# Codeforces Round #401 (Div. 2)

## A. Shell Game

time limit per test: 0.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bomboslav likes to look out of the window in his room and watch lads outside playing famous shell game. The game is played by two persons: operator and player. Operator takes three similar opaque shells and places a ball beneath one of them. Then he shuffles the shells by swapping some pairs and the player has to guess the current position of the ball.

Bomboslav noticed that guys are not very inventive, so the operator always swaps the left shell with the middle one during odd moves (first, third, fifth, etc.) and always swaps the middle shell with the right one during even moves (second, fourth, etc.).

Let's number shells from $0$ to $2$ from left to right. Thus the left shell is assigned number $0$, the middle shell is $1$ and the right shell is $2$. Bomboslav has missed the moment when the ball was placed beneath the shell, but he knows that exactly $n$ movements were made by the operator and the ball was under shell $x$ at the end. Now he wonders, what was the initial position of the ball?

### Input

The first line of the input contains an integer $n$ $(1 \le n \le 2 \cdot 10^9)$ — the number of movements made by the operator.

The second line contains a single integer $x$ $(0 \le x \le 2)$ — the index of the shell where the ball was found after $n$ movements.

### Output

Print one integer from $0$ to $2$ — the index of the shell where the ball was initially placed.

**Examples**

| input |
| --- |
| 4<br>2 |

| output |
| --- |
| 1 |

| input |
| --- |
| 1<br>1 |

| output |
| --- |
| 0 |

### Note

In the first sample, the ball was initially placed beneath the middle shell and the operator completed four movements.

1. During the first move operator swapped the left shell and the middle shell. The ball is now under the left shell.
2. During the second move operator swapped the middle shell and the right one. The ball is still under the left shell.
3. During the third move operator swapped the left shell and the middle shell again. The ball is again in the middle.
4. Finally, the operators swapped the middle shell and the right shell. The ball is now beneath the right shell.

# B. Game of Credit Cards

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

After the fourth season Sherlock and Moriary have realized the whole foolishness of the battle between them and decided to continue their competitions in peaceful game of Credit Cards.

Rules of this game are simple: each player bring his favourite $n$-digit credit card. Then both players name the digits written on their cards one by one. If two digits are not equal, then the player, whose digit is smaller gets a flick (knock in the forehead usually made with a forefinger) from the other player. For example, if $n = 3$, Sherlock's card is $123$ and Moriarty's card has number $321$, first Sherlock names $1$ and Moriarty names $3$ so Sherlock gets a flick. Then they both digit $2$ so no one gets a flick. Finally, Sherlock names $3$, while Moriarty names $1$ and gets a flick.

Of course, Sherlock will play honestly naming digits one by one in the order they are given, while Moriary, as a true villain, plans to cheat. He is going to name his digits in some other order (however, he is not going to change the overall number of occurences of each digit). For example, in case above Moriarty could name $1, 2, 3$ and get no flicks at all, or he can name $2, 3$ and $1$ to give Sherlock two flicks.

Your goal is to find out the minimum possible number of flicks Moriarty will get (no one likes flicks) and the maximum possible number of flicks Sherlock can get from Moriarty. Note, that these two goals are different and the optimal result may be obtained by using different strategies.

## Input

The first line of the input contains a single integer $n$ $(1 \le n \le 1000)$ — the number of digits in the cards Sherlock and Moriarty are going to use.

The second line contains $n$ digits — Sherlock's credit card number.

The third line contains $n$ digits — Moriarty's credit card number.

## Output

First print the minimum possible number of flicks Moriarty will get. Then print the maximum possible number of flicks that Sherlock can get from Moriarty.

**Examples**

| input |
| --- |
| 3<br>123<br>321 |
| output |
| 0<br>2 |

| input |
| --- |
| 2<br>88<br>00 |
| output |
| 2<br>0 |

## Note

First sample is elaborated in the problem statement. In the second sample, there is no way Moriarty can avoid getting two flicks.

# C. Alyona and Spreadsheet

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

During the lesson small girl Alyona works with one famous spreadsheet computer program and learns how to edit tables.

Now she has a table filled with integers. The table consists of $n$ rows and $m$ columns. By $a_{i,j}$ we will denote the integer located at the $i$-th row and the $j$-th column. We say that the table is sorted in non-decreasing order in the column $j$ if $a_{i,j} \le a_{i+1,j}$ for all $i$ from $1$ to $n$ - $1$.

Teacher gave Alyona $k$ tasks. For each of the tasks two integers $l$ and $r$ are given and Alyona has to answer the following question: if one keeps the rows from $l$ to $r$ inclusive and deletes all others, will the table be sorted in non-decreasing order in at least one column? Formally, does there exist such $j$ that $a_{i,j} \le a_{i+1,j}$ for all $i$ from $l$ to $r$ - $1$ inclusive.

Alyona is too small to deal with this task and asks you to help!

## Input

The first line of the input contains two positive integers $n$ and $m$ ($1 \le n \cdot m \le 100\,000$) — the number of rows and the number of columns in the table respectively. Note that your are given a constraint that bound the product of these two integers, i.e. the number of elements in the table.

Each of the following $n$ lines contains $m$ integers. The $j$-th integers in the $i$ of these lines stands for $a_{i,j}$ ($1 \le a_{i,j} \le 10^9$).

