FSU COP 4530 / CGS 5425 (Spring 2009)
Data Structures, Algorithms, and Generic Programming

Due: April 16

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. (15 points) In the following questions, show the max-heap using both, pointer and array representations, after each of the following sequence of operations is completed.

   A. Initialize a heap with the following initial data: 5 6 4 2 8 1 using the O(n) time max-heap initialization algorithm.
   B. To the heap above: push(5), push(9), push(1).
   C. To the heap above: pop(), pop().

2. (15 points) Show the state of the binary search tree after the operations in (a) are complete, and also after those in (b) are complete.

   A. insert(9), insert(4), insert(2), insert(7), insert(8), insert(5), insert(6)
   B. delete(4), delete(2)
   C. Show the order in which nodes are visited in a pre-order traversal of the tree in (A).

Show the intermediate trees after each operation, in order to get partial credit for an incorrect answer.

3. (10 points) Show the state of an AVL tree after the operations in (a) are complete, and also after those in (b) are complete.

   A. insert(9), insert(4), insert(2), insert(7), insert(8), insert(5), insert(6)
   B. *delete(4), delete(2)*

Show the intermediate trees after each operation, in order to get partial credit for an incorrect answer.

4. (10 points) 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)*

5. (10 points) 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.

6. (10 points) 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.

7. (10 points) If we start at the root of a *BST* and keep taking the right child of each node (if the right child is not null) until we can go no further, then the last node reached has the largest value among all nodes in the BST. Prove this formally.

8. (10 points) 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.

9. (10 points) Consider a simple spam (junk email) filter as described below. We keep track of senders (say, the *from* field in the email) whose messages should be tagged as spam. Initially, no one is listed as a spammer. Each time the user marks an email as spam, we record that sender as a spammer. Each time we receive an email, if the sender has been recorded earlier as a spammer, then the message is tagged as spam. Suggest a suitable data structure, from among those that we have discussed in class, to store the records of spammers. State any reasonable assumptions that you make, and justify your answer.