A. Sea Battle

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

In order to make the "Sea Battle" game more interesting, Boris decided to add a new ship type to it. The ship consists of two rectangles. The first rectangle has a width of w_1 and a height of h_1, while the second rectangle has a width of w_2 and a height of h_2, where $w_1 \ge w_2$. In this game, exactly one ship is used, made up of two rectangles. There are no other ships on the field.

The rectangles are placed on field in the following way:

- the second rectangle is on top the first rectangle;
- they are aligned to the left, i.e. their left sides are on the same line;
- the rectangles are adjacent to each other without a gap.

See the pictures in the notes: the first rectangle is colored red, the second rectangle is colored blue.

Formally, let's introduce a coordinate system. Then, the leftmost bottom cell of the first rectangle has coordinates $(1, 1)$, the rightmost top cell of the first rectangle has coordinates (w_1, h_1), the leftmost bottom cell of the second rectangle has coordinates $(1, h_1 + 1)$ and the rightmost top cell of the second rectangle has coordinates $(w_2, h_1 + h_2)$.

After the ship is completely destroyed, all cells neighboring by side or a corner with the ship are marked. Of course, only cells, which don't belong to the ship are marked. On the pictures in the notes such cells are colored green.

Find out how many cells should be marked after the ship is destroyed. The field of the game is infinite in any direction.

Input

Four lines contain integers w_1, h_1, w_2 and h_2 ($1 \le w_1, h_1, w_2, h_2 \le 10^8$, $w_1 \ge w_2$) — the width of the first rectangle, the height of the first rectangle, the width of the second rectangle and the height of the second rectangle. You can't rotate the rectangles.

Output

Print exactly one integer — the number of cells, which should be marked after the ship is destroyed.

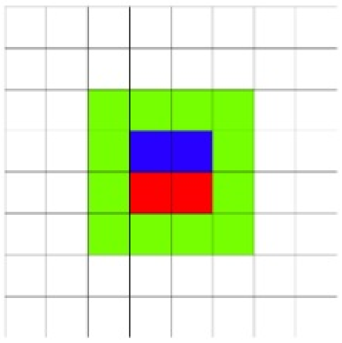
Examples

input
```
2 1 2 1
```
output
```
12
```

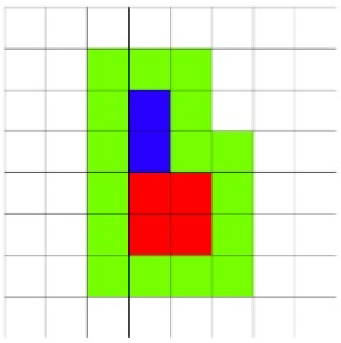
input
```
2 2 1 2
```
output
```
16
```

Note

In the first example the field looks as follows (the first rectangle is red, the second rectangle is blue, green shows the marked squares):



In the second example the field looks as:



B. Draw!

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You still have partial information about the score during the historic football match. You are given a set of pairs (a_i, b_i), indicating that at some point during the match the score was "a_i: b_i". It is known that if the current score is «x:y», then after the goal it will change to "$x+1$:y" or "x:$y+1$". What is the largest number of times a draw could appear on the scoreboard?

The pairs "a_i:b_i" are given in chronological order (time increases), but you are given score only for some moments of time. The last pair corresponds to the end of the match.

Input

The first line contains a single integer n ($1 \le n \le 10000$) — the number of known moments in the match.

Each of the next n lines contains integers a_i and b_i ($0 \le a_i, b_i \le 10^9$), denoting the score of the match at that moment (that is, the number of goals by the first team and the number of goals by the second team).

All moments are given in chronological order, that is, sequences x_i and y_j are non-decreasing. The last score denotes the final result of the match.

Output

Print the maximum number of moments of time, during which the score was a draw. The starting moment of the match (with a score 0:0) is also counted.

Examples

input

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

output

```
2
```

input

```
3
0 0
0 0
0 0
```

output

```
1
```

input

```
1
5 4
```

output

```
5
```

Note

In the example one of the possible score sequences leading to the maximum number of draws is as follows: 0:0, 1:0, 2:0, 2:1, 3:1, 3:2, 3:3, 3:4.

