CS 301: Midterm #2

Problem 0 (5 points): Write your name at the top of this page.

Problem 1 (30 points):
   (a) Show the binary search tree that results from inserting elements in the following order: 20 50 70 10 40 90 35 45 5 30 42.

   (b) Repeat (a) but for an AVL tree.

   (c) Show the binary search tree that results from deleting 50 and then 45 from the tree obtained at the end of (a).
Problem 2 (25 points): Consider the following picture of a doubly linked list. Assume that the Node structure contains fields named info, next, and prev and that listData and tail have been declared and initialized as implied by the picture.

(a) Write the statement(s) needed to declare a pointer called dead and make it point to the node containing 30.

(b) Write the code needed to delete the node containing 30 from the list. Access the nodes through the dead pointer initialized in (a), not through listData and tail. Be sure to release the memory occupied by that node.
Problem 3 (20 points): Write the addItem method for a Queue class implemented with a dynamically allocated array. If the array is full, addItem should resize it. addItem should run in O(1) amortized time. Do not use any other methods in the Queue class unless you write them as well. Assume Queue has the following data members:

- ItemType* items, the array holding the items on the queue;
- int maxItems, which holds size of that array;
- int numItems, which is the number of used elements in the array;
- int first, which is the index of the element at the front of the queue; and
- int last, which is the index of element at the end of the queue (these indices wrap around so that last may be less than first.)
Problem 4 (20 points): Write a function `isAVLTree` that takes a `Node` pointer as a parameter and returns the height of the subtree rooted at that `Node` through a reference parameter and as the return value `true` or `false` depending on whether or not the subtree is an AVL tree. This function should

- check for base cases,
- determine if the left subtree is an AVL tree and determine its height,
- determine if the right subtree is an AVL tree and determine its height,
- determine the height of the subtree rooted at the `Node` passed in as a parameter, and
- return `true` or `false` based on whether or not that subtree is an AVL tree.