The next line of the input contains an integer $k$ ($1 \le k \le 100\,000$) — the number of task that teacher gave to Alyona.

The $i$-th of the next $k$ lines contains two integers $l_i$ and $r_i$ ($1 \le l_i \le r_i \le n$).

## Output

Print "Yes" to the $i$-th line of the output if the table consisting of rows from $l_i$ to $r_i$ inclusive is sorted in non-decreasing order in at least one column. Otherwise, print "No".

**Example**

input

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

output

```
Yes
No
Yes
Yes
Yes
No
```

## Note

In the sample, the whole table is not sorted in any column. However, rows 1–3 are sorted in column $1$, while rows 4–5 are sorted in column $3$.

# D. Cloud of Hashtags

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

Vasya is an administrator of a public page of organization "Mouse and keyboard" and his everyday duty is to publish news from the world of competitive programming. For each news he also creates a list of hashtags to make searching for a particular topic more comfortable. For the purpose of this problem we define hashtag as a string consisting of lowercase English letters and exactly one symbol '#' located at the beginning of the string. The *length* of the hashtag is defined as the number of symbols in it **without** the symbol '#'.

The head administrator of the page told Vasya that hashtags should go in lexicographical order (take a look at the notes section for the definition).

Vasya is lazy so he doesn't want to actually change the order of hashtags in already published news. Instead, he decided to delete some suffixes (consecutive characters at the end of the string) of some of the hashtags. He is allowed to delete any number of characters, even the whole string except for the symbol '#'. Vasya wants to pick such a way to delete suffixes that the total number of deleted symbols is **minimum** possible. If there are several optimal solutions, he is fine with any of them.

## Input

The first line of the input contains a single integer $n$ ($1 \le n \le 500\,000$) — the number of hashtags being edited now.

Each of the next $n$ lines contains exactly one hashtag of positive length.

It is guaranteed that the total length of all hashtags (i.e. the total length of the string except for characters '#') won't exceed $500\,000$.

## Output

Print the resulting hashtags in any of the optimal solutions.

**Examples**

| input |
| --- |
| 3<br>#book<br>#bigtown<br>#big |
| output |
| #b<br>#big<br>#big |

| input |
| --- |
| 3<br>#book<br>#cool<br>#cold |
| output |
| #book<br>#co<br>#cold |

| input |
| --- |
| 4<br>#car<br>#cart<br>#art<br>#at |
| output |
| #<br>#<br>#art<br>#at |

| input |
| --- |
| 3<br>#apple<br>#apple<br>#fruit |
| output |
| #apple<br>#apple<br>#fruit |

## Note

Word $a_1, a_2, ..., a_m$ of length $m$ is *lexicographically not greater* than word $b_1, b_2, ..., b_k$ of length $k$, if one of two conditions hold:

- at first position $i$, such that $a_i \neq b_i$, the character $a_i$ goes earlier in the alphabet than character $b_i$, i.e. $a$ has smaller character than $b$ in the first position where they differ;
- if there is no such position $i$ and $m \leq k$, i.e. the first word is a prefix of the second or two words are equal.

The sequence of words is said to be sorted in lexicographical order if each word (except the last one) is lexicographically not greater than the next word.

For the words consisting of lowercase English letters the lexicographical order coincides with the alphabet word order in the dictionary.

According to the above definition, if a hashtag consisting of one character '`#`' it is lexicographically not greater than any other valid hashtag. That's why in the third sample we can't keep first two hashtags unchanged and shorten the other two.

# E. Hanoi Factory

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

Of course you have heard the famous task about Hanoi Towers, but did you know that there is a special factory producing the rings for this wonderful game? Once upon a time, the ruler of the ancient Egypt ordered the workers of Hanoi Factory to create as high tower as possible. They were not ready to serve such a strange order so they had to create this new tower using already produced rings.

There are $n$ rings in factory's stock. The $i$-th ring has inner radius $a_i$, outer radius $b_i$ and height $h_i$. The goal is to select some subset of rings and arrange them such that the following conditions are satisfied:

- Outer radiuses form a non-increasing sequence, i.e. one can put the $j$-th ring on the $i$-th ring only if $b_j \le b_i$.
- Rings should not fall one into the the other. That means one can place ring $j$ on the ring $i$ only if $b_j > a_i$.
- The total height of all rings used should be maximum possible.

## Input

The first line of the input contains a single integer $n$ ($1 \le n \le 100\,000$) — the number of rings in factory's stock.

The $i$-th of the next $n$ lines contains three integers $a_i$, $b_i$ and $h_i$ ($1 \le a_i, b_i, h_i \le 10^9$, $b_i > a_i$) — inner radius, outer radius and the height of the $i$-th ring respectively.

## Output

Print one integer — the maximum height of the tower that can be obtained.

**Examples**

| input |
| --- |
| 3<br>1 5 1<br>2 6 2<br>3 7 3 |
| output |
| 6 |

| input |
| --- |
| 4<br>1 2 1<br>1 3 3<br>4 6 2<br>5 7 1 |
| output |
| 4 |

## Note

In the first sample, the optimal solution is to take all the rings and put them on each other in order $3, 2, 1$.

In the second sample, one can put the ring $3$ on the ring $4$ and get the tower of height $3$, or put the ring $1$ on the ring $2$ and get the tower of height $4$.