Seminar #4

Strategy: problems should be solved pencil & paper based. All questions, analysis and tracings are assumed to be finalized BEFORE running the programs.

Objectives:

- sorting algorithms with emphasize on and analysis (= what happens and WHY; which particular sequence of code explains the behavior):
  - Correctness (justify)
  - Strategy (forward vs backward; how do you know)
  - Efficiency (estimate)
  - Stability (justify; which predicate AND/OR statement in the code is responsible for it)
  - Tracing (what is displayed at each step of the execution)
- Implementation of loops:
  - For
  - While
  - Repeat-until

Take home knowledge:

The ability to perform efficiency and stability analysis in Prolog.
The ability to use any kind of loops in Prolog.

1. Generate the permutations of a list. Postponed from Seminar #3
   Discussion: Prolog is just executable specification.
   The number of permutations is n!
   Start from the recurrence relation of n!, n!=n*(n-1) which is almost Prolog code.

   ```
   n! =            % head of the clause, one perm of a set of size n
           n       %the way you can select the first item for the head of the output list
           (n-1)!  %recursive call on the input list without the selected item.
   ```

   Solutions:
   a. How many ways can you select all the elements in the list (one at a time). Think the nondeterministic way? Each way provides a different solution.
   b. How many ways can you find the list from which the selected element is missing?
   c. From the solutions above mentioned, is something that can be omitted? If so, is there requirement for the rest of the predicates? Which and why?

   Be aware: when the nondeterministic approach is employed for solving a problem, the predicate which should have the nondeterministic behavior needs to be cut-free (the implementation does NOT contain cut!).
2. Insertion sort – forward and backward. Start with an insert predicate.
3. Selection sort – forward and backward
4. Bubble sort. Start with a swap predicate. Next, make implementations with all kind of loops. Finally, nondeterministic approach?

Homework:

To be defined at the end of the class.