

SAMPLE TESTS

input 2 2 1 3 **output** 1 2 3	**input** 3 1 6 4 3 5 2 7 **output** 3 6 2 1 4 5 7

Clarification of the first and second example: The examples correspond to the images in the task.

Mirko likes to play with strings of characters, but this time he has taken it too far – he put an "explosion" in the string! An explosion is a series of characters which, if found in the vicinity of fire, explodes and starts a chain reaction.

Mirko, being as negligent as he usually is, forgot that his string contained an explosion and placed it near a candlelight. Thus the chain reaction began.

The chain reaction takes place in the following way:

- if a string contains explosions, they **all** explode and a new string is formed by concatenating the pieces **without** the exploding parts
- this concatenation could possibly create new explosions
- the chain reaction repeats while there are explosions in the string

Now Mirko wants to know whether anything will be left after this series of chain reactions. If **nothing** remains, output "FRULA" (without quotes). If, by any chance, **something** is left, output the final string remaining after all the reactions.

Please note: The explosion will not contain two equal characters.

INPUT

The first line of input contains **Mirko's string**, (1 ≤ **|Mirko's string|** ≤ 1 000 000).

The second line of input contains **the explosion string**, (1 ≤ |**explosion**| ≤ 36).

Both Mirko's string and the explosion string consist of uppercase and lowercase letters of the English alphabet and digits 0, 1, … 9.

OUTPUT

The first and only line of output must contain the final string remaining after all the reactions as stated in the task.

SCORING

In test cases worth 50% of total points, **N** will not exceed 3000.

SAMPLE TESTS

input mirkovC4nizCC44 C4 **output** mirkovniz	**input** 12ab112ab2ab 12ab **output** FRULA

Clarification of the second example: Firstly, the bombs on positions 1 and 6 explode. Then we are left with ****1****2ab (where * marks the character that exploded) and when that string is put together, we get 12ab. Sadly, that is an explosion all over again so it disappears.

Mirko has a chessboard with **N** rows and just three columns. Slavica has written an integer on each field. Mirko has **K** dominoes at his disposal, their dimensions being **2x1**, and has to arrange **all of them** on the board without overlapping, in a way that each domino covers exactly two fields of the board. He can rotate the dominoes as he pleases.

Help Mirko cover **the largest sum of numbers possible** with the dominoes!

INPUT

The first line of input contains the integer **N** (1 ≤ **N** ≤ 1000), the number of rows, and **K** (1 ≤ **K** ≤ 1000), the number of dominoes available.

Each of the following **N** lines contains three integers written in the **i**th row of the board. All numbers will be lesser than 10^6 by absolute value.

OUTPUT

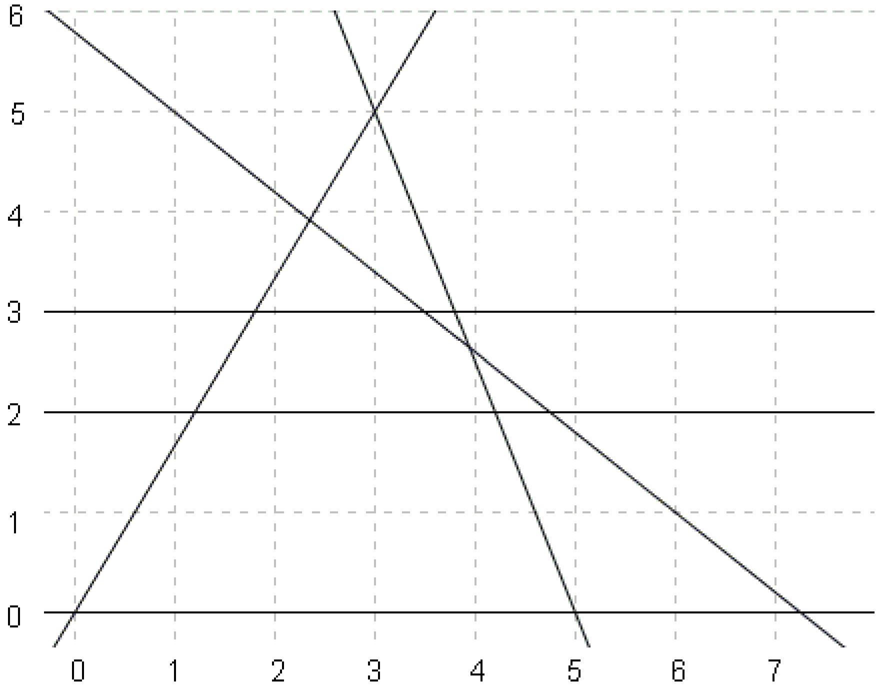
The first and only line of output must contain the maximal sum possible to cover with exactly **K** dominoes.

