COP 4530 / CGS 5425 (Fall 2005)
Data Structures, Algorithms, and Generic Programming

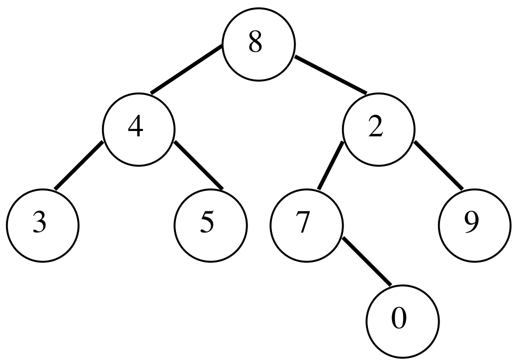
Final exam: Max points: 100 (+10 bonus points), Time: 2 hours

---

First Name: _______________ Last Name: _______________

This is a closed book examination.

1. (a) (5 points) Show the order in which nodes are visited in an *inorder* traversals of the binary tree shown below.



(b) (5 points) Assume that a `Node` class is defined as shown below. Also assume that a function *void visit(Node *)* already exists, which performs some operation on a node (such as printing its value). Using an *STL* stack, write a function called *levelorder*, which performs a level order traversal of a binary tree. You need not show code to include the STL stack header file.

```
class Node{
public:
  int key;
  Node *P, *LC, *RC; // P: parent, LC: left child,
                     // RC: right child
};
```

```
void levelorder(Node *root)
{
```

2. (30 points) In each question below, draw a figure to show the state of the data structure after the sequence of operations given below is complete.

a. Draw the *BST* tree that results after the following sequence of operations on a BST tree that is initially empty: *insert(9), insert(4), insert(1), insert(7), insert(0), insert(8), insert(3), insert(6), insert(2), insert(2.5), Delete(4), Delete(3), Delete(0).*

b. Draw the *AVL* tree that results after the following sequence of operations on an AVL tree that is initially empty: *insert(9), insert(4), insert(1), insert(7), insert(0), insert(8), insert(3), insert(2), insert(1.5), insert(10), insert(-1), insert((3.5), Delete(7).*

c. Show the *max-heap* that results from applying the *O(n)* heap initialization algorithm that we discussed in class to the following array: 3, 6, 10, 5, 1, 9, 7, 4, 0, 2, 8. Draw the *pointer* representation.

3. (a) (5 points) Give good asymptotic time complexities for each of the following operations on the data structures given below.

Av.: Average, Amort.: amortized, WC: Worst case, AvAm.: Average amortized time

	push	*pop*	*top/ front*	*erase*	*push front*	*push back*	*pop front*	*pop back*	*search*
vector	x	x	x			Amort:			
sorted vector		x	x		x	x	x	x	
deque	x	x	x	x	Amort:	Amort:			x
stack				x	x	x	x	x	x
queue				x	x	x	x	x	x
BST	WC: Av:	x	x	WC: Av:	x	x	x	x	WC: Av:
AVL tree		x	x		x	x	x	x	
heap				x	x	x	x	x	x
hash table	AvAm:	x	x	Av:	x	x	x	x	Av:
doubly linked list	x	x	x						

(b) (5 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 to store the records of spammers. State any reasonable assumptions that you make, and justify your answer.

4. (a) (15 points) Write code to implement a *Delete* member function of a Binary Search Tree class named *BST*. A BST object contains a variable *Node *Root*, which points to the root of the tree (it is *NULL* if the tree is empty), where a *Node* is as defined in question *1b*. You may assume that functions *Node *GetPredecessor(Node *)* and *Node *GetSuccessor(Node *)* are available for your use.

```
void BST::Delete(Node *n)
{ // Delete the node n. Assume n is a valid node.
```

(b) (15 points) Write a member function of the above class, called *LeftRotate*, which performs a left rotation on a node, which you can assume is a valid node that is not the root.

```
void BST::LeftRotate(Node *n)
{ // Rotate n up. Assume n is the right child of its parent.

}
```

5. (a) (10 points) Consider a strange type of tree, where the root can have at most *2* children, nodes in the root's left subtree can have at most l children each, and nodes in the root's right subtree can have at most r children each. Derive a formula for the maximum number of nodes in such a tree of height h, where *h, l, and r are greater than 1*.

b. (10 points) *Disprove* the following statement with a counterexample: *In an AVL tree, if the balance factor for a node is 0, then the balance factors for all its descendents too are 0.*

Bonus points:

6. (10 points) Prove that if a node in a BST has a successor, but has no right child, then its successor must be an ancestor. (We will consider only BSTs with distinct elements.)