Codeforces Round #506 (Div. 3)

A. Many Equal Substrings

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

You are given a string t consisting of n lowercase Latin letters and an integer number k.

Let's define a substring of some string s with indices from l to r as $s[l \dots r]$.

Your task is to construct such string s of minimum possible length that there are exactly k positions i such that $s[i \dots i+n-1] = t$. In other words, your task is to construct such string s of minimum possible length that there are exactly k substrings of s equal to t.

It is guaranteed that the answer is always unique.

Input

The first line of the input contains two integers n and k ($1 \le n, k \le 50$) — the length of the string t and the number of substrings.

The second line of the input contains the string t consisting of exactly n lowercase Latin letters.

Output

Print such string s of minimum possible length that there are exactly k substrings of s equal to t.

It is guaranteed that the answer is always unique.

Examples

input
```
3 4
aba
```
output
```
ababababa
```

input
```
3 2
cat
```
output
```
catcat
```

B. Creating the Contest

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

You are given a problemset consisting of n problems. The difficulty of the i-th problem is a_i. It is guaranteed that all difficulties are distinct and are given in the increasing order.

You have to assemble the contest which consists of some problems of the given problemset. In other words, *the contest you have to assemble should be a subset of problems (not necessary consecutive) of the given problemset*. There is only one condition that should be satisfied: for each problem but the hardest one (the problem with the maximum difficulty) there should be a problem with the difficulty greater than the difficulty of this problem but not greater than twice the difficulty of this problem. In other words, let $a_{i_1}, a_{i_2}, \dots, a_{i_p}$ be the difficulties of the selected problems in increasing order. Then for each j from 1 to $p-1$ $a_{i_{j+1}} \le a_{i_j} \cdot 2$ should hold. *It means that the contest consisting of only one problem is always valid.*

Among all contests satisfying the condition above you have to assemble one with the maximum number of problems. Your task is to find this number of problems.

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of problems in the problemset.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — difficulties of the problems. **It is guaranteed that difficulties of the problems are distinct and are given in the increasing order**.

Output

Print a single integer — maximum number of problems in the contest satisfying the condition in the problem statement.

Examples

input

```
10
1 2 5 6 7 10 21 23 24 49
```

output

```
4
```

input

```
5
2 10 50 110 250
```

output

```
1
```

input

```
6
4 7 12 100 150 199
```

output

```
3
```

Note

Description of the first example: there are 10 valid contests consisting of 1 problem, 10 valid contests consisting of 2 problems ($[1, 2], [5, 6], [5, 7], [5, 10], [6, 7], [6, 10], [7, 10], [21, 23], [21, 24], [23, 24]$), 5 valid contests consisting of 3 problems ($[5, 6, 7], [5, 6, 10], [5, 7, 10], [6, 7, 10], [21, 23, 24]$) and a single valid contest consisting of 4 problems ($[5, 6, 7, 10]$).

In the second example all the valid contests consist of 1 problem.

In the third example are two contests consisting of 3 problems: $[4, 7, 12]$ and $[100, 150, 199]$.

C. Maximal Intersection

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given n segments on a number line; each endpoint of every segment has integer coordinates. Some segments can degenerate to points. Segments can intersect with each other, be nested in each other or even coincide.

The intersection of a sequence of segments is such a maximal set of points (not necesserily having integer coordinates) that each point lies within every segment from the sequence. If the resulting set isn't empty, then it always forms some continuous segment. The length of the intersection is the length of the resulting segment or 0 in case the intersection is an empty set.

For example, the intersection of segments $[1;5]$ and $[3;10]$ is $[3;5]$ (length 2), the intersection of segments $[1;5]$ and $[5;7]$ is $[5;5]$ (length 0) and the intersection of segments $[1;5]$ and $[6;6]$ is an empty set (length 0).

Your task is to remove exactly one segment from the given sequence in such a way that the intersection of the remaining $(n-1)$ segments has the maximal possible length.

Input

The first line contains a single integer n ($2 \le n \le 3 \cdot 10^5$) — the number of segments in the sequence.

Each of the next n lines contains two integers l_i and r_i ($0 \le l_i \le r_i \le 10^9$) — the description of the i-th segment.

Output

Print a single integer — the maximal possible length of the intersection of $(n-1)$ remaining segments after you remove exactly one segment from the sequence.

Examples

input

```
4
1 3
2 6
0 4
3 3
```

output

```
1
```

input

```
5
2 6
1 3
0 4
1 20
0 4
```

output

```
2
```

input

```
3
4 5
1 2
9 20
```

output

```
0
```

input

```
2
3 10
1 5
```

output

```
7
```

Note

In the first example you should remove the segment $[3;3]$, the intersection will become $[2;3]$ (length 1). Removing any other segment will result in the intersection $[3;3]$ (length 0).

In the second example you should remove the segment $[1;3]$ or segment $[2;6]$, the intersection will become $[2;4]$ (length 2) or $[1;3]$ (length 2), respectively. Removing any other segment will result in the intersection $[2;3]$ (length 1).

