CSC 212 Tutorial #8 Solution
Binary Trees

Problem 1

```
public int countLeafNodes() {
    return countLeafNodesRec(root);
}

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

// non-recursive solution
public int countLeafNodes() {
    if (root == null)
        return 0;
    Stack<BTNode<T>> st = new LinkedStack<BTNode<T>>();
    st.push(root);
    int count = 0;
    while (!st.empty()) {
        BTNode<T> p = st.pop();
        if (p.left ==  null &&  p.right == null)
            count++;
        else {
            if (p.left != null)
                st.push(p.left);
            if (p.right != null)
                st.push(p.right);
        }
    }
    return count;
}
```

Problem 2

```
public int getHeight() {
    return getHeightRec(root);
}
```

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

Problem 3

```
public static <T> void swapMost(BinaryTree<T> bt) {
    if (bt.empty())
        return;
    bt.find(Relative.Root);
    while (bt.find(Relative.LeftChild));
    T left = bt.retrieve();
    bt.find(Relative.Root);
    while (bt.find(Relative.RightChild));
    T right = bt.retrieve();
    bt.update(left);
    bt.find(Relative.Root);
    while (bt.find(Relative.LeftChild));
    bt.update(right);
}
```