1. [15 points]
   (a) Give an algorithm to remove duplicates (multiple occurrences of the same key) from an unsorted doubly linked list with a sentinel. If a key occurs several times in the list, the first occurrence should remain and the rest should be deleted.
   Give your solution using detailed pseudocode without assuming the knowledge of any algorithms presented in the course. Your algorithm does not need to be particularly fast.
   (b) What is the running time (in $O$-notation) for your algorithm in part (a)? How could you solve the problem faster using other data structures presented in the course?
   It is sufficient to give a short explanation of the principles and running time of your algorithm; no detailed pseudocode is necessary.

2. [10 points] The problem deals with binary search trees that are not assumed to be balanced.
   (a) We wish to include in each internal node $x$ a value $\text{size}[x]$ that gives the number of descendants of $x$. Give an algorithm for filling in the $\text{size}$ values in a given binary tree. (Clarification: for this part of the problem it is not important whether the binary tree is actually a search tree.)
   (b) Assume now that a binary search tree contains the proper $\text{size}$-values as defined in part (a). Give an efficient algorithm that given a natural number $k$ returns the $k$th largest key in the tree. The running time of your algorithm should be linear in the height of the tree.

In both parts, give your algorithms as detailed pseudocode.

3. [15 points] The input consists of an unsorted array $A[1..n]$ that contains $n$ different natural numbers, and a parameter $k$ where $1 \leq k \leq n$. The task is to print out the $k$ smallest numbers in the array $A$. Give for the task three different algorithms with running times
   (a) $O(kn)$
   (b) $O(n \log n)$ and
   (c) $O(n + k \log n)$.

Justify the running times and compare them for different values of $k$. You may use as subroutines any algorithms presented in the course and use their known running times in your analysis. There is no need to give detailed pseudocode, a high level description of the algorithms is sufficient as long as the justification for the running times can be seen.

4. [20 points]
   (a) An undirected graph $G = (V, E)$ is given using adjacency lists. The task is to find out whether the graph is connected or not. Give (in detailed pseudocode) an algorithm for the task and analyse its running time. For full credit, the running time should not exceed $O(|V| + |E|)$.
   (b) Suppose now the graph $G$ is directed. How would you decide in time $O(|V| + |E|)$ whether the graph is strongly connected or not? Explain the basic idea of your algorithm (no detailed pseudocode is needed) and justify its running time.

The maximum score from the examination is 60 points.