Data Structures and Algorithms

Assignment #5

Due: Paper Assignment Wednesday May 10th in class

This assignment is "paper only". The programming part will be given after you turn in Assignment #4. Problem 5 won't be graded (so you don't have to turn it in) but you are strongly encouraged to do it since it will be good practice for the programming assignment. Note that the programming assignment will NOT ask for exactly the same methods that Problem 5 does.

1. Given a binary search tree, write a function (in pseudo-code) that returns true if the BST is an AVL tree and false otherwise (Hint: there is a particular tree traversal order that is most convenient).

2. Problems R-9.5, R-9.6 and R-9.7

3. Problem R-9.14

4. Problem C-9.7

5. Design algorithms to handle insertion, deletion, and find for open addressing hashing using linear probing.

Assume that we want to insert non-negative integers in a table of size M using the hashing function *hash()* (hash(i) returns an integer between 0 and M-1; you don't have to write the hash function for this paper assignment but you will need to write one of your own for the forthcoming programming assignment).

Initially the table is empty and all entries are set to -2. When an entry has been inserted and then deleted, the entry becomes -1.

Give pseudocode algorithms for the following 3 functions

- *Find(x,T)* which returns the index of where x is stored if it is in the table T and -1 otherwise.
- *Insert(x,T)* which inserts x into the table T. You can assume that x is not currently in the table. Naturally, a location with a -1 or a -2 in it can be used to place a newly inserted item. Your algorithm should indicate when the hash table T is full and x cannot be inserted (for example, a successful insertion can return a value of 0 and an unsuccessful one a value of 1).
- *Delete(x,T)* which removes x from the table T replacing x by the value -1 in the location where x was. You can assume that x is in the table.