**Exercise 4.31**

**public** **class** Exercise4\_31<T> {

**public** **class** BinaryNode<T> {

**public** BinaryNode<T> left;

**public** BinaryNode<T> right;

}

//Part A) Finds Number of nodes in tree t

//O(N) runtime

**public** **int** nodes(BinaryNode<T> t){

**if** (t!=**null**){

**return** 1 + nodes(t.left) + nodes(t.right);

} **else** **return** 0;

}

//Part B) Finds number of leaves in tree t

//O(N) runtime

**public** **int** leaves(BinaryNode<T> t){

**if** (t.left==**null** && t.right==**null**)

**return** 1;

**else** **if** (t.left!=**null** && t.right!= **null**)

**return** leaves(t.left)+leaves(t.right);

**else** **if** (t.left!=**null**)

**return** leaves(t.left);

**else** **return** leaves(t.right);

}

//Part C) Finds number of full nodes in tree t

//O(N) runtime

**public** **int** fullNodes(BinaryNode<T> t){

**if** (t!=**null**){

**if** (t.left!=**null** && t.right!=**null**)

**return** 1 + fullNodes(t.left) + fullNodes(t.right);

**else**

**return** fullNodes(t.left) + fullNodes(t.right);

} **else** **return** 0;

}

}

Exercise 6.8

A) Because in a heap children must be larger than their parents, it impossible

for a node that has children to be the maximum. The largest element must be a

leaf because the largest element, by the heap–order property, does not have

any children–any node with children is at the least smaller than those children

B) In a binary tree of height h, there are (2^h) leaf nodes, and (2^(h+1) - 1)

total nodes. Thus, in a binary heap, there are always at most 2^h leaf nodes

(this occurs if it is a perfect binary heap). Because 2\*(2^h) = (2^(h+1) - 1)),

we can see that the total number of leaf nodes is half of the total number

of nodes in the tree.

C) Because the maximum has to be a leaf, we know that we can find the maximum by

examining the leaves. Unfortunately, the heap-order property says only that

children must be larger than their parents; this means that while all leaves are

larger than the parents (and the maximum is therefore definitely a leaf), we must

actually check each leaf to find it.