MINIA UNIVERSITY
FACULTY SCIENCE
Department of Computer Science
Data Structures Using Python

Exercises #8
Tree ADT

8.1 Add a method get_tree_height() to LinkedBinaryTree that calculates the height of a binary tree, where:

height of a node = max(height of left subtree, height of right subtree) + 1

8.2 Add a method delete_subtree(p) to LinkedBinaryTree that removes the entire subtree rooted at position *p*, making sure to maintain the count on the size of the tree.

8.3 Add a method swap(p,q) to LinkedBinaryTree that has the effect of restructuring the tree so that the node referenced by p takes the place of the node referenced by q, and vice versa. Make sure to properly handle the case when the nodes are adjacent.

8.4 Give an efficient algorithm that computes and prints, for every position *p* of a tree *T*, the element of *p* followed by the height of *p*'s subtree.

8.5 Write a Python method, insertBST(k, T), to insert a new key k, possibly a duplicate key in a BST T, possibly empty (i.e., None), and return the tree where the new key was inserted.

8.6 Write a Python method to create a Binary Search Tree (BST) using a given unsorted array elements.

8.7 Write a Python method to create a balanced Binary Search Tree (BST) using a given array elements, where array elements are sorted in ascending order. (Hint: Make the root of the BST to be the mid value of the array.)
(A *node* in a tree is *height-balanced* if the heights of its subtrees differ by no more than 1.)

8.8 Write Python methods that perform the following operations for a binary tree *T*:

- preorder_next(p): Return the position visited after *p* in a preorder traversal of *T* (or None if *p* is the last node visited).
- inorder_next(p): Return the position visited after *p* in an inorder traversal of *T* (or None if *p* is the last node visited).
- postorder_next(p): Return the position visited after *p* in a postorder traversal of *T* (or None if *p* is the last node visited).

8.9 Implement the binary tree ADT using the array-based representation described in Lecture 10.

8.10 Implement the tree ADT using a linked structure as described in Lecture 10. Provide a reasonable set of update methods for your tree.