Extra Credit Assignment 1

**Please read turn-in checklist at the end of this document before you start doing exercises.**

**Section 1: Pen-and-paper Exercises**

1. Given a BST, return its height.

The height of a tree is the length of the longest path from the root node down to a leaf node.

For example, in the following BST, the longest path from the root node down to a leaf node is 20 – 8 – 12 – 10 or 14, therefore the tree height is 3.



Assume the tree is balanced, and the tree height is O(log n). Design an O(logn) time algorithm to solve this problem.

(i) describe the idea behind your algorithm in English (2 points);

Since the tree is balanced we can simply go down one side of the tree. I chose left. If p.left is null then the answer is zero otherwise return p.left + 1.

(ii) provide pseudocode (5 points);

height(Node p)

if(p.left == null)

return 0;

end if

else

return height(p.left) + 1;

end else

(iii) analyze its running time (3 points).

The running time is O(logn) since it is a balanced tree which means it only needs to follow one branch.

**Note: Full credit (10 points) will be awarded for an algorithm that is O(logn). Algorithms that are slower than O(logn) will be scored out of 3 points.**

**Section 2: Java Implementation**

1. Implement the problem 1 in Java.

Note:

Find a file called BST.java in assignment 7 folder.

Complete the method of Height().

Test your method in the main method provided following the comments.

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt/.pdf), and to Section 2 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**
2. **Create a folder and name it 'FirstName\_LastName\_Extra\_1'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**

}

}