Largest Elements (left)

Given a vector of values each of which is larger than zero, for each value, find the closest element that comes before it and is larger than it. For example, consider the following vector (in the following, I will refer to it as the input vector.)

originalVector.png

For this input vector, your function should produce the following vector (you will be given this vector. Your job is to store values in it. In the following, I will use output vector to refer to this vector.)

largerLeft.png

A few notes...

The first element of the output vector is 0 because the first element of the input vector, 10, is not preceded by any values that is larger than it. The second element of the output vector is 10 because 10, in the input vector at index 0, is the closest element that comes before 5 (the second element of the input vector) that is larger than it. Likewise, the third element of the output vector contains 10. That is because the values of the elements that come before 9, the third element of the input vector are 5 and 10 (moving backwards from index 2 towards the beginning of the array.) The first of these two values that is larger than 9 is 10 and as a result, we have stored 10 in the third element of the output vector.
Element at index 6 of the output vector contains 200. This is because to find the element for index 6 of the output vector, we first look at the element at index 5 of the input vector and it contains 50, which is not larger than 70, the element at index 6 of the input vector. The next element of the input vector that we inspect is the one with index 4, which contains 200. 200 is larger than 70 and as a result, it is the first element, going backwards, that comes before 70 and is larger than 70. Therefore, we use it at index 6 of the output vector.
The input vector will not contain any zeros. Any element of the input vector that doesn't have a value that comes before it and is larger than it will produce a zero in the output vector. In the above example, value 70, in the input vector at index 3, is preceded by 9, 5, and 10 (in the order that you will see them as you go backwards.) Neither of these values is larger than 70. As a result, we have stored 0 in the corresponding element (element 4 at index 3) of the output vector. For this very reason we have stored zeros for input values 200, 300, 310, 360, and 400 in the output vector.
Here is a more formal description of this problem.

Base Name: largest_left

Function prototypes

void largest_left_iterative(const std::vector<int> &numbers, std::vector<int> &result);

```
void largest_left_recursive(const std::vector<int> &numbers, std::vector<int> &result, int n);
```

Both function prototypes are stored in project2.hpp . The first argument in both cases is a vector that contains the input values. The second argument is an empty vector; you store the result of your computation in this vector by use of vector's push_back function.

In both functions, numbers.size() will produce the number of elements in "numbers". In the recursive version, n is initially numbers.size(). However, in the subsequent calls to the recursive function, argument n helps to reduce the number of elements under consideration. We have used this technique during the lecture frequently.