Course 'Imperative Programming' (IPC031)
Bonus Assignment 9: *Insertion sort reloaded*

1. Assignment

Part 1: Another insertion sort
The *insertion sort* algorithm as explained in the lecture uses the *insert* function to put the element-to-be-properly-sorted in the right place in the already sorted front part of the array / vector. The *insert* function seeks the proper location by starting at the *beginning* of the front part of the array / vector. As a consequence, if the element-to-be-properly-sorted happens to be larger than the current last element of the sorted front part, all elements of the sorted front part are compared with the element-to-be-properly-sorted.
To eliminate this effect, adapt the *insert* function in such a way that it starts at the *end* of the sorted front part of the array / vector instead of its beginning. Verify that the new version of *insertion sort* yields the same results when compared with your implementations of the mandatory assignment **2**, using both versions of comparisons.

Part 2: Redo part 4 of mandatory assignment
As in **part 4** of the mandatory assignment **2**, redo the counting experiments of the new *insertion sort* algorithm with the two versions of comparisons.
Compare the outcomes with the results that you have obtained for the mandatory assignment **2**. Do you experience significant differences? If so, explain them. Document the outcomes and your explanations by means of comments in *"main.cpp"*.

Part 3: Redo part 5 of mandatory assignment
As in **part 5** of the mandatory assignment **2**, redo the visualization experiments for the new *insertion sort* algorithm with the two versions of comparisons. Document the outcomes by means of comments at the end of *"main.cpp"*.

2. Products
As product-to-deliver you only need to upload to Brightspace *"main.cpp"* that you have created with solutions for each part of the assignment.

⇒ **Deadline:** Tuesday November 20 2018, 8:30h.