Forming societies

A long street has n buildings in a row, located on one side of the street, numbered sequentially from 0 to $n-1$. The ith building has f_i flats, for $0 \leq i < n$. Some of the building residents want to join together to form a co-operative housing society. A restriction imposed is that buildings in a society must occur consecutively along the street, that is, should be numbered from i to j for some $0 \leq i \leq j < n$. In a society, the building that has more flats than any other building in the society has control of the society. If two buildings have equal number, neither has control. You have to find for each building i, the number of distinct societies in which it will have control. More formally, given a sequence of numbers $f_0, \ldots, f_{n-1}$, for each i, you have to find the number of substrings $f_j, \ldots, f_k$ such that $j \leq i \leq k$ and f_i is the unique maximum element in the substring.

A society of buildings from i to j, for $i \leq j$, is said to be well-formed if building i has minimum number of flats in the society and building j has the maximum number. The maximum and minimum may not be unique in this case. In other words, f_i is the minimum of the substring $f_i, \ldots, f_j$ and f_j is the maximum. A society with a single building is always well-formed. In the second part of the problem, you have to compute the total number of non-empty well-formed societies possible.

Input/ Output The first line of input will specify n, the number of buildings, which will be at most 10^6. The next line will contain n positive integers giving the number of flats in each building. Each number will be at most 1000 and the numbers will be separated by a single space. The first line of output should contain n numbers, the ith of which is the number of societies the ith building can control. The numbers should be printed on a single line separated by a space. The next line should contain the total number of well-formed socities. Note that the numbers may be large so use long long int type.

Sample Input	Sample Output
4	2 1 6 1
2 1 3 2	5

Note: For the second part, it may help to use the vector class from the C++ standard library. This is essentially an implementation of the sequence abstract data type, and allows many operations. The functions most useful here would be the push_back function that adds an element at the end of the sequence, and the lower_bound function defined in the algorithm class, that finds the smallest element in a vector that is not less than a given element. You can read details about this online, at for example C++.com, in case you are not familiar with vectors. You can also do it with just arrays, but may have to implement binary search in that case. This is not required for the first part. You will get half credit if you do the first part correctly. Use the algorithm in the last quiz for this.

Submission: Submit a single file named RollNo_6.cpp