

58131 Data Structures (Spring 2008)

1st course examination, 25 February

To facilitate grading, please answer to each of problems 1–3 on a separate sheet of paper. In Problem 1 you may answer to parts (a)–(c) on a single sheet, and similarly in Problem 2. Write your name and the name of the course on each sheet. Return at least one sheet with your name for each problem, even if you do not have solutions to all of the problems.

In problems where you are asked to give pseudocode, you may instead use e.g. Java if you prefer.

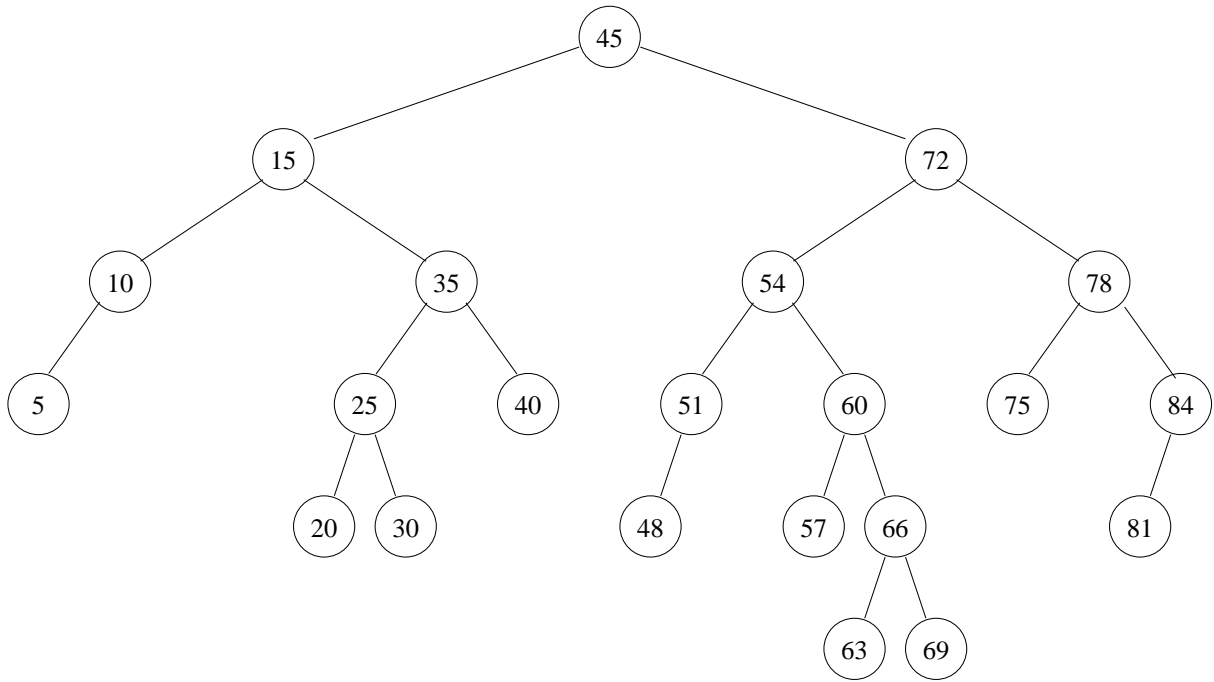
1. [8 points] The task is to implement a dictionary, where the keys are names of persons, each consisting of a (single) first name and a last name. In addition to the usual search, insert and delete operations, the following operations should be supported:
 - PRINT-LAST-NAMES prints out each last name in the structure exactly once. The names may be printed out in any order. The operation should run in time $O(n)$, where n is the number of different last names. Thus, the operation should not be slowed down in case there are a lot of different first names associated with some of the last names.
 - PRINT-FIRST-NAMES(*lastname*) prints out all the first names associated with the last name *lastname*. The first names may be printed in any order. The operation should run in time $O(n+m)$, where n is the number of different last names and m the number of first names to be printed. Thus, the operation should not be slowed down in case some other last names have lots of first names associated with them.
 - (a) Explain the basic idea for a suitable data structure for this. A suitable level of detail would be a picture with some clarifying text.
 - (b) Give detailed pseudocode for the operations PRINT-LAST-NAMES and PRINT-FIRST-NAMES(*lastname*).
 - (c) Give detailed pseudocode for the operation INSERT(*firstname*, *lastname*).

You may assume that the same pair (*firstname*, *lastname*) will never be inserted more than once. For full credit your solution must achieve the stated running times, but less efficient solutions can give partial credit.

2. [8 points]
 - (a) What is the balancing condition for an AVL tree? What is the advantage of using an AVL tree over a non-balanced binary search tree?
 - (b) Show the intermediate results when the keys 3, 8 and 27 (in this order) are inserted into the AVL tree given on the reverse side.
 - (c) Show the intermediate results when the keys 60 and 75 (in this order) are deleted from the AVL tree given on the reverse side.

“Intermediate results” means the tree after each insertion or deletion has been completed, and any situations where balancing operations are performed. You need not reproduce parts of the tree that remain unchanged, as long as you clearly show what the changes are.

3. [8 points] Answer **either** to (a) **or** to (b).
 - (a) Show that in a non-empty full binary tree,
$$\text{number of leaves} = \text{number of internal nodes} + 1.$$
 - (b) Give detailed pseudocode for an algorithm DELETE-SMALLER(T, k), which removes from a binary search tree T all keys less than k . For full credit the algorithm should run in time $O(h)$, where h is the height of the tree. You do not need to care about what happens to nodes removed from the tree (i.e. garbage collection).



The AVL tree for Problems 2(b) and 2(c).