Assignment 5

**Valeria Fionda, Mouna Kacimi,
Werner Nutt, Simon Razniewski**

List Operations and Sorting with Lists

Instructions: Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. It is good academic standard to acknowledge collaborators, so if you worked together with other students, please list their names.
Be prepared to present your solution at the lab. If you are not able to explain your solution, this will be considered as if you had not done your work at all.
You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word. Experience shows that Word is in general difficult to use for this kind of task.
For a programming task, your solution must contain (i) an explanation of your solution to the problem, (ii) the Java code, in a form that we can run it, (iii) instructions how to run it. Also put the source code into your solution document. For all programming tasks, it is not allowed to use any external libraries ("import") or advanced built-in API functions (for example, `String.indexof("a")` or `String.substring(1,5)`), if not stated otherwise.
Please, include name, matriculation number and email address in your submission.

1. Iterative and Recursive List Operations

Implement a data type List that realizes linked lists consisting of nodes with integer values. The type List must have the following methods:

1. boolean isEmpty(),
2. void insertFirst(int i),
3. void insertLast(int i),
4. Node search(int i),
5. void deleteFirst(),

6. void delete(int i),

7. void print().

Also, develop both an iterative and a recursive version for the following methods: insertLast, search, delete, print.

(12 Points)

2. Sorting with Lists

To compare several algorithms for sorting integers, you are asked to

1. develop a list-based version of Insertion Sort and of Quicksort and document them using pseudo-code (for each algorithm, use the list version that is most appropriate);

2. implement the two algorithms in Java;

3. compare the two list-based sorting algorithms, that is, find out which algorithm is faster for which input size;

4. for Insertion Sort, compare in addition the version over lists wit the one over arrays.

Instructions: When working on the exercise, please take into account the following instructions.

- For each algorithm, choose the list version that is most appropriate (linked list, linked list with head and tail, doubly linked list, etc.) and explain your choice.

- For the running time experiments, generate random inputs of varying size. To obtain a realistic picture, generate several inputs for each input size and measure how long the algorithm runs on them. Gradually increase the number of integers contained in the lists (the array and the list, resp.) until the difference between the two algorithms is noticeable.

- For each comparison, make sure you measure the running time for the same input. When comparing the two list-based algorithms, first generate the input sequence, for instance in the form of an array, then create the input list for Insertion Sort and the one for Quicksort, and finally run each algorithm over its input. Proceed in an analogous way when comparing the array-based and the list-based version of Insertion Sort.

- State your observations and discuss possible reasons for them.

Hint: Consider increasing the memory size if needed, using virtual machine option `-XX:AggressiveHeap`.

(18 Points)

Submission: Until Sat, 4 May 2013, 11:59 pm, to

`dsa-submissions AT inf DOT unibz DOT it.`

Submit your work in two files, one PDF document and one .tar or .jar file with your code.