Joshua Nelson
CSCE 146

Lab Report 08

Problem

Create a binary search tree where all nodes have lower values to left and greater values to the right. The tree should have the methods add, delete, and depth. The tree should support pre-, post-, and in-order traversal. The tree does not need to be balanced. The tree does not have to handle multiple instances of the same value.

Proposed Solution

Insert: Create a recursive algorithm which starts at the root. For any node, if the value to insert is less, insert to the left; otherwise insert to the right. Call the method starting at current node. If node is null, create a new node with value and end procedure.

Delete: Create a recursive algorithm which starts at the root. Find the value using binary search. If the node holding the value is a leaf, set it to null. If it has a single child, replace it with the child. Otherwise switch the current value with the smallest value that is greater than it. Delete the new node using recursive algorithm.

Tests and Results

```
Inserting 4

Inserting 3

Inserting 15

Inserting 13

Inserting 31

3 4 13 15 31


Printing preorder: 4 3 15 13 31


Depth of 13 is 2
```

Joshua Nelson
CSCE 146

```
Depth of 4 is 0

Depth of 32 is -1


Removing 4

3 13 15 31

Inserting 8

Inserting 13

Inserting 3

Inserting 4

Inserting 18

Inserting 19

Inserting 10

Inserting 1

Inserting 9

Inserting 2

Printing pre-order

8 3 1 2 4 13 10 9 18 19

Removing 4

8 3 1 2 13 10 9 18 19
```

Problems Encountered

Java does not support modification of values passed to a function; i.e. ‘delete(Node n) { n = null; }` has no effect. Dr. Shepherd suggested a recursive algorithm which may pass back null. While this would have work I found it difficult to understand. I instead create a ‘parent’ node data as part of the Node class; the modified method would be `delete (Node n) { if (n.parent.left == n) n.parent.left = null; else n.parent.right = null; }`. Since the ‘==’ operator in Java compares memory references for objects, this would have the intended effect even if the nodes held the same value.