C. Birthday

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input
output: standard output

Cowboy Vlad has a birthday today! There are n children who came to the celebration. In order to greet Vlad, the children decided to form a circle around him. Among the children who came, there are both tall and low, so if they stand in a circle arbitrarily, it may turn out, that there is a tall and low child standing next to each other, and it will be difficult for them to hold hands. Therefore, children want to stand in a circle so that the maximum difference between the growth of two neighboring children would be minimal possible.

Formally, let's number children from 1 to n in a circle order, that is, for every i child with number i will stand next to the child with number $i+1$, also the child with number 1 stands next to the child with number n. Then we will call the discomfort of the circle the maximum absolute difference of heights of the children, who stand next to each other.

Please help children to find out how they should reorder themselves, so that the resulting discomfort is smallest possible.

Input

The first line contains a single integer n ($2 \leq n \leq 100$) — the number of the children who came to the cowboy Vlad's birthday.

The second line contains integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) denoting heights of every child.

Output

Print exactly n integers — heights of the children in the order in which they should stand in a circle. You can start printing a circle with any child.

If there are multiple possible answers, print any of them.

Examples

input

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

output

```
1 2 3 2 1
```

input

```
3
30 10 20
```

output

```
10 20 30
```

Note

In the first example, the discomfort of the circle is equal to 1, since the corresponding absolute differences are 1, 1, 1 and 0. Note, that sequences $[2, 3, 2, 1, 1]$ and $[3, 2, 1, 1, 2]$ form the same circles and differ only by the selection of the starting point.

In the second example, the discomfort of the circle is equal to 20, since the absolute difference of 10 and 30 is equal to 20.

D. Gourmet choice

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

Mr. Apple, a gourmet, works as editor-in-chief of a gastronomic periodical. He travels around the world, tasting new delights of famous chefs from the most fashionable restaurants. Mr. Apple has his own signature method of review — in each restaurant Mr. Apple orders two sets of dishes on two different days. All the dishes are different, because Mr. Apple doesn't like to eat the same food. For each pair of dishes from different days he remembers exactly which was better, or that they were of the same quality. After this the gourmet evaluates each dish with a positive integer.

Once, during a revision of a restaurant of Celtic medieval cuisine named «Poisson», that serves chestnut soup with fir, warm soda bread, spicy lemon pie and other folk food, Mr. Apple was very pleasantly surprised the gourmet with its variety of menu, and hence ordered too much. Now he's confused about evaluating dishes.

The gourmet tasted a set of n dishes on the first day and a set of m dishes on the second day. He made a table a of size $n \times m$, in which he described his impressions. If, according to the expert, dish i from the first set was better than dish j from the second set, then a_{ij} is equal to ">", in the opposite case a_{ij} is equal to "<". Dishes also may be equally good, in this case a_{ij} is "=".

Now Mr. Apple wants you to help him to evaluate every dish. Since Mr. Apple is very strict, he will evaluate the dishes so that the maximal number used is as small as possible. But Mr. Apple also is very fair, so he never evaluates the dishes so that it goes against his feelings. In other words, if a_{ij} is "<", then the number assigned to dish i from the first set should be less than the number of dish j from the second set, if a_{ij} is ">", then it should be greater, and finally if a_{ij} is "=", then the numbers should be the same.

Help Mr. Apple to evaluate each dish from both sets so that it is consistent with his feelings, or determine that this is impossible.

Input

The first line contains integers n and m ($1 \leq n, m \leq 1000$) — the number of dishes in both days.

Each of the next n lines contains a string of m symbols. The j-th symbol on i-th line is a_{ij}. All strings consist only of "<", ">" and "=".

Output

The first line of output should contain "Yes", if it's possible to do a correct evaluation for all the dishes, or "No" otherwise.

If case an answer exist, on the second line print n integers — evaluations of dishes from the first set, and on the third line print m integers — evaluations of dishes from the second set.

Examples

input

```
3 4
>>>>
>>>>
>>>>
```

output

```
Yes
2 2 2
1 1 1 1
```

input

```
3 3
>>>
<<<
>>>
```

output

```
Yes
3 1 3
2 2 2
```

input

```
3 2
==
=<
==
```

output

```
No
```

Note

In the first sample, all dishes of the first day are better than dishes of the second day. So, the highest score will be 2, for all dishes of the first day.