In the third example the intersection will become an empty set no matter the segment you remove.

In the fourth example you will get the intersection $[3;10]$ (length 7) if you remove the segment $[1;5]$ or the intersection $[1;5]$ (length 4) if you remove the segment $[3;10]$.

D. Concatenated Multiples

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

You are given an array a, consisting of n positive integers.

Let's call a concatenation of numbers x and y the number that is obtained by writing down numbers x and y one right after another without changing the order. For example, a concatenation of numbers 12 and 3456 is a number 123456.

Count the number of ordered pairs of positions (i, j) ($i \neq j$) in array a such that the concatenation of a_i and a_j is divisible by k.

Input

The first line contains two integers n and k ($1 \le n \le 2 \cdot 10^5$, $2 \le k \le 10^9$).

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$).

Output

Print a single integer — the number of ordered pairs of positions (i, j) ($i \neq j$) in array a such that the concatenation of a_i and a_j is divisible by k.

Examples

input

```
6 11
45 1 10 12 11 7
```

output

```
7
```

input

```
4 2
2 78 4 10
```

output

```
12
```

input

```
5 2
3 7 19 3 3
```

output

```
0
```

Note

In the first example pairs $(1, 2)$, $(1, 3)$, $(2, 3)$, $(3, 1)$, $(3, 4)$, $(4, 2)$, $(4, 3)$ suffice. They produce numbers 451, 4510, 110, 1045, 1012, 121, 1210, respectively, each of them is divisible by 11.

In the second example all $n(n-1)$ pairs suffice.

In the third example no pair is sufficient.

E. Tree with Small Distances

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

You are given an undirected tree consisting of n vertices. An undirected tree is a connected undirected graph with $n-1$ edges.

Your task is to add the minimum number of edges in such a way that the length of the shortest path from the vertex 1 to any other vertex is at most 2. Note that you are not allowed to add loops and multiple edges.

Input

The first line contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

The following $n-1$ lines contain edges: edge i is given as a pair of vertices u_i, v_i ($1 \le u_i, v_i \le n$). It is guaranteed that the given edges form a tree. *It is guaranteed that there are no loops and multiple edges in the given edges.*

Output

Print a single integer — the minimum number of edges you have to add in order to make the shortest distance from the vertex 1 to any other vertex at most 2. Note that you are not allowed to add loops and multiple edges.

Examples

input

```
7
1 2
2 3
2 4
4 5
4 6
5 7
```

output

```
2
```

input

```
7
1 2
1 3
2 4
2 5
3 6
1 7
```

output

```
0
```

input

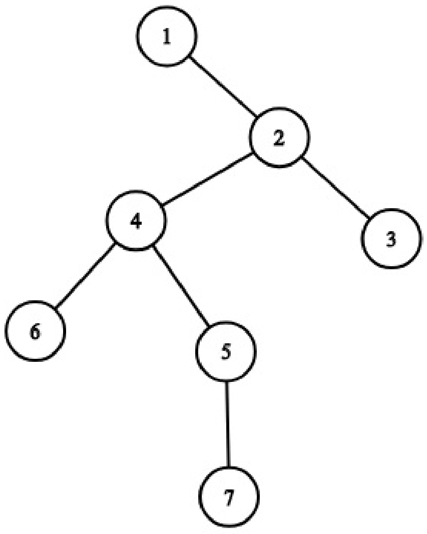
```
7
1 2
2 3
3 4
3 5
3 6
3 7
```

output

```
1
```

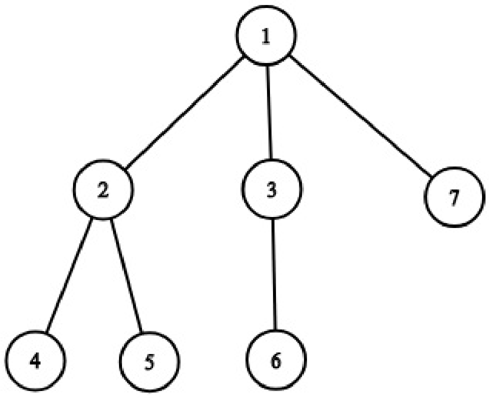
Note

The tree corresponding to the first example:



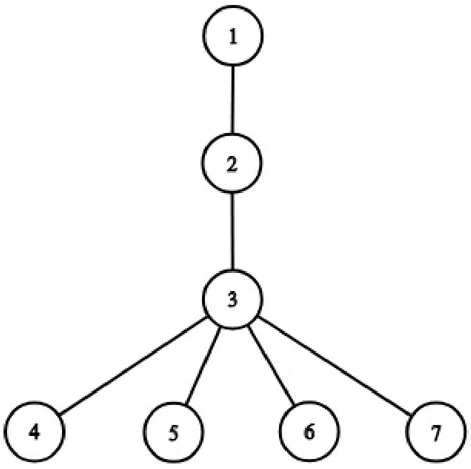
The answer is 2, some of the possible answers are the following: $[(1,5),(1,6)]$, $[(1,4),(1,7)]$, $[(1,6),(1,7)]$.

