Data Structures - Assignment no. 3

Date due: 13/8/02

Remarks:

- For every algorithmic problem, try to find the most efficient solution. Analyze the complexity of every algorithm you give, and in case it is not trivial, prove its correctness.
- Unless you are explicitly asked otherwise, please describe algorithms only to the extent which makes them clear to a reasonable reader. You may use plain language, or pseudo-code, whichever is more convenient.

1. Given a binary tree T with n leaves, give a direct proof showing that the number of internal nodes in T is no less than $n-1$. Show that equality holds if and only if T is a full tree (Reminder: a full binary tree is a binary tree where every internal node has two direct offsprings).
2. A binary tree T is an AVL tree if it has the following properties: Every internal node in T has a brother, and the difference between the heights of any two brothers in T is at most 1.
   (a) Give an algorithm that, given a pointer to the root of binary tree T, determines whether T is an AVL tree.
   (b) Prove that an AVL tree with n nodes and height h must satisfy $h \prec \log n$.
   (c) Define $F(i)$ recursively by $F(0) \stackrel{def}{=} 1$, $F(1) \stackrel{def}{=} 1$, and $F(i) \stackrel{def}{=} F(i-1) + F(i-2)$ for $i > 1$. Prove that every AVL tree with n leaves and height h satisfies $n \geq F(h)$, and that this bound is tight.
3. Show linear-time algorithms that use $O(1)$ memory, for the following tasks. Given a pointer to the root of a binary tree T,
   (a) Perform an in-order walk on T.
   (b) Find the height of T.
4. Given the inorder and postorder walks of a binary tree T, in which all nodes have distinct keys, show an algorithm which reconstructs T (i.e. generates a tree whose inorder and postorder walks are the same as of T) in time $O(n \log n)$. Hint: use sorting.
5. Give a detailed proof for the fact that the Breadth-First-Search visits a node x in a tree T after it visits a node y, if and only if either of the following is true:
   - The depth of x in T is higher than that of y
   - x and y have the same depth in T, but x is to the right of y.