Lab 5

Trees

Question 1

Part A

- Given image of eight trees and some tree names
- Match which image belongs to which tree

Part B

- Make an AVL tree with some given keys

Question 1

- No coding.
- Submit a PDF file with your answers.
- Name your file **Lab5Question1.pdf**.

Check:

- Lecture 9 Tree Part 2 - 4

Question 2

- Lab5Q2Test.java >> tests Binary Search Tree (BST) methods
- Check the TreeNode.java for the instance variables and (get/set) methods
- You need to complete several methods inside the BST.java file.
- Check Slide: Lecture 7 Part 2
- Check Slide: Lecture 8 Part 3

Question 2

insert(): inserts a key/item to a BST

- similar to the add method (recursive) discussed in the class
- Instead of using recursive code, use *iterative* (e.g., loop)

recPrintTreeDesc(): prints nodes in descending order using recursion

- Already a printTree() method in this class which prints all the nodes in non-descending order.
- [Hints: for pre-order traversal (in ascending order), we traverse the left subtree, then visit the root, and then traverse the right subtree. To visit the items in descending order, we switch the left and right traversals in the recursive algorithm]

Question 2

recursivePrintLeafs(): recursive method to show the leaf nodes of the BST

- leaf nodes do not have both left and right child. For such nodes
- [Hints: check if the node is null, recursively call the left and right sub-trees and check if the left and right sub-trees are null]

recursiveInternalNodes(): recursive method to show the internal nodes (i.e., non-leaf nodes) of the BST

recursiveCountNodes(): return the number of nodes in the subtree rooted at a given key value

- first search the key value to find the root, then count the root itself and all its descendants.
- If the root is not found in the tree, the program should display 0 node.

Question 2

isPerfectTree() : return true if the tree is **Perfect**, return false otherwise. (Check lecture 8, part 3, slide 39)

isFullTree() : return true if the tree is **Full**, return false otherwise. (Check lecture 8 tree **-** part 3 slide 37)

isCompleteTree(): return true if the tree is **Complete**, return false otherwise. (Check lecture 8 tree **-** part 3 **-** page 38)

- [Hints: if the depth of a tree is d and the total number of nodes is n, then n should always be n >= 2d + 1 for the left subtrees, and n>= 2d + 2 for the right subtrees in each level of the tree].

Question 2

getHeight(): return the height of the tree.

isDegenerateBinaryTree() : return true if the tree is a **Degenerate Binary Tree**, return false otherwise.

- Hint: Check the lecture and observe the relation between the number of nodes and the height of the tree.

- **2 marks (BONUS)] Method findSmallest():** find the minimum value stored in the subtree rooted at a given key value (i.e., subtree root value). Assume that the root is always present in the tree.