DEEPAK AND HIS JOURNEY

Deepak loves to travel but he wants to minimize the total travel expenditure. His journey will be through N checkpoints. Checkpoints are numbered from 0 to N−1. At the start of his journey he is present at the checkpoint 0. Also checkpoint N−1 will lead to his final destination.

Each checkpoint has a petrol pump which can be used to fill petrol in the car. Now cost of petrol per litre at different checkpoints is given by array C which has 0-based indexing where C[i] is the cost per litre of petrol at the ith checkpoint. Note that there is an infinite amount of supply at each checkpoint and car tank is also of infinite capacity. Now another array L is given which has 0-based indexing where L[i] denotes the amount of petrol in litres required to reach the (i+1)th checkpoint from the ith checkpoint. Note that the extra petrol remaining in the tank does not vanishes after reaching a checkpoint. Also, Deepak should have atleast L[i] litres of petrol at each checkpoint in order to reach the next checkpoint.

Help Deepak to estimate the minimum cost required in order complete the journey.

Input Format

First line of the input contains test cases denoted by T.

For each of the test cases, first line contains a single integers N denoting the number of checkpoints.

The next line contains N-space separated integers representing the array C which has 0-based indexing where the integer denotes the cost per litre of petrol at ith checkpoint.

The last line contains array L, which has 0-based indexing, consisting of N space separated integers where the ith integer denotes the required amount of petrol needed to reach the (i+1)th checkpoint from the ith checkpoint.

Constraints

$1 \le T \le 5$
$1 \le N \le 10^5$
$1 \le C[i], L[i] \le 10^5$

Output Format

For each of the test cases, output the required answer in a separate line.

Sample Input

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

Sample Output

```
7
```

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int t  = sc.nextInt();

        for(int i=0; i<t; i++){
            int n = sc.nextInt();
            long [] c = new long[n];
            long [] l = new long[n];

            for(int j=0; j<n;j++){
                c[j] = sc.nextInt();
            }

            for(int j=0; j<n;j++){
                l[j] = sc.nextInt();
            }

            long Total_cost = 0;
            int k=0;
            long current_cost = c[0];
            long current_litre = 0;
            while(k<n){
                if(current_cost <= c[k]){
                    Total_cost += current_cost*l[k];
                }
                else{
                    current_cost = c[k];
                    Total_cost += current_cost*l[k];
                }
                k++;
            }

            System.out.println(Total_cost);
        }
    }
}
```

GUDDU BHAIYA JOINS KALEEN BHAIYA GANG

Kaleen bhaiya is a big gangster in mirzapur and runs his own gang.

Guddu bhaiya wants to join Kaleen bhaiya's gang. Kaleen bhaiya has many enemies. So to defeat those enemies Kaleen bhaiya wants to recruit someone who is logically strong.

So if Guddu bhaiya wants to join the gang he has to pass a test.

In the test he will be given an enemy name, and he has to spoil that name by performing a special operation query.

The operation query involves moving the last letter to the place of the first letter and shifting all other letters one position ahead.

He has to perform the operation on part of the name which starts with index **i** and ends with index **j**.

He has to perform this operation query on the part **k** number of times.

He will be given n such operation queries.

And then he will give back the spoiled name to Kaleen bhaiya after performing all n operation queries.

You have to help Guddu bhaiya to do the task.

Input Format

First line will contain the enemy name in the form string **name**
Second line will contain an integer **n** denoting number of operation queries Guddu bhaiya has to perform.
Each of the next n lines will contain three integers i, j and k.

Constraints

Output Format

Print the spoiled name after performing all n operation queries.

Sample Input

```
dedfded
2
3 6 1
1 4 2
```

Sample Output

```
eddefdd
```

Explanation

For the name "dedfded",

After performing 1st operation query Guddu bhaiya will get "deedfdd".

After performing 2nd operation query he will get "eddefdd"

```
import java.util.*;
```

```
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        String name = sc.next();
        char [] char_name = new char[name.length()];
        for(int i=0; i<name.length();i++){
            char_name[i] = name.charAt(i);
        }

        int operations = sc.nextInt();
        for(int o=0; o<operations; o++){
            int i = sc.nextInt();
            int j = sc.nextInt();
            int k = sc.nextInt();

            int new_j = j-1;
            int new_i =  i-1;
            while(k>0){
                j = new_j;
                i = new_i;
                char temp = char_name[j];
                while(j>i){
                    char_name[j] = char_name[j-1];
                    j--;
                }
                char_name[j] = temp;
                k--;
            }
        }

        String new_s = "";
        for(int m=0; m<char_name.length;m++){
            new_s+=char_name[m];
        }
        System.out.println(new_s);
    }
}
```

THIRD MAJORITY

In yet another Number Line Election, Integers have come together to elect a representative amongst themselves. In contrast to normal elections, they have decided that anyone who secures more than one-third of the votes shall be declared a representative.

You are given a list of integers denoting a vote for that integer. Find the representative(s).

Input Format

First line contains an integer N.
Second line contains N space separated integers.

Constraints

$1 <= N <= 10^4$
$|A_i| <= 10^9$

Output Format

Print the representatives on a newline.

Sample Input

