Introduction: In the lab-classes this week, we will look at an implementation of the **Merge-Sort** algorithm. We will look at its runtime and compare it to the runtime of the implementation of the **Insertion-Sort** algorithm from the previous week. Again, we are considering as inputs upwards sorted lists, downwards sorted lists and random lists.

Update the environment:
- Login to your Linux account, open a terminal and switch to your home directory.
- If you still have the directory `cd CS-242-Algorithms` from last week in your home directory, change to it and enter at prompt
  `git pull`
- If you do not have the directory `CS-242-Algorithms`, perform the steps from the previous week: Enter at prompt
  `git clone git://github.com/OKullmann/CS-242-Algorithms.git`
- Change to the subdirectory for week 2:
  `cd ~/CS-242-Algorithms/201112/Week02/`

Basic setup: The main file for this week is `Sorting.java` containing the implementation of the **Insertion-Sort** algorithm, and an implementation of **Merge-Sort**. As a start we will compile the programs by running `make` in this week’s directory.

Making Experiments: The task for this lab session is to run some experiments on upwards sorted, downwards sorted, and random lists, using both algorithms. For this, run the different experiments for `N = 2000` (N being the input to the experiments) and explore the data which the experiment is outputting using the program `R`, by plotting for the same type of data the running times of insertion versus merge sort.

Think about the following questions, referring to the plot for the two sorting algorithms on random lists.
- Can you explain the two graphs using your knowledge about upper bounds on the running time from the lectures?
- Can you give ranges of sizes of inputs on which you would prefer one algorithm over the other?

Show the plots of your data (as produced by `R`) to the postgrads. Discuss your answers to the above questions with them.

For the ambitious student If you have time left after finishing the above tasks, you could try to extend the experiments by implementing another sorting algorithm of your choice. For your convenience, here is the pseudo code of a popular sorting algorithm, called **Bubble Sort**, which you could implement:
### BUBBLE-SORT(A)

1. swapped = FALSE
2. repeat
   3. for i = 2 to A.length
   5. swap(A[i - 1], A[i])
   6. swapped = TRUE
3. until not swapped

### Example Session

```
~> cd 201011/CS-242-Algorithms/
~/CS-242-Algorithms/201011> git pull
~/CS-242-Algorithms/201011> cd Week02/
~/CS-242-Algorithms/201011/Week02> make
~/CS-242-Algorithms/201011/Week02> java Experiment_MergeSort_rand 10
# Merge-sort on random lists
<table>
<thead>
<tr>
<th>size</th>
<th>executionTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6827</td>
</tr>
<tr>
<td>1</td>
<td>5628</td>
</tr>
<tr>
<td>2</td>
<td>425837</td>
</tr>
<tr>
<td>3</td>
<td>10283</td>
</tr>
<tr>
<td>4</td>
<td>13076</td>
</tr>
<tr>
<td>5</td>
<td>17663</td>
</tr>
<tr>
<td>6</td>
<td>30971</td>
</tr>
<tr>
<td>7</td>
<td>26027</td>
</tr>
<tr>
<td>8</td>
<td>28163</td>
</tr>
<tr>
<td>9</td>
<td>37002</td>
</tr>
<tr>
<td>10</td>
<td>52412</td>
</tr>
</tbody>
</table>

Week02> java Experiment_MergeSort_rand 2000 > out-m-rand.txt
Week02> java Experiment_MergeSort_up 2000 > out-m-up.txt
Week02> java Experiment_MergeSort_down 2000 > out-m-down.txt
Week02> java Experiment_InsertionSort_rand 2000 > out-i-rand.txt
Week02> java Experiment_InsertionSort_up 2000 > out-i-up.txt
Week02> java Experiment_InsertionSort_down 2000 > out-i-down.txt
Week02> R
[Previously saved workspace restored]
> EI=read.table("out-i-up.txt", header=TRUE)
> EM=read.table("out-m-up.txt", header=TRUE)
> z=EM$executionTime
> plot(EI,type="l")
> lines(z,col="red")
> EI=read.table("out-i-down.txt", header=TRUE)
> EM=read.table("out-m-down.txt", header=TRUE)
> z=EM$executionTime
> plot(EI,type="l")
> lines(z,col="red")
> EI=read.table("out-i-rand.txt", header=TRUE)
> EM=read.table("out-m-rand.txt", header=TRUE)
> z=EM$executionTime
> plot(EI,type="l")
> lines(z,col="red")
> q()
Save workspace image? [y/n/c]: n
Week02>
```