CMPUT 275 - Tangible Computing
Interview Problem: Leaders

Description

A **leader** in an array of integers is an entry that is greater than all other entries that come after it. Specifically, we say that $a[i]$ is a leader in an array $a[]$ if $a[i] > a[j]$ for all other indices $j > i$ in the array.

The **group** of a **leader** at index i is the longest contiguous subarray starting at index $i + 1$ that does not contain any leaders.
e.g. Given a leader at i, if the next leader after i is j, the group of i is $a[i+1], \ldots, a[j-1]$.

Note, a group **can** be empty if consecutive entries in the array are leaders (e.g. $a[i]$ and $a[i + 1]$ are both are leaders, so the group of i is empty).

Finally, we also have a group before the first leader. That is, if the first leader is at index i then $a[0], \ldots, a[i-1]$ is a group (which could be empty if the first leader is at index $i = 0$).

Given an array of n integers, find all leaders and reverse all groups.

The Reverse Function

You may create as many functions as you like, but you **must** have the following function:
`void reverse(int* begin, int* end);`

The function `reverse` takes two pointers that are pointing into an array $a[]$ and reverses all entries between these pointers including the entry pointed to by `begin` but **not** the entry pointed to by `end`.

For example, consider the following snippet of code:

```
int a[] = {1, 2, 3, 4, 5, 6, 7};
reverse(a+2, a+6);
// now the array a is {1, 2, 6, 5, 4, 3, 7}
```

You may need to review pointer arithmetic. Note, we can also compare pointers via operators like `<` and `==`. For example, if `ptr1` and `ptr2` are both pointers then `ptr1 < ptr2` will evaluate to true if and only if `ptr1` is a lower memory address than `ptr2`. You may assume this function will only be called if `ptr1 <= ptr2` (i.e. no need to perform error handling). In particular, if this function is called with `ptr1 == ptr2` then nothing should be done because the range of items to be reversed is empty.

You will certainly want to build and test this function separately before integrating it into your solution to the interview question.

Restrictions

For full marks, you **must** create your own `reverse` function specified as above, you may not use the standard `reverse` function located in the C++ standard template library (if you are aware of this). More generally, you **cannot include any file** other than `<iostream>`.

Input

The first line of input will contain a single integer $1 \leq n \leq 100,000$, the length of the array. The second line of input will contain n space separated positive integers, each no larger than $1,000,000$.

Output

The first line of output should contain a list of all leaders in the order they appear within the initial array.

The second line of output should contain the space separated representation of the edited array after all groups are reversed.

Sample Input 1

```
4
8 2 1 8
```

Sample Output 1

```
8
1 2 8 8
```

Explanation:

The only leader is at index $i = 3$, therefore the group from $a[0], \ldots, a[2]$ must be reversed.

Sample Input 2

```
10
12 17 17 4 8 3 9 5 1 2
```

Sample Output 2

```
17 9 5 2
17 12 17 3 8 4 9 5 1 2
```

Grading Comments

Despite the fact this appears similar to a morning problem, it will be graded like a weekly exercise. In particular:

- Style matters. Use appropriate comments, proper indentation, etc. Include a file header. Consult the style guide on eClass.
- You must use the function signature `void reverse(int* begin, int* end)`. You may change the names of `begin` and `end` if you wish, but their type and the return type of the function must be as we state. Deviating from this will result in a deduction.

- You must adhere exactly to the output specification: for example, if you output in the wrong order or print extra whitespaces then you will receive a deduction. The test centre must accept the output without any presentation error.
- The only functionality of the C++ standard library you may use is those pertaining to input and output (e.g. `cin`, `cout`, and `endl` located within `<iostream>`). Including any other file will result in a deduction.
- You were only given a few test cases in the test centre files on eClass. We will test your solution on additional test cases that adhere to the input specification.
- Partial credit may be obtained if your solution works on some inputs but not all inputs in the described range.
- Adhere closely to the submission instructions for the weekly exercise. See the eClass code submission link for details.