```
8
1 1 2 2 3 1 3 3
```

Sample Output

```
1
3
```

Explanation

The integers 1 and 3 both occur thrice, i.e., more than 8/3 times and hence are elected as representatives.

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        int [] votes = new int[t];
        for(int i=0; i<t; i++){
            votes[i] = sc.nextInt();
        }

        Arrays.sort(votes);

        int i=0;
        int wins = t/3+1;
        while(i<t){
            int current = votes[i];
            int count = 0;
            while(i<t && current==votes[i]){
                count++;
                i++;
            }
            if(count >= wins){
                System.out.println(current);
            }
        }
    }
}
```

POLITE SORT

Dima has been assigned the task of arranging the books as per their identification number (an integer) on a shelf. She is very exhausted, so she asks one of her friends for help. Being polite she will only ask him to sort the shortest part (continuous) of the shelf such that if he only sorts all the books in that part, the whole shelf is sorted.
You are given an array A, of size N, where each element denotes the ID of the book on the shelf at that position. Find the minimum length of the shelf that Dima would ask his friend to sort to complete her task.

Input Format

First line contains an integer N.
Second line contains N space-separated integers.

Constraints

$1 <= N <= 10^5$
$|A_i| <= 10^9$

Output Format

Single Integer, denoting the length of the shelf that needs sorting.

Sample Input

```
6
2 4 3 8 7 9
```

Sample Output

```
4
```

Explanation

If only the books from 2^{nd} to 4^{th} position are sorted, all the books would be sorted

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] books = new int[n];
        int [] new_books = new int[n];
        for(int i=0; i<n; i++){
            books[i] = sc.nextInt();
            new_books[i] = books[i];
        }

        Arrays.sort(books);
        int start = -1;
        int end = 0;
        for(int i=0; i<n; i++){
            if(new_books[i] != books[i]){
                end = i;
                if(start == -1){
                    start = i;
                }
            }
        }

        if(start != -1){
            System.out.println(end-start+1);
        }
        else{
            System.out.println(0);
        }
    }
}
```

GOOD SEQUENCE

You are given an array A, of size N representing a sequence. The sequence is considered to be Good if no element is greater than its next. However, if you are allowed to modify at most one element. Would you be able to make the sequence Good?

Input Format

First line contains an integer N.
Second line contains N space-separated integers.

Constraints

$1 <= N <= 10^4$
$|A_i| <= 10^5$

Output Format

"true" if the sequence can be Good, else "false".

Sample Input

```
4
3 1 2 6
```

Sample Output

```
true
```

Explanation

We can modify 3 to 0 to make the sequence Good

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

         int []sequence = new int [n];
         for(int i=0; i<n; i++){
             sequence[i] = sc.nextInt();
         }

         int count = 0;
         for(int i=1; i<n; i++){
             if(sequence[i-1] > sequence[i]){
                 count++;
                 if(i-2 >= 0 && sequence[i-2] > 0){
                     count++;
                 }
                 if(count > 1){
                     break;
                 }
             }
         }

         if(count <= 1){
             System.out.println(true);
         }
         else{
             System.out.println(false);
         }
    }
}
```

KEB'S PARTICLE

Keb is an experimental physicist. His works primarily relate to the field of Subatomic particles.
Keb is working on behavior of certain particles, which he calls **Z-particles**. These particles tend to interact with every other particle in its group. Keb worked out that, similar to electrons experiencing the *Screening Effect* in an atom, every other Z-particle experiences a similar effect. If he places N of these particles in a contained environment where each Z-particle has some amount of charge A_i, then the effect on i^{th} particle is the product of the charge of every other particle in the system.

Keb has set up a system which has N number of Z-particles. An array A, of size N is given where A_i is the charge on the i^{th} particle. Find the effect on each of the Z-particles in the given system.

Input Format

First line contains an integer N.
Second line contains N space separated integers, denoting the charge $\mathbf{A_i}$.

Constraints

$1 <= N <= 10^5$
$|A_i| <= 100$

Output Format

Print the effect on the particles of the system in a space-separated manner.

Sample Input

```
3
1 2 3
```

Sample Output

```
6 3 2
```

Explanation

The System has 3 particles, wherein effect on 1^{st} particle is $2 \times 3 = 6$, on 2^{nd} particle is 1×3, and on 3rd particle is 1×2, giving the output as 6, 3, 2.
Note: Multiplying all the charges and dividing the product by the charge for i^{th} Z-particle, to calculate the effect on it, may give unexpected results.

