Due date: *October 17, 2008 at 18:00 CEST*

Instructions. This is an individual assignment. You may discuss it with others, but your formulations, your code, and all the required material must be written on your own. Write your solutions (formulations, graphs, etc.) in a single PDF file. Package this file along with all your program source files in a TAR archive file, and submit that archive following the instructions on the course Web page.

Exercise 1. Consider the following sorting algorithms described in class:

- INSERTION-SORT
- SELECTION-SORT
- BUBBLE-SORT
- MERGE-SORT

1. Implement the algorithms in Python. The implementation must read a sequence of numbers from the standard input, with one number per line, and it must output the same sequence in sorted ascending order.
2. Compare the performance (execution times) of your implementations on random sequences. You can generate the random sequences using the script `random_sequence.py` provided on the course Web page.

   Create a graph with the results of your comparison in the following way:

   - X-axis shows the number of elements in the input sequence;
   - Y-axis (for each algorithm implementation) shows the execution time representing the median of 10 execution times.

   The range of Y-axis should be up to 60 seconds. Therefore you should extend your experiments along the X-axis so as to reach execution times of up to 60 seconds.
3. Implement the algorithms in Java.
4. Compare the performance of the implementations as described in 2.

Suggestions. You may use the *time* utility to measure the execution time of your program. You should use the *gnuplot* tool to produce the graphs. The course Web page provides an example on how to use gnuplot. You should combines the two graphs (one for Python, one for Java) in a single LaTeX document.