QUEENS COLLEGE Department of Computer Science
CSCI 313 Practice problems on Search Trees

Instructor: Alex Ryba

These problems were given on exams for this course. Some older problems did not make use of generics in Java, but generic implementations are now required in this course.

Problem 1 For each of the following structures list the additional requirements that a binary tree must satisfy to qualify.

(a) A binary search tree.

(b) A binary heap.

(c) An AVL Tree.

(d). Implement the following method:

```
BNode<T> leftmostRightDescendant(BNode<T> n)
```

The method should return the leftmost right descendant of a binary node (or return *null* if there is no right descendant.

Problem 2 Consider the following diagram showing the state of a binary tree.

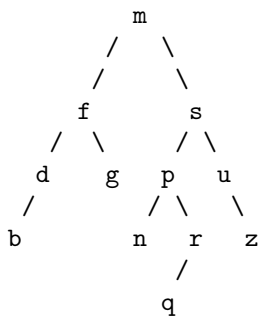

Diagram T.

(a) Write down the inorder traversal of the tree.

(b) Write down the postorder traversal of the tree.

(c) List all single lower case letters whose insertion into an AVL Tree represented by Diagram T would require a rebalance of the tree. (Remember: AVL trees can not contain duplicate data items.)

(d) Assume that Diagram T represents the state of an AVL Tree. Show how the tree is changed when the data item *g* is removed.

Problem 3 The generic class `BinaryNode<K implements Comparable<K>>` is implemented with standard instance variables called *parent, left, right* that have type `BinaryNode<K>` and an instance variable called *size* that gives the number of items in the subtree rooted at the node. Complete the implementation for a method *decOrder* that returns the data items in the subtree (rooted at the node) in descending order given that the tree satisfies BinarySearchTree order.

```
public K[] decOrder() {  // continue from here
```