Poisonous Plants

There are a number of plants in a garden. Each of the plants has been treated with some amount of pesticide. After each day, if any plant has more pesticide than the plant on its left, being weaker than the left one, it dies.

You are given the initial values of the pesticide in each of the plants. Determine the number of days after which no plant dies, i.e. the time after which there is no plant with more pesticide content than the plant to its left.

Example

$p = [3, 6, 2, 7, 5]$ // pesticide levels

Use a 1-indexed array. On day 1, plants 2 and 4 die leaving $p' = [3, 2, 5]$. On day 2, plant 3 in p' dies leaving $p'' = [3, 2]$. There is no plant with a higher concentration of pesticide than the one to its left, so plants stop dying after day 2.

Function Description

Complete the function *poisonousPlants* in the editor below.

poisonousPlants has the following parameter(s):

- *int p[n]*: the pesticide levels in each plant

Returns
- *int*: the number of days until plants no longer die from pesticide

Input Format

The first line contains an integer n, the size of the array p.
The next line contains n space-separated integers $p[i]$.

Constraints

$1 \leq n \leq 10^5$
$0 \leq p[i] \leq 10^9$

Sample Input

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

Sample Output

```
2
```

Explanation

Initially all plants are alive.

Plants = {(6,1), (5,2), (8,3), (4,4), (7,5), (10,6), (9,7)}

Plants[k] = (i,j) => j^{th} plant has pesticide amount = i.

After the 1^{st} day, 4 plants remain as plants 3, 5, and 6 die.

Plants = {(6,1), (5,2), (4,4), (9,7)}

After the 2nd day, 3 plants survive as plant 7 dies.

Plants = {(6,1), (5,2), (4,4)}

Plants stop dying after the 2nd day.

Contest ends in 2 months

Submissions: 88
Max Score: 10
Difficulty: Hard

Rate This Challenge:

More

Java 8

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;

class Result {

    /*
     * Complete the 'poisonousPlants' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY p as parameter.
     */

    public static int poisonousPlants(List<Integer> p) {
    // Write your code here
        int max=0;
        int day=0;

         Stack<ArrayList<Integer>> stack=new Stack<>();

            for (int i=0;i<p.size();i++){
                 day=0;
                int plant=p.get(i);
                while(!stack.empty() && stack.peek().get(0)>=plant ){
                        day=Math.max(day,stack.pop().get(1));
                }
                if(!stack.empty()){
                    day++;
                }
                else{
                    day=0;
                }
                 max=Math.max(day,max);
                 ArrayList<Integer> arr=new ArrayList<Integer>();
                 arr.add(plant);
                 arr.add(day);
                 stack.push(arr);
                 //System.out.println(stack);
                 // System.out.println(arr);
            }
    return max;
```

```
50
51
52
53      }
54
55  }
56
57  public class Solution {
58      public static void main(String[] args) throws IOException {
59          BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
60          BufferedWriter bufferedWriter = new BufferedWriter(new
    FileWriter(System.getenv("OUTPUT_PATH")));
61
62          int n = Integer.parseInt(bufferedReader.readLine().trim());
63
64          List<Integer> p = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))
65              .map(Integer::parseInt)
66              .collect(toList());
67
68          int result = Result.poisonousPlants(p);
69
70          bufferedWriter.write(String.valueOf(result));
71          bufferedWriter.newLine();
72
73          bufferedReader.close();
74          bufferedWriter.close();
75      }
76  }
77
```

Line: 27 Col: 10

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔ | Testcase 1 ✔ | Testcase 2 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

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

Your Output (stdout)

```
2
```

Expected Output

```
2
```