SAMPLE TESTS

input 5 3 2 1 -1 1 3 2 0 2 3 2 1 1 3 3 0 **output** 16	**input** 2 2 0 4 1 3 5 1 **output** 13

Clarification of the first example: It is optimal to place all dominoes horizontally and along the right edge of the second row, right edge of the third row and along the left edge of the final row.

You are given **N** lines, their equations being $\mathbf{A}_i x + \mathbf{B}_i y + \mathbf{C}_i = 0$ in the coordinate plane. Calculate the number of triangles whose sides lie on the given lines. Since the result can be very large, output the number modulo 1 000 000 007.



A possible position of lines.

Important note: No three lines will intersect at the same point.

INPUT

The first line of input contains the integer **N** ($1 \le \mathbf{N} \le 300\,000$), the number of lines.

Each of the following **N** lines contains three integers: $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{C}_i$, the numbers defining the **i**th line.

All numbers will be lesser than 10^9.

OUTPUT

The first and only line of output must consist of the required number from the task.

SCORING

In test cases worth 40% of total points, **N** will be lesser than 1000.

SAMPLE TESTS

input 6 0 1 0 -5 3 0 -5 -2 25 0 1 -3 0 1 -2 -4 -5 29 **output** 10	**input** 5 -5 3 0 -5 -3 -30 0 1 0 3 7 35 1 -2 -1 **output** 10

Clarification of the first example: The example corresponds to the image in the task.

Mirko has **N** items (labeled with numbers from 1 to **N**) and **L** drawers (labeled with numbers from 1 to **L**). All items are currently scattered throughout his room, so he decided to clean them up. Each drawer can contain one item, and in order to make it easier for Mirko to find them later, he has determined in advance **exactly two** drawers ($\mathbf{A_i}$ and $\mathbf{B_i}$) for each item **i**.

Mirko stores the items in order from 1 to **N** using the first rule he can apply:

1. If the drawer $\mathbf{A_i}$ is empty, he stores the item **i** in that drawer
2. If the drawer $\mathbf{B_i}$ is empty, he stores the item **i** in that drawer
3. Try to move the item from $\mathbf{A_i}$ to its other drawer; if that one's filled too, try moving that item to its other drawer, and so on until you either succeed or get back to a previously seen drawer. In case of success, store the item **i** in the drawer $\mathbf{A_i}$. In case of failure, continue to next rule.
4. Try moving the item from $\mathbf{B_i}$ to its other drawer; if that one's filled too, try moving that item to its other drawer, and so on until you either succeed or get back to a previously seen drawer. In case of success, store the item **i** in the drawer $\mathbf{B_i}$. In case of failure, continue to next rule.
5. Give up and throw away the item **i**.

For given pairs of drawers for each item, determine which items will be stored and which will be thrown away.

INPUT

The first line of input consists of two integers, **N** and **L** ($1 \leq \mathbf{N}, \mathbf{L} \leq 300\ 000$), the number of items and the number of drawers.

Each of the following **N** lines contains two integers: $\mathbf{A_i}$ and $\mathbf{B_i}$ ($1 \leq \mathbf{A_i}$ and $\mathbf{B_i} \leq \mathbf{L}$), pair of drawers corresponding to the item **i**. The numbers $\mathbf{A_i}$ and $\mathbf{B_i}$ will be different.

OUTPUT

For each item, respectively, output where it ends up.

In case the item is stored successfully, output "LADICA" (without quotes, Croatian word for drawer).

In case the item is thrown away, output "SMECE" (without quotes, Croatian word for trash).

SCORING

In test cases worth 50% of total points, both **N** and **L** will be lesser than 2000.

SAMPLE TESTS

input	**input**

5 3
1 2
1 3
1 2
1 3
1 2
	9 10
	1 2
	3 4
	5 6
	7 8
	9 10
	2 3
	1 5
	8 2
	7 9

output	**output**
LADICA LADICA LADICA SMECE SMECE	LADICA LADICA LADICA LADICA LADICA LADICA LADICA LADICA LADICA

Clarification of the first example: The first item goes to drawer 1 by rule 1). The second item goes to drawer 3 by rule 2). The third item goes to drawer 2 by rule 2).

For the fourth and fifth item, both drawers are already taken and cannot be emptied.

Clarification of the second example: The first six items go into drawers 1, 3, 5, 7, 9, 2 (respectively), by rule 1). For the seventh item, applying the rule 3), we try to move the item in drawer 1 to drawer 2, the item in drawer 2 to drawer 3, the item in drawer 3 to drawer 4, which we succeed because the drawer is empty.

The eighth item goes to drawer 8 which was empty from the beginning.

For the ninth item, applying the rule 3), we try to move the item in drawer 7 to drawer 8, the item in drawer 8 to drawer 2, the item in drawer 2 to drawer 1, the item in drawer 1 to drawer 5, the item in drawer 5 to drawer 6, which we succeed because the drawer is empty.