CSC212

Tutorial 8

Binary Trees

Problem 1:

Write the method ***leafNodes*** part of the Binary Tree ADT. It should return the number of leaf nodes in the tree.

Example:

Method: *public int leafNodes()*

Solution:

```
public int leafNodes(){
        return leafNodes_rec(root);
}

private int leafNodes_rec(BTNode<T> p) {
        if(p == null)
                return 0;
        if(p.left == null && p.right == null)
                return 1;
        return leafNodes_rec(p.left) + leafNodes_rec(p.right);
}
```

Problem 2:

Write the method ***treeHeight*** part of the Binary Tree ADT. It should return the height of the tree. The height of the tree is the longest path from the root to a leaf node.

Example:

Method: *public int treeHeight()*

Solution:

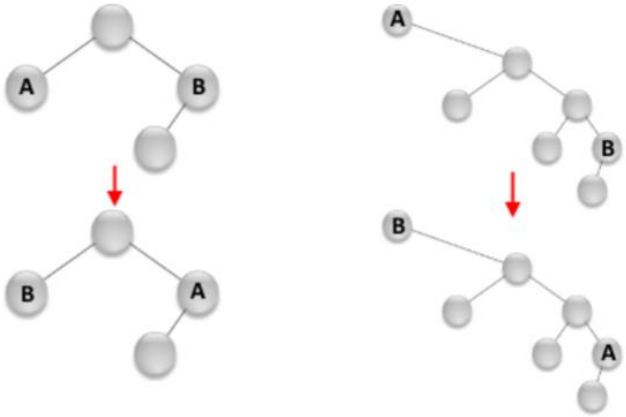
```
public int treeHeight(){
        return treeHeight_rec(root);
}

private int treeHeight_rec(BTNode<T> p) {
        if(p == null)
                return 0;
        int leftHight = treeHeight_rec(p.left);
        int rightHight = treeHeight_rec(p.right);
        return leftHight > rightHight ? leftHight + 1 : rightHight + 1;
}
```

Problem 3:

Write the static method ***swapMost*** that takes a Binary tree ***bt*** and swaps the data of the left most node with the right most node

Example:



Method: *public static <T> void swapMost(BinaryTree<T> bt)*

Solution:

```
public static <T> void swapMost(BT<T> bt){
        if(!bt.empty()) {
                bt.find(Relative.Root);
                while(bt.find(Relative.LeftChild));
                T mostLeft = bt.retrieve();
                bt.find(Relative.Root);
                while(bt.find(Relative.RightChild));
                T mostRight = bt.retrieve();
                bt.update(mostLeft);
                bt.find(Relative.Root);
                while(bt.find(Relative.LeftChild));
                bt.update(mostRight);
        }
}
```