CS F211
Data Structures and Algorithms
Assignment - 9

Allowed languages: **C**

March 26, 2021

General Tips

- Try to use functions as much as possible in your code. Functions increase reusability and the pass-by-value feature provides a significant help sometimes. Modularizing your code also helps you to debug efficiently.
- Use `scanf` to read characters/strings from STDIN. Avoid using `getchar`, `getc` or `gets`. Try to read up about character suppression in `scanf` as it will be very helpful in some of the problems.
- Use `printf` instead of `putc`, `putchar` or `puts` to print character/string output on STDOUT.
- Indent your code appropriately and use proper variable names. These increase readability and writability of the code. Also, use comments wherever necessary.
- Use a proper IDEs like Sublime Text or VSCode as they help to run and test your code on multiple test-cases easily. You can install Windows Subsystem Linux (WSL) or MinGW 7.3.0, if you are Windows user to compile and run your programs. Alternatively, you can run and test your codes on Online GDB. If you are using WSL or Linux to run your programs, make sure that the gcc version is `gcc 5.4.1 c99`.

A: The Shortest Statement You'll Ever Find

You are given an array of N integers and q queries with the i^{th} query containing the number x_i. For the i^{th} query print the frequency of x_i in the array. Implement this using a red black tree. *Note: Try to make your implementation as general as possible since subsequent problems also might require the usage of a Red-Black tree.*

Input

The first line contains a single integer N as described above ($1 \leq N \leq 10^5$). The second line of input contains N integers where the i^{th} number represents a_i ($1 \leq a_i \leq 10^9$). The third line contains a single integer q. The next q lines contain a single integer each, representing the i^{th} query.

Output

Print q lines, the i^{th} line containing the answer for the i^{th} query.

---

input
```
7
2 5 1000000000 7 1000000000 5 5
3
5
1000000000
1
```

output
```
3
2
0
```

explanation

In the first query, the frequency of 5 is 3. In the second query, the frequency of 1000000000 is 2. In the third query, there is no 1 anywhere in the array, hence the frequency is 0

B: The Best Part of Being a Scout

Levi is a complete neat freak and his squad has to clean up his entire room before he gets back from a scouting expedition. They're finished with almost everything and they need your help to clean up one last thing. There is an array of N elements with the i^{th} element being a_i. Levi thinks that the array is clean if $frequency(a_i) = a_i$ for all i. That is, the frequency of a_i should be a_i. Note that an empty array is considered clean. Levi's array is a little messed up and isn't clean right now. Find the smallest number of elements to delete to make the array clean.

Input

The first line contains an integer N representing the number of elements in the array ($1 \leq N \leq 10^5$). The next line contains N numbers representing the i^{th} element of the array a_i ($1 \leq a_i \leq 10^9$).

Output

Print a single integer, the minimum number of elements to delete to make the array clean

---

```
input
5
2 4 1 4 2

output
2
```

explanation

There is no way to get 4 elements with the value 4, so delete all the elements with value 4. Hence we would remove 2 elements and the remaining array is $[2, 1, 2]$. This is a clean array since 2 occurs 2 times and 1 occurs 1 time.

C: Sleepwalking

Dorothy Unsworth sucked you up into her dream world again and is willing to let you out only if you beat her at a game. You are given N positive integers. In one operation, you can choose some even number c and divide *all* elements in the array equal to c by 2. The goal of the game is to perform operations until the array consists entirely of odd numbers. If you stay in the dream world too long you'll fall asleep and die so you have to find the minimum possible number of operations to make the array consist of only odd numbers.

Input

The first input line has a single integer N ($1 \leq N \leq 10^5$) The next line contains N integers representing the array ($1 \leq a_i \leq 10^9$).

Output

Output a single integer representing the smallest number of moves needed to make every element in the array odd.

```
input
6
40 6 40 3 20 1

output
4
```

explanation

Here the optimal sequence of moves is as follows: Choose c as 40 and make all the 40s in the array equal to 20. Now we have: $20, 6, 20, 3, 20, 1$. Now if we choose c as 20, we have $10, 6, 10, 3, 10, 1$. Next, take c as 10. The array becomes $5, 6, 5, 3, 5, 1$. Lastly, take c as 6 to get the final array as $5, 3, 5, 3, 5, 1$ where all the numbers are odd. This took a total of 4 steps and it is easy to see that this is the minimum number of steps.

D: Captain and Vice Captain

Luffy made the biggest mistake of his life and sent Zoro out to get groceries. Now it's too late and Zoro went around in circles trying to find it although somehow he finally reached it. Luffy has gotten really good at using observation haki and using that, he's able to determine the exact path that Zoro takes. The path that Zoro takes can be imagined on the coordinate plane with his possible moves being up,right,left, or down by one unit. Assume his initial coordinate is $(0, 0)$. The path is represented by a string containing the letters 'L' (Left), 'R' (Right), 'U' (Up), and 'D' (Down). Luffy wants to save Zoro some time and to do that he can remove any non empty contiguous substring from the path *without changing the Zoro's final destination.* Since Luffy is lazy and doesn't actually care that much, he wants to find the smallest possible such substring by length. Help him find it by printing the indices of the starting and ending elements of the substring. If there is no such substring print -1. Print the indices with 1-indexing.

