- This problem set has *three* questions. The programming problems may be harder, or require more time, than their point value suggests.
- Please type your solutions to the written component and send a pdf file to **REDACTED**. The pdf filename should be "EDS-A3-<your_user_name>.pdf"
- The deadline is **17.05.2014** anywhere on Earth.

(40) 1. **Permuting arrays:** Suppose we are given an array $A[1..n]$ and asked to permute it according to an arbitrary permutation π. If $A = [10, 12, 15, 20]$ and $\pi = (2, 3, 4, 1)$ then the permuted array should contain $[20, 10, 12, 15]$. As in the previous homework question, we use $\pi(i)$ to denote the position that the i-th element is swapped to according to π: e.g., $\pi(1) = 2$, $\pi(2) = 3$, etc. Suppose that we are only allowed to:

- swap elements in A;
- access an individual $\pi(i)$ in $\Theta(1)$ time for any $i \in [1, n]$: assume this is given as an oracle;
- make comparisons, perform assignment operations, etc., but no bit tricks;
- store $\Theta(1)$ extra data (numbers, array elements, etc).

Describe a $\Theta(n^2)$ time algorithm for permuting the array A according to π.

(40) 2. **LOBM:** In class we talked about Jacobson's Level Order Binary Marked (LOBM) encoding scheme for binary trees. However, we did not prove that the navigation operations actually work. Prove that the navigation operations left-child and right-child described on slide #45 in the word-RAM slides work correctly: i.e., for a 1 bit at position i, the left child is given by $2\texttt{Rank}(i)$ and the right child is given by $2\texttt{Rank}(i) + 1$. There are several ways to do this, but "most intuitive" proof will get some kind of bonus (to be determined).

(SPOJ:20) 3. `http://www.spoj.com/DS/problems/LEXCOMPR/`