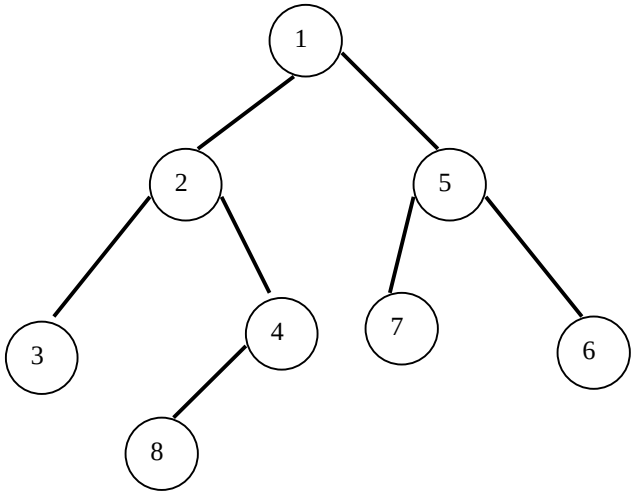
FSU COP 4530 / CGS 5425 (Spring 2008)
Data Structures, Algorithms, and Generic Programming

Due: April 15

Assignment 6: Max points: 100

Please show steps or give justifications for all your answers, unless we specify otherwise.
Each question is worth 10 points.

1. Show the order in which nodes are visited in the following traversals of the binary tree shown below: (a) *post-order*, and (b) *pre-order*.



2. Draw the *AVL* tree that results after the following sequence of operations on a tree that is initially empty: *insert(9), insert(4), insert(2), insert(7), insert(8), insert(3), insert(0), insert(2.5), Delete(2), Delete(0), Delete(3), Delete(2.5)*. Show the intermediate trees after each operation, in order to get partial credit for an incorrect answer.

3. Show the heap that results from using the O(n) time max-heap initialization algorithm on the following array: *5 6 9 2 8 1*. Show both, pointer and array representations.

4. (a) Show the heap that results from applying the following operations to the heap of #3: *push(5), push(9), push(1)*. (b) Show the heap that results from applying the following operations to the result of #4a: *pop(), pop()*. (*pop = DeleteMax*.)

5. Draw an *open addressed hash table* of size *7* after the sequence of operations given below. Also give the *number* of table entries checked for each of the operations. The following hash function is used:

```
int hash(string s)
{
  int sum=0, I;
  for(I=0; I<s.size(); I++)
     sum += s[I]-'a';
  return sum;
}
```

The probe function is: *p(i) = i*i*.

Operations*: insert(ee), insert(bb), insert(f), insert(b), search(d), Delete(b), search(d)*

6. Give good *upper and lower* bounds on the height of a tree with n nodes (that is, the largest and smallest possible heights), where each node of the tree may have at most d children.

7. Let a function f be defined such that $f(0) = 2$; $f(n) = 2*f(n/k)$, for some constant $k>1$, when $n>0$. (n/k is performed in integer arithmetic; that is, the floor is taken.) Derive the time complexity of computing this function recursively.

8. Prove formally that if a node x in a BST (with distinct elements) has a right child, then the successor of x must be the smallest valued node in the right subtree of x.

9. We know that an *inorder* traversal of a binary search tree visits its nodes in sorted order. However, an inorder traversal of a *min-heap* does not necessarily visit nodes in sorted order. In fact, neither do the other three traversals that we discussed in class. We might wonder if there is some other traversal taking $O(N)$ time that visits the nodes of a min-heap in sorted order. Either describe a linear time algorithm for visiting the nodes of a min-heap in sorted order, or prove that such a traversal does not exist. You may use the fact that comparison based sorting requires $\Omega(N \log N)$ time.

10. Assume that in a particular application, if an element is searched for at some point in time, then it is very **unlikely** that it will be searched for the next few times. Suggest a *self-adjusting strategy for a BST, based on rotations,* for this application.