In the third sample, the table is contradictory — there is no possible evaluation of the dishes that satisfies it.

E. String Multiplication

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Roman and Denis are on the trip to the programming competition. Since the trip was long, they soon got bored, and hence decided to came up with something. Roman invented a pizza's recipe, while Denis invented a string multiplication. According to Denis, the result of multiplication (product) of strings s of length m and t is a string $t + s_1 + t + s_2 + \ldots + t + s_m + t$, where s_i denotes the i-th symbol of the string s, and "+" denotes string concatenation. For example, the product of strings "abc" and "de" is a string "deadebdecde", while the product of the strings "ab" and "z" is a string "zazbz". Note, that unlike the numbers multiplication, the product of strings s and t is not necessarily equal to product of t and s.

Roman was jealous of Denis, since he invented such a cool operation, and hence decided to invent something string-related too. Since Roman is beauty-lover, he decided to define the *beauty* of the string as the length of the longest substring, consisting of only one letter. For example, the beauty of the string "xayyaaabca" is equal to 3, since there is a substring "aaa", while the beauty of the string "qwerqwer" is equal to 1, since all neighboring symbols in it are different.

In order to entertain Roman, Denis wrote down n strings $p_1, p_2, p_3, \ldots, p_n$ on the paper and asked him to calculate the beauty of the string $(\ldots (((p_1 \cdot p_2) \cdot p_3) \cdot \ldots) \cdot p_n$, where $s \cdot t$ denotes a multiplication of strings s and t. Roman hasn't fully realized how Denis's multiplication works, so he asked you for a help. Denis knows, that Roman is very impressionable, he guarantees, that the beauty of the resulting string is at most 10^9.

Input

The first line contains a single integer n $(2 \leq n \leq 100\,000)$ — the number of strings, wroted by Denis.

Next n lines contain non-empty strings $p_1, p_2, \ldots, p_n$, consisting of lowercase english letters.

It's guaranteed, that the total length of the strings p_i is at most $100\,000$, and that's the beauty of the resulting product is at most 10^9.

Output

Print exactly one integer — the beauty of the product of the strings.

Examples

input

```
3
a
b
a
```

output

```
3
```

input

```
2
bnn
a
```

output

```
1
```

Note

In the first example, the product of strings is equal to "abaaaba".

In the second example, the product of strings is equal to "abanana".

F. Asya And Kittens

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

Asya loves animals very much. Recently, she purchased n kittens, enumerated them from 1 and n and then put them into the cage. The cage consists of one row of n cells, enumerated with integers from 1 to n from left to right. Adjacent cells had a partially transparent partition wall between them, hence there were $n-1$ partitions originally. Initially, each cell contained exactly one kitten with some number.

Observing the kittens, Asya noticed, that they are very friendly and often a pair of kittens in neighboring cells wants to play together. So Asya started to remove partitions between neighboring cells. In particular, on the day i, Asya:

- Noticed, that the kittens x_i and y_i, located in neighboring cells want to play together.
- Removed the partition between these two cells, efficiently creating a single cell, having all kittens from two original cells.

Since Asya has never putted partitions back, after $n-1$ days the cage contained a single cell, having all kittens.

For every day, Asya remembers numbers of kittens x_i and y_i, who wanted to play together, however she doesn't remember how she placed kittens in the cage in the beginning. Please help her and find any possible initial arrangement of the kittens into n cells.

Input

The first line contains a single integer n ($2 \le n \le 150\,000$) — the number of kittens.

Each of the following $n-1$ lines contains integers x_i and y_i ($1 \le x_i, y_i \le n$, $x_i \ne y_i$) — indices of kittens, which got together due to the border removal on the corresponding day.

It's guaranteed, that the kittens x_i and y_i were in the different cells before this day.

Output

For every cell from 1 to n print a single integer — the index of the kitten from 1 to n, who was originally in it.

All printed integers must be distinct.

It's guaranteed, that there is at least one answer possible. In case there are multiple possible answers, print any of them.

