**Introduction Algorithms and Data Structures (Alternative Assignment)**

## Assignment 1 (arrays):

Implement the following algorithm:

Given an array of **N** numbers a1,a2,a3........an, find all numbers that are not smaller than the rest of the numbers on the right side of the array.

#### Input Format

Input will be read from the standard input (stdin) and will have the following format:

First line of input contains single value **N**, size of list  
Second line of input contains **N** space separated integers

*Tip: You can read standard input with scanf function.*

#### Output Format

Output all found integers separated in the array from left to the right that are not smaller than those on the right side

#### Constraints

* 0 < N <= 1000000
* 0 <= ai <= 10000

#### Example

*Input:*

6

16 17 4 3 5 2

*Output:*

17 5 2

*Explanation:*

17 is not smaller than 4 3 5 2

5 is not smaller than 2

2 is the last number and is also put in the array

*Build/Delivery:*

To implement and test all sub-assignments of assignment1 use the provided ass1.zip file. In the folder 1 you can find template file ass1.c for this assignment.

This file includes a small framework for test functionality (don’t change that) and comments that give you hint how to implement this first assignment.

You can build your implementation by running ‘make’ and remove temporary files by running ‘make clean’.

*Testing:*

To test your code you can pass the provided test files as standard input e.g. like this:

*./ass1* < *in1\_1*

If you want to run all tests (testing with all included test files) run

*make test*

If you are interested in achieved performance please run

*make perf\_tests*

This will include also the times that your algorithm takes to execute with the provided test files. This can be used to evaluate the most time-efficient implementation.

## Assignment 2 (sorting research):

This assignment is extension to the assignment that you have already done. The only difference is that you should now implement and research 2 more algorithms, so there will be 5 of them.

The text of the original assignment for 3 algorithms:

You are going to research 3 different sort of sorting algorithms.

You should use algorithms with at least 2 different big O values.

To study different sorting algorithms you can use for example the following links or any other reliable source.

<https://en.wikipedia.org/wiki/Sorting_algorithm>

<https://betterexplained.com/articles/sorting-algorithms/>

You are going to run all implemented algorithms for arrays with these numbers of elements:

1. N = 10
2. N = 1000
3. N = 10000

Document used algorithms, motivation of their choice and which sources were used for study. Compare the time performance of the chosen algorithms and their Big O values for different number of elements N. Document comparison and make conclusion on your research.

*Build/Delivery:*

To implement and test all sub-assignments of assignment1 use the provided ass1.zip file. In the folder 3 you can find template file ass3.c for this assignment.

This file includes a small framework for test functionality (don’t change that) and empty functions alg1, alg2 and alg3 where you should put your implementation of the sorting algorithms.

Feel free to use internet to choose your sorting algorithm but please, don’t use library functions (like qsort).

You can build your implementation by running ‘make’ and remove temporary files by running ‘make clean’.

*Testing:*

To test your code just run

*make test*

This will run all three algorithms with all different values of N with a provided test file (*in2*). It will check the output of the sorting algorithms and will show you time performance for different N numbers and different algorithms. You can use these performance numbers for your document.

### Final Delivery:

For the sake of automatic testing, please deliver your code in ass1.zip file with exactly the same structure as it was provided to you. Only the files ass1.c and ass2.c should be of course completed. Before delivery, please run ‘make clean’ on the highest level (top directory), so you don’t deliver any temporary product files.

The second part of the delivery is your updated research document for sub-assignment 2.