Radiation

Martin has a few chemical elements stored in his cupboard. These elements are put in a single line, so that they are placed next to each other. Each element is either on the left or right of some other element.

After one year storing the elements, to Martin's surprise, some of the elements are gone! It turns out that the elements that Martin stores are no ordinary elements. These elements can emit radioactive waves and affect other elements. If an element is more radioactive than the element located at the left of it, it decays.

Being a curious fellow, Martin wants to know how many years will it take until no element will decay. Therefore, he asks his brother, Kevin, a computer science student, to help him create a program that can automatically give the answer to the million-dollar question: "how many years will it take until no elements in Martin's cupboard decay"? That is, for all elements left in the cupboard, there are no elements that are located to the right of a less-radioactive element.

Good luck!

Input

The first line contains a single integer **N** ($1 <= N <= 1{,}000{,}000$), the number of chemical elements inside Martin's cupboard. The next line contains N integers $a_0 \ \ldots \ a_{n-1}$ where $0 <= a_i <= 10$, each representing the radioactivity of the element at the i^{th} position, from the leftmost to the rightmost element.

Output

Print the **minimum** number of years until no chemical element decays as described above.

Sample Input 1

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

Sample Output 1

```
1
```

Sample Input 2

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

Sample Output 2

```
0
```

Sample Input 3

```
7
6 5 8 4 7 10 9
```

Sample Output 3

```
2
```

Explanation

Sample Input 1: **1 year**

1. In the first year, the element with strength "5" will decay since it is more radioactive than the one on its left ("4"). In the same year, the element with strength "4" will also decay because of it being more radioactive than the one on its left ("3"), and so on. After 1 year, we're left with a single element, the element with strength "1".

Sample Input 2: **0 year**

1. No chemical element will cause another element to decay.

Sample Input 3: **2 years**

1. In the first year, the elements with strengths "8", "7", and "10" will decay because they are located **at the right of a weaker element (they are more radioactive than the one on their left).**
2. At the start of the second year, our elements are like this: 6 5 4 9. This year, the element with strength "9" will decay as it is located at the right of a less radioactive element. It is more radioactive than the one on its left, hence it decays.
3. No more elements will decay after the second year.

Skeleton

You are given the skeleton file `Radiation.java`. The class "Element" is provided as a hint to get the **O(N)** solution.

```
/**
 * Name         :
 * Matric No.   :
 * PLab Acct.   :
 */

import java.util.*;

public class Radiation {

    public void run() {
        // implement your "main" method here...
    }

    public static void main(String[] args) {
        Radiation myChemicalElements = new Radiation();
        myChemicalElements.run();
    }
}
```

```
class Element {
    private int strength;
    private int yearsBeforeDecay;

    public Element(int strength, int yearsBeforeDecay) {
        this.strength = strength;
        this.yearsBeforeDecay = yearsBeforeDecay;
    }

    public int getStrength() {
        return this.strength;
    }

    public int getYearsBeforeDecay() {
        return this.yearsBeforeDecay;
    }
}
```

Notes:

1. You must either use **stack** or **queue** to solve this problem, whichever is suitable.
2. Your program might give out the correct answer, but killed on CodeCrunch. It means that your program is not efficient enough (i.e. it runs too slow). You should design a more efficient algorithm to solve this problem if this is the case. Consider the above note as a hint.