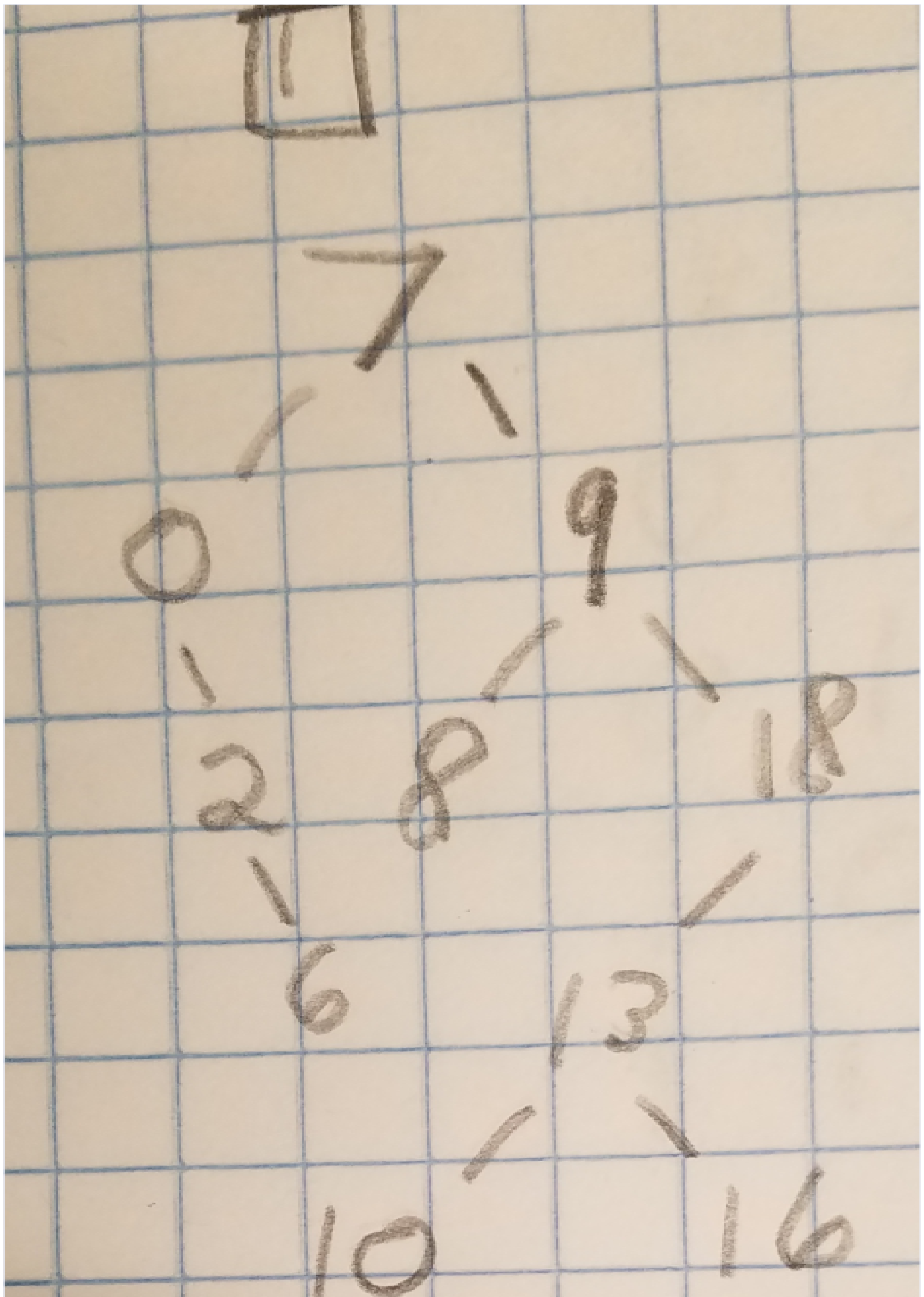
Conclusions and Discussion

I greatly over-engineered this class because I will extending it for the extra-credit balanced trees (AVL and Red-Black). There are very many methods and helper functions that were not necessary for the assignment. I also made use of the functional API available in Java 8; this is not supported on all versions of Eclipse but is supported by Dr. Java and javac. I enjoy functional programming and look forward to taking CSCE 330.

Joshua Nelson
CSCE 146

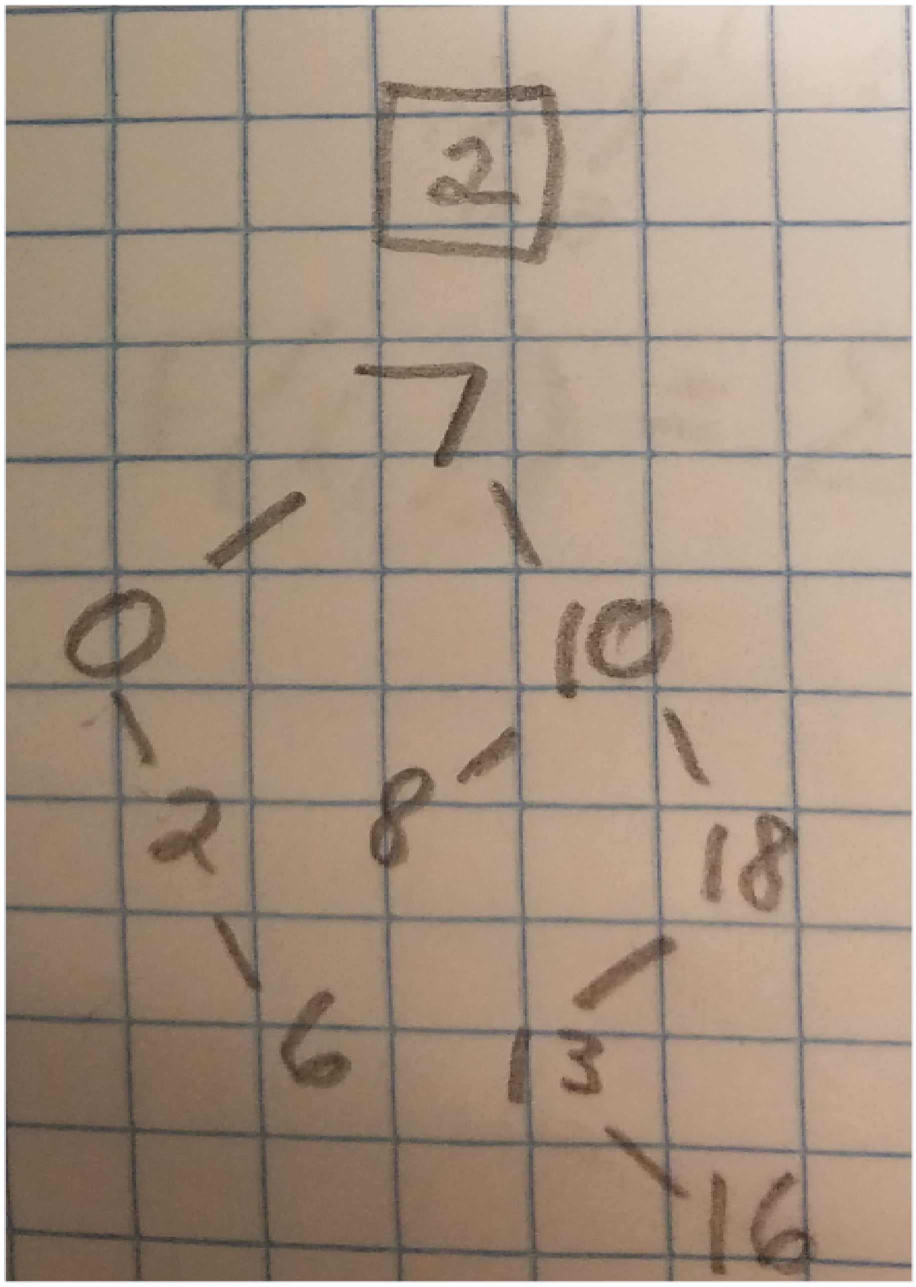
Additional Questions

1. Draw a Binary Search tree with the following values inserted in this order 7, 9, 18, 0, 8, 13, 2, 10, 16, 6



Joshua Nelson
CSCE 146

2. Remove 9 from the previous tree and draw that



3. If a Binary Search tree is balanced what is the Big O complexity to search for an item in the tree?

log_2(n), same as binary search.