Problem E. Classics

Time limit: 1 second
Memory limit: 512 megabytes

You are probably familiar with the classic problem of finding the longest increasing subsequence in an array. Let a be an array consisting of n integers. A subsequence $i_1 < i_2 < \ldots < i_k$ is called *increasing* if $a_{i_1} < a_{i_2} < \ldots < a_{i_k}$. The longest increasing subsequence is the increasing subsequence of maximum length. Of course, we will not ask you to solve the classic problem; you will have to solve its more complicated version...

Initially, there is an empty array a. Then, the numbers $1, 2, \ldots, n$ are added to the array in this order. The number i is added to the array at position p_i. Positions in the array are numbered with integers from 1 to k, where k is the current size of the array. When adding an element at position p in an array of size k, all elements that previously had positions from p to k are shifted one position to the right, and the current element is added to the freed space.

Your task is to determine the length of the longest increasing subsequence in the array after each addition of a new element.

Input

The first line contains one integer n ($1 \le n \le 200\,000$) — the number of added elements.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le i$) — p_i denotes the position where element i is added.

Output

Output n integers — the length of the longest increasing subsequence of the array after each addition of a new element.

Examples

standard input	standard output
5 1 2 1 3 4	1 2 2 2 3
1 1	1

Note

The array in the first example changed as follows: $[] \to [1] \to [1, 2] \to [3, 1, 2] \to [3, 1, 4, 2] \to [3, 1, 4, 5, 2]$.