Example

input

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

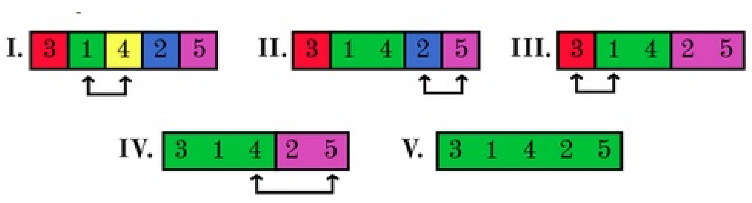
output

```
3 1 4 2 5
```

Note

The answer for the example contains one of several possible initial arrangements of the kittens.

The picture below shows how the cells were united for this initial arrangement. Note, that the kittens who wanted to play together on each day were indeed in **adjacent cells**.



G. Most Dangerous Shark

time limit per test: 2 seconds

memory limit per test: 768 megabytes

input: standard input

output: standard output

Semyon participates in the most prestigious competition of the world ocean for the title of the most dangerous shark. During this competition sharks compete in different subjects: speed swimming, masking, map navigation and many others. Now Semyon is taking part in «destruction» contest.

During it, m dominoes are placed in front of the shark. All dominoes are on the same line, but the height of the dominoes may vary. The distance between adjacent dominoes is 1. Moreover, each Domino has its own cost value, expressed as an integer. The goal is to drop all the dominoes. To do this, the shark can push any domino to the left or to the right, and it will begin falling in this direction. If during the fall the domino touches other dominoes, they will also start falling in the same direction in which the original domino is falling, thus beginning a chain reaction, as a result of which many dominoes can fall. A falling domino touches another one, if and only if the distance between them **was strictly less than** the height of the falling domino, the dominoes do not necessarily have to be adjacent.

Of course, any shark can easily drop all the dominoes in this way, so the goal is not to drop all the dominoes, but do it with a minimum cost. The cost of the destruction is the sum of the costs of dominoes that the shark needs to push to make all the dominoes fall.

Simon has already won in the previous subjects, but is not smart enough to win in this one. Help Semyon and determine the minimum total cost of the dominoes he will have to push to make all the dominoes fall.

Input

In order to reduce input size, the heights and costs of the dominoes are described with blocks.

The first line contains two integers n and m ($1 \le n \le 250\,000, 1 \le m \le 10^7$) — the number of the blocks and the total number of the dominoes Semyon must drop.

Then descriptions of n blocks follow. Description of every block consists of three lines.

The first line of block's description contains a single integer k_i ($1 \le k_i \le 250\,000, \sum_{i=1}^{n} k_i \le 250\,000$) — the number of dominoes in the block.

The second line of block's description contains k_i integers a_j ($1 \le a_j \le m$) — the heights of the dominoes in the blocks.

The third line contains k_i integers c_j ($1 \le c_j \le 100\,000$) — the costs of the dominoes in the block.

Then the domino sequence is described (from left to right).

The first line of this description contains a single integer q ($n \le q \le 250\,000$) — the number of blocks in the sequence of domino sequence.

Each of the following q lines contains integers id_i, mul_i ($1 \le id_i \le n$, $1 \le mul_i \le 100\,000$), denoting that the next k_{id_i} dominoes are dominoes of the block id_i, with their cost multiplied by mul_i.

It's guaranteed, that $\sum_{i=1}^{q} k_{id_i} = m$, and that's every block is used in the sequence at least once.

Output

Print exactly one integer — the minimum cost to make all the dominoes fall.

Examples

input

```
2 7
3
1 2 2
1 2 1
1
3
2
3
2 2
1 3
1 1
```

output

```
5
```

input

```
1 1
1
1
100000
1
1 100000
```

output

```
10000000000
```

Note

In the first example, there are 7 dominoes in front of the Semyon. Their heights are equal to $[3, 1, 2, 2, 1, 2, 2]$, and their costs are equal to $[4, 3, 6, 3, 1, 2, 1]$. Semyon should drop the domino with index 7 to the left, it will fall and drop the domino 6 as well. The domino 6 during the fall will drop the domino 5, however the latter will not drop any more dominoes. Then Semyon should drop domino with number 1 to the right and it will drop dominoes 2 and 3 after falling. And the domino 3 will drop the domino 4 after falling. Hence all dominoes are fallen this way.

In the second example, there is a single domino of cost 10000000000.