Poisonous Plants

locked

by vatsalchanana

There are a number of plants in a garden. Each of these 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. Print the number of days after which no plant dies, i.e. the time after which there are no plants with more pesticide content than the plant to their left.

For example, pesticide levels $p = [3, 6, 2, 7, 5]$. Using a 1-indexed array, day 1 plants 2 and 4 die leaving $p = [3, 2, 5]$. On day 2, plant 3 of the current array dies leaving $p = [3, 2]$. As there is no plant with a higher concentration of pesticide than the one to its left, plants stop dying after day 2.

Function Description

Complete the function *poisonousPlants* in the editor below. It must return an integer representing the number of days until plants no longer die from pesticide.

poisonousPlants has the following parameter(s):

- *p*: an array of integers representing pesticide levels in each plant

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

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

Output Format

Output an integer equal to the number of days after which no plants die.

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)}

After the 2nd day the plants stop dying.

Submissions: 51
Max Score: 18

Rate This Challenge:

More

Current Buffer (saved locally, editable) C

```c
#include <assert.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char** split_string(char*);

// Complete the poisonousPlants function below.
int poisonousPlants(int p_count, int* p) {


}

int main()
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    char* n_endptr;
    char* n_str = readline();
    int n = strtol(n_str, &n_endptr, 10);

    if (n_endptr == n_str || *n_endptr != '\0') { exit(EXIT_FAILURE); }

    char** p_temp = split_string(readline());

    int* p = malloc(n * sizeof(int));

    for (int i = 0; i < n; i++) {
        char* p_item_endptr;
        char* p_item_str = *(p_temp + i);
        int p_item = strtol(p_item_str, &p_item_endptr, 10);

        if (p_item_endptr == p_item_str || *p_item_endptr != '\0') { exit(EXIT_FAILURE); }

        *(p + i) = p_item;
    }

    int p_count = n;

    int result = poisonousPlants(p_count, p);

    fprintf(fptr, "%d\n", result);

    fclose(fptr);

    return 0;
}
```

```
54
55 char* readline() {
56     size_t alloc_length = 1024;
57     size_t data_length = 0;
58     char* data = malloc(alloc_length);
59
60     while (true) {
61         char* cursor = data + data_length;
62         char* line = fgets(cursor, alloc_length - data_length, stdin);
63
64         if (!line) { break; }
65
66         data_length += strlen(cursor);
67
68         if (data_length < alloc_length - 1 || data[data_length - 1] == '\n') { break; }
69
70         size_t new_length = alloc_length << 1;
71         data = realloc(data, new_length);
72
73         if (!data) { break; }
74
75         alloc_length = new_length;
76     }
77
78     if (data[data_length - 1] == '\n') {
79         data[data_length - 1] = '\0';
80     }
81
82     data = realloc(data, data_length);
83
84     return data;
85 }
86
87 char** split_string(char* str) {
88     char** splits = NULL;
89     char* token = strtok(str, " ");
90
91     int spaces = 0;
92
93     while (token) {
94         splits = realloc(splits, sizeof(char*) * ++spaces);
95         if (!splits) {
96             return splits;
97         }
98
99         splits[spaces - 1] = token;
100
101         token = strtok(NULL, " ");
102     }
103
104     return splits;
105 }
106
```

Line: 1 Col: 1

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