Ben Walker
CPSC 5081: Data Structures + Algorithms
HW # 4

1) Exercises 5.3 #1

```
public int LevelsDivideConquer(bt head) {
if(head == null){
    return 0;
}
else {
    return Max(LevelsDivideConquer(bt head.left),
    LevelsDivideConquer(bt head.right)) + 1;
}
private int Max(int first, int second){
    if (first > second){
        return first;
    else{
        return second;
    }
}
```

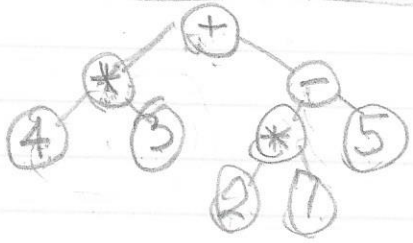
$T(n) = \Theta(n)$ because you have to traverse every node before determine which subtree path has the most levels.

* 2) Exercise 5.3 #6.

```
public void InOrderTraversal(bt head){
    if(head.left != null){
        InOrderTraversal(head.left);
    }
    Consol.Write(head.data);
    if(head.right != null){
        InOrderTraversal(head.right);
    }
}
```

T(n) =

3) Section 5.3 - BST traversal




a. preorder (C L R):
+ * 4 3 - * 2 7 5

b. inorder (L C R):
4 * 3 + 2 * 7 - 5

c. postorder (L R C):
4 3 * 2 7 * 5 - +

4) Section 6.4 - Heap construction

Bottom-up heap construction inserts new nodes at end and percolates up. Top-down heap construction inserts new nodes at the root and percolates down. Bottom-up is O(n) b/c in the percolate up, you have to compare the new node against every other node. Top-down is O(log n) because you only have to traverse 1 of 2 subtrees in the percolate down portion.

5) Section 6.4 - Heapsort: Assuming Mat Heap