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int[n];
        for(int i=0; i<n;i++){
            arr[i] = sc.nextInt();
        }

        int []l_arr = new int[n];
        l_arr[0] = 1;
        for(int i=1; i<n; i++){
            l_arr[i] = l_arr[i-1]*arr[i-1];
        }

        int  [] r_arr = new int[n];
        r_arr[n-1] = 1;
        for(int i=n-2; i>-1; i--){
            r_arr[i] = r_arr[i+1]*arr[i+1];
        }

        for(int i=0; i<n; i++){
            System.out.print(l_arr[i]*r_arr[i]+" ");
        }
    }
}
```

EMILY BIRTHDAY

It's Emily's birthday today, so all her friends brought candy for her. It's customary at her school that every classmate of hers gives her one unique candy. Some of her classmates were absent on that day, so Emily's best friend, being mischievous, snuck some extra candies in place of those friends into her Candy Bag.
You are given the contents of Emily's Candy Bag, could you find which candies are from her best friend?

Input Format

First line contains an integer N, the number of candies in Emily's Bag.
Second line contains N space separated integers A_i, representing a candy from the person A_i.

Constraints

$1 <= N <= 10^5$
$1 <= A_i <= N$

Output Format

Print the value that represents the candies from Emily's best friend.

Sample Input

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

Sample Output

```
2
```

Explanation

As there are two candies numbered 2, Emily's best friend gave her candy numbered 2.

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] candies = new int[n];
        for(int i=0; i<n; i++){
            candies[i] = sc.nextInt();
        }
        Arrays.sort(candies);
        for(int i=1; i<n; i++){
            if(candies[i-1]==candies[i]){
                System.out.print(candies[i]);
                break;
            }
        }
    }
}
```

CORNERS

Corners is a game played in Eastern states of India.
It's usually played with 4 or more players, where each player occupies a spot, representing corners of a polygon. A player who's chosen to be the **Tat** must go to a player at one of the corners, who will then redirect him/her to one of the other corners. While the **Tat** is on his/her way to the redirected corner, others try to exchange their spot and Tat would try to take it as an opportunity to capture the corner, leaving one of the others as the new Tat, similar to *Musical Chairs*.
Mili was one day playing Corners with her friends, and a thought struck her mind. If all the players holding the corners make up their mind, and have decided where to redirect the Tat, then it may be possible that the Tat is stuck in a loop (or cycle) indefinitely.
Find the loop having the maximum number of corners.
Note: A player may redirect the Tat to himself/herself, representing a loop involving a single corner.

Input Format

First line contains an integer N, denoting the number of corners.
Next line contains N space separated integers, from the range [0, N-1], where each integer represents the index (or position), where that player would redirect the Tat.
Note: Each value is unique.

Constraints

$1 <= N <= 10^5$
$1 <= A_i < N$

Output Format

Print the number of corners in the longest loop.

Sample Input

```
7
5 4 0 3 1 6 2
```

Sample Output

```
4
```

Explanation

The player at index 0 would redirect the Tat to index 5 (A_0 = 5), where he/she would be redirected to index 6 (A_5 = 6). Then from index 6, the Tat would be redirected to index 2 (as A_6 = 2). From the index 2, he/she would be redirected to index 0 (A_2 = 0) bringing him to the place he started.
The loop forms a cycle as, 0 -> 5 -> 6 -> 2 and then back to 0. The length is 4.

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int n= sc.nextInt();

        int[] directions = new int [n];
        for(int i=0; i<n; i++){
            directions[i] = sc.nextInt();
        }

        int start = 0;
        int total = 0;
        int count = 0;
        while(start < n){
            int last = directions[start];
            count = 1;
            while(start != last){
                last = directions[last];
                count++;
            }
            if(count > n/2){
                System.out.println(count);
                break;
            }
            else{
                total = Math.max(total,count);
            }
            start++;
        }

        if(count <= n/2){
            System.out.println(total);
        }
    }
}
```

LOCAL MAXIMA

You may have studied about Local Maxima when working with Functions and its graph on the Cartesian Plane. A Local Maxima is defined here as a point on the graph whose value is strictly greater than its neighbors.
You are given an array A, of size N. Find the position of Local Maxima among the given values.
NOTE: The first and last element need only be compared to the element next and prior to them respectively.
Try to find the local maxima in O(log N) time.

Input Format

First line contains an integer N. Second line contains N space separated integers.

Constraints

$1 <= N <= 10^4$
$1 <= A_i <= 10^9$

Output Format

Print the index of the Local Maxima.

Sample Input

```
4
1 2 3 1
```

Sample Output

```
2
```

Explanation

Element at index 2 is greater than element at indices 1 and 3.

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] values = new int [n];
        for(int i=0; i<n; i++){
            values[i] = sc.nextInt();
        }

        if(n>1){
        if(values[0]>values[1]){
            System.out.println(0);
        }
        else if(values[n-1]>values[n-2]){
            System.out.println(n-1);
        }
        else{
            for(int i=1; i<n-1;i++){
                if(values[i] > values[i-1] && values[i] >
values[i+1]){
                    System.out.println(i);
                    break;
                }
            }
        }
        }
        else{
            System.out.println(values[0]);
        }
    }
}
```