# Homework 9 Part B

Due date: Apr 9, 2020, 9:30am

Homework 9 Part A has to be completed on the course website. Homework 9 Part A and B combined account for 50 points, like any other prior homework.

1. (10 points)

Perform the following operations on an initially empty binary search tree. Draw the tree after *each* operation. The nodes should be sorted by the lexicographical order (i.e. like names in a phone book).

1. insert(“goose”)
2. insert(“horse”)
3. insert(“rooster”)
4. insert(“cat”)
5. insert(“dog”)
6. insert(“cow”)
7. insert(“hen”)
8. insert(“pig”)
9. delete(“cat”)
10. delete(“horse”)

After all inserts

After the deletes

1. (10 points)

Write Java-like code for the constructor and the method addAsLeftChildOf. The constructor and method addAsLeftChildOf have to meet the specification in the comments.

/\*\*

\* BinaryTree

\*

\* **@author** CS3151

\* **@param** <T> type of the node values

\*/

**public** **class** BinaryTree<T> {

**private** BinaryNode root;

/\*\*

\* Instantiates a new binary tree with three nodes: The root of the new tree has

\* two children where the root has the specified value valueRoot, the left child

\* of root has the specified value valueLeft, and the right child of root has

\* the specified value valueRight.

\*

\* **@precondition** valueRoot != null && valueLeft != null && valueRight != null

\* **@param** valueRoot value of the root

\* **@param** valueLeft value of the root's left child

\* **@param** valueRight value of the root's right child

\*/

**public** BinaryTree(T valueRoot, T valueLeft, T valueRight) {

if (valueRoot != null && valueLeft != null && valueRight != null) {

BinaryNode root = new BinaryNode(valueRoot);

BinaryNode left = new BinaryNode(valueLeft);

BinaryNode right = new BinaryNode(valueRight);

left.parent = root;

right.parent = root;

root.left = left;

root.right = right;

this.root = root;

}

}

/\*\*

\* Adds a new node with the specified value as a left child of the specified

\* node. If parentNode has already a left child, then the left child of

\* parentNode becomes the left child of the new node.

\*

\* **@precondition** node != null && value != null

\* **@param** value the value of the new node to be added

\* **@param** parentNode the parent of the new node

\*/

**public** **void** addAsLeftChildOf(T value, BinaryNode parentNode) {

if (node != null && value != null) {

BinaryNode left = new BinaryNode(value);

if (parentNode.left.value != null) {

left.left = parentNode.left;

}

left.parent = parentNode;

parentNode.left = left;

}

}

…

/\*\*

\* Class BinaryNode

\*

\* **@author** CS3151

\*/

**protected** **final** **class** BinaryNode {

**private** T value;

**private** BinaryNode parent;

**private** BinaryNode left;

**private** BinaryNode right;

**private** BinaryNode(T value) {

**this**.value = value;

**this**.parent = **null**;

**this**.left = **null**;

**this**.right = **null**;

}

}

}

Submission

Submit a single PDF file or a single MS Word document with your solutions. No other file formats are accepted. If you prefer to write (or draw) your solution by hand and you do not have a scanner, take pictures of your hand-written solutions and imbed the pictures in a Word document.