Input

The first line contains a single integer N representing the length of the string ($1 \leq N \leq 10^5$). The next line contains a string consisting of only the above mentioned 4 letters.

Output

print two integers l and r representing the starting and ending indices of the substring in 1-indexing. If there is no such non empty substring then print -1.

```
input
10
LURDLLLRRD

output
7 8

explanation
You could also remove the substring from 1 to 4 and maintain the same destination,
but the size of that substring is not minimum.  Any other substring would change the
final destination
```

E: Gojou is on Another Vacation

You have encountered a somewhat powerful spirit and it already has you captured in a maze that is in the form of a tree (with nodes and edges). To escape the maze you must first place a starting cursed energy bomb and an ending cursed energy bomb on two distinct nodes u and v respectively and then traverse the shortest path from u to v. There is one condition though. There exist two special nodes x and y such that if you reach y after reaching x somewhere in the path from u to v, then your plan fails. The special nodes x and y are given to you beforehand. Find all possible pairs u, v where you can place the cursed energy bombs.

Input

The first line contains 3 integers, N, x, y where N is the number of nodes and x and y are the special nodes ($1 \leq N \leq 10^5, 1 \leq x, y \leq N, x \neq y$). The next $N-1$ lines contain two integers each, u_i and v_i representing an edge between u_i and v_i.

Output

Print a single integer representing the number of pairs of nodes that you can place the bombs on.

```
input
3 1 3
1 2
1 3

output
4

explanation
```

In this example, you can go from 2 to 1, 1 to 2, 3 to 2, and 3 to 1. Any other path will pass through x and y in that order

---

F: BITSian Standard Time (BST)

You are given a series of N numbers. For each of these numbers, you will have to perform operations on a Binary Search Tree (BST).

1. begin with an empty BST.
2. read the next number in the series from input. Let us call this input A_i.
3. if A_i is already present in the BST, ignore that input.
4. if A_i is not present in the BST, insert it into the tree as a leaf at the appropriate position. Do not perform any rebalancing.

Print the *postorder traversal* of the formed BST. Note: Take a look at this article for more information on a BST. You can also use this visualiser to help you plot BSTs if you're having trouble.

Input

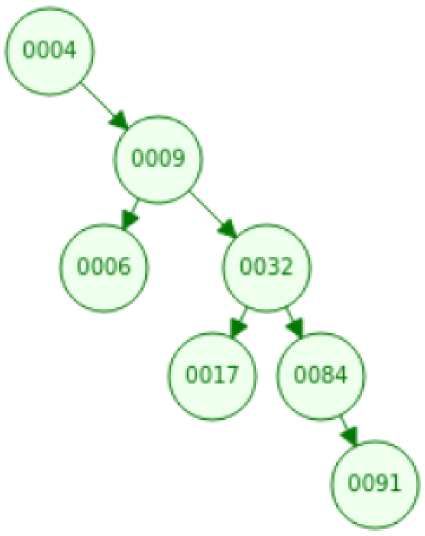
The first line contains the integer N ($0 \leq N \leq 10^4$), the number of elements in our series. The next line contains N space separated integers - A_i ($1 \leq A_i \leq 10^9$) - representing the series of numbers.

Output

Print space separated integers denoting the *postorder traversal* of the BST formed after processing all elements. *Note*: The number of elements in the output is not necessarily equal to the number of elements in the input series. This is because repeated elements in the input are ignored.

```
input
8
4 9 4 32 17 84 6 91

output
6 17 91 84 32 9 4
explanation
The final BST formed is shown on
the right.  We perform a postorder
traversal on that.
```



G. Bugs

There are N bugs on a horizontal string of length S units (from $x = 0$ to $x = S$). Each of the bugs is initially located at position x_i ($0 \leq x_i \leq S$) and each of the bugs is moving either to the left ($-X$ direction) or the right ($+X$ direction) with the same speed of 1 unit/second. When a bug reaches positions $x = 0$ or $x = S$, they immediately fall off the string. When two bugs travelling in opposite directions end up at the same position, they collide *elastically*, i.e. after collision, both bugs start moving in directions opposite to what they were originally moving in, with the same 1 unit/second speed. It is guaranteed that each bug is initially at a unique position, i.e. no two bugs share the same initial position. How long will it take for all the bugs fall off the string?

Input

The first line of input contains two integers N and $S(0 \leq N \leq 5 \times 10^5, 1 \leq S \leq 10^9)$. The next N lines contain two space space separated values - one integer and one character ("L" or "R"), representing the original position and original direction of the i^{th} bug.

