Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Lehrstuhl für numerische Programmierung

WS 2002/2003
Worksheet 10

14.1.2003

Fundamental Algorithms

Exercise 1

Write an algorithm that copies all keys that are stored in a binary tree into an array of appropriate size. In the resulting array, the keys shall be sorted in descending order.

Exercise 2

Prove or disprove the following statement:
If, in a binary search tree, the keys k_1 and k_2 are deleted, the resulting search tree does not depend on whether k_1 or k_2 is deleted first (i.e. deleting k_1 before k_2 will lead to the same search tree as deleting k_2 before k_1).

Exercise 3

Show that an AVL tree containing m inner nodes* that have exactly two non-empty sons, has at most $\frac{\sqrt{5}-1}{2}m$ nodes with a height balance different from 0.

* *inner nodes* are nodes that are not leaves.

Exercise IV (CSE)

Consider the binary tree given by the expression

```
x = (5, (3, emptyTree, (4, emptyTree, emptyTree)),
     (8, (6, emptyTree, emptyTree), (10, (9, emptyTree, emptyTree),
     (13, emptyTree, emptyTree))))
```

- draw a diagram of this binary tree and decide whether its a binary search tree
- perform the following operations (using the resp. algorithms from the lectures), and draw a diagram of the search tree after each operation:
  - `TREE_INSERT(x,11)`
  - `TREE_DELETE(x,5)`
  - `TREE_INSERT(x,5)`
  - `TREE_INSERT(x,12)`

Exercise V (CSE)

Decide whether the binary tree given in exercise IV is an AVL tree

- before the insert/delete operations, and
- after each of the regular insert/delete operations.

Again, perform the insert/delete operations given in exercise IV, and name and perform the rotation(s) to restore the AVL property after each step (if required). Draw a diagram of the search tree after each of your insert/delete, or rotation operations.