The tree corresponding to the second example:



The answer is 0.

The tree corresponding to the third example:



The answer is 1, only one possible way to reach it is to add the edge $(1,3)$.

F. Multicolored Markers

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input

output: standard output

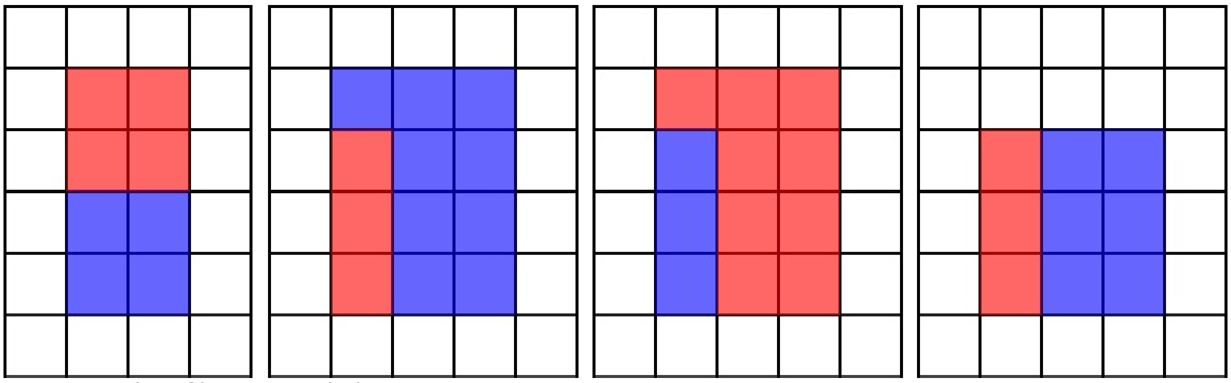
There is an infinite board of square tiles. Initially all tiles are white.

Vova has a red marker and a blue marker. Red marker can color a tiles. Blue marker can color b tiles. If some tile isn't white then you can't use marker of any color on it. Each marker must be drained completely, so at the end there should be exactly a red tiles and exactly b blue tiles across the board.

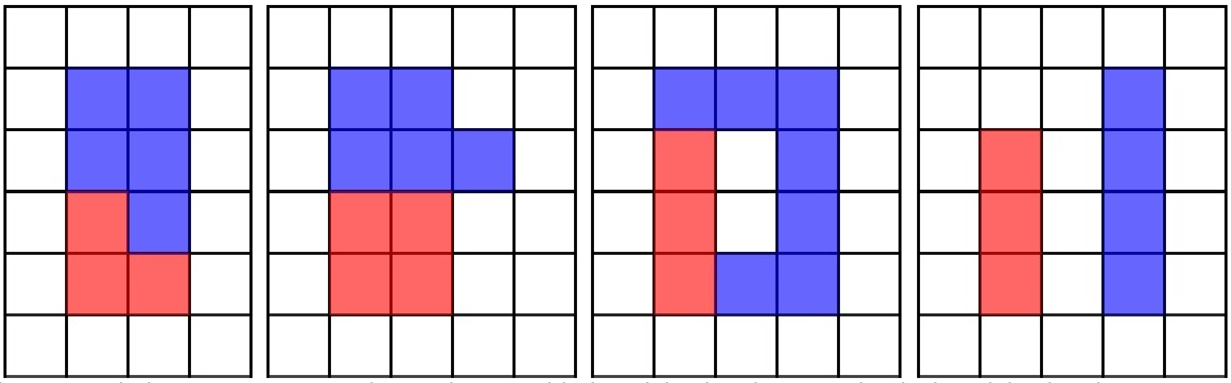
Vova wants to color such a set of tiles that:

- they would form a **rectangle**, consisting of exactly $a + b$ colored tiles;
- all tiles of at least one color would also form a **rectangle**.

Here are some examples of correct colorings:



Here are some examples of incorrect colorings:



Among all correct colorings Vova wants to choose the one with the minimal perimeter. What is the minimal perimeter Vova can obtain?

It is guaranteed that there exists at least one correct coloring.

Input

A single line contains two integers a and b ($1 \le a, b \le 10^{14}$) — the number of tiles red marker should color and the number of tiles blue marker should color, respectively.

Output

Print a single integer — the minimal perimeter of a colored rectangle Vova can obtain by coloring exactly a tiles red and exactly b tiles blue.

It is guaranteed that there exists at least one correct coloring.

Examples

input

```
4 4
```

output

```
12
```

input

```
3 9
```

output

```
14
```

input

```
9 3
```

output

```
14
```

input

```
3 6
```

output

```
12
```

input

```
506 2708
```

output

```
3218
```

Note

The first four examples correspond to the first picture of the statement.

Note that for there exist multiple correct colorings for all of the examples.

In the first example you can also make a rectangle with sides 1 and 8, though its perimeter will be 18 which is greater than 8.

In the second example you can make the same resulting rectangle with sides 3 and 4, but red tiles will form the rectangle with sides 1 and 3 and blue tiles will form the rectangle with sides 3 and 3.