Output

Print a single integer denoting the time at which all bugs have fallen off the string.

```
input
3 7
2 L
4 R
6 L

output
6
```

explanation

We have three bugs - B_0, B_1, and B_2. B_0 does not collide with any other bugs and falls off the left side of the table at $t = 2$. B_1 and B_2 collide at $t = 1$ at position $x = 5$. B_1 that starts going to the left and B_2 starts going to the right. B_1 falls off the left edge at $t = 6$, while B_2 falls off the right edge at $t = 3$.

H: Tower Defense (Part I)

You are sitting on a tower at the point $(0, 0)$ in a 2D plane. There are N enemies approaching the tower, with the locations of the enemies being $(x_1, y_1), (x_2, y_2) \ldots (x_N, y_N)$. You, a defender in the tower, have a laser beam that can shoot a laser in a straight line in any direction. One shot from the laser will obliterate all life on a line beginning at the tower $(0, 0)$ and extending in any direction. Calculate the minimum number of lazer shots required to eliminate all advancing enemies.

Input

The first line contains the integer N $(1 \leq N \leq 10^3)$, the number of enemies. The next N lines contain one coordinate each (x_i, y_i) (one indexed) representing the position of the i^{th} enemy. $(-10^8 \leq x_i, y_i \leq 10^8)$.

Output

Output one integer, the number of laser shots required.

```
input
2
1 0
-1 0

output
2
explanation
Two laser shots are needed since
both points are on opposite sides of
the tower.
```

```
input
5
3 4
0 9
-12 -36
9 12
-6 -18

output
3
explanation
(-12, -36) and (-6, -18) can be
eliminated in one shot.  Similarly
for (3,4) and (9,12)
```

I: Tower Defense (Part II)

You are sitting on a tower at the point $(0,0)$ in a 2D plane. There are N enemies approaching the tower, each of which is arriving from a certain angle, A_i, in a straight line towards the tower. You, the defender of the tower, have upgraded the laser allowing you to attack more enemies at once. Your new laser can attack everyone bounded between an arc of k degrees and the centre of the circle. Assume that one blast from the laser beam can affect *all* enemies located within the range of angles $[i, (i+k)\%360]$ (both extremes inclusive), for any integral $i < 360$. What is the *minimum* number of laser blasts needed to kill all enemies?

Additional thinking (not a part of this problem): Let us say the enemies are all wearing shields of varying strengths to protect themselves. Each i^{th} enemy now has a different health h_i. An enemy with health h requires h hits from the laser before falling dead. How would you solve this extension to the problem?

Note: Notice that with $k = 0$ this problem reduces to the previous one.

Input

The first line contains the integer N $(1 \leq N \leq 10^3)$, the number of enemies and k $(1 \leq k \leq 90)$. The second line of input contains N space separated integers denoting the angle of approaches, A_i $(0 \leq A_i \leq 360)$, of the advancing enemies.

Output

Output one integer, the number of laser shots required.

```
input
3 90
45 60 75
output
1
explanation
One shot of the laser is enough to
eliminate them all.
```

```
input
6 10
15 16 17 18 19 78
output
2
explanation
One shot of the laser is enough to
eliminate them all.
```

```
input
5 10
114 114 114 355 5
output
2
explanation
Enemies approaching from 355, 5 can
be eliminated with one shot.
```

```
input
3 45
13 58 178

output
2
explanation
Enemies approaching from 13 and 45
can be eliminated in one shot.
```

J. Inverting a Binary Tree

You are given a binary tree in the form of an array A, where $A[0]$ is the root. For any node $A[i]$, the left and right children are located in $A[2i+1]$ and $A[2i+2]$ respectively. All values in the binary tree are guaranteed to be positive integers. If a node does not exist at a certain index, the value -1 will be located in the array. Invert the binary tree by:

1. Start at the root node. Reverse the digits in the value at this node. For eg, if the value in root node is 123, change it to 321. Then, swap the positions of the entire right subtree with the entire left subtree.
2. Once the root node is processed, go to each of their children, and repeat this process of reversing the value in the node and swapping the positions of its children subtrees.
3. repeat this step all the way down to leaves. For each leaf, after reversing the value in each of the leaves, swapping its children (NULL and NULL) causes no change in the tree.

Output the inorder traversal of the final tree.*Note: after reversing the digits of a number, you need to preserve leading zeroes. i.e. reverse of 980 is 089.*

Input

The first line contains N ($1 \leq N \leq 10^4$), the number of elements in array A. The second line contains N space separated integers representing the binary tree ($A_i \leq 10^9$) in array format.

Output

Print space separated integers denoting the inorder traversal of the final tree.

```
input
7
123 45 92 70 60 -1 5

output
5 29 321 06 54 07

explanation
We notice that our input has 7 elements (one empty element denoted by -1).  However,
the number of integers in the output is equal to the number of elements in the tree,
6.
```
