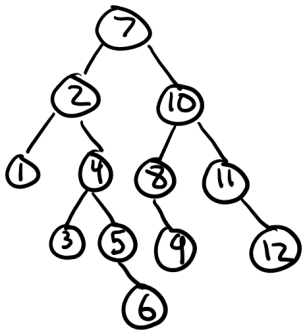
HW 2: trees (Braden Morrison, 430002127)

Problem 1 (14 points)

(1) (2 points) Show the result of inserting 7, 10, 2, 4, 5, 11, 1, 8, 9, 6, 12, 3 into an initially empty binary search tree (in that order).



a. (2 points) Which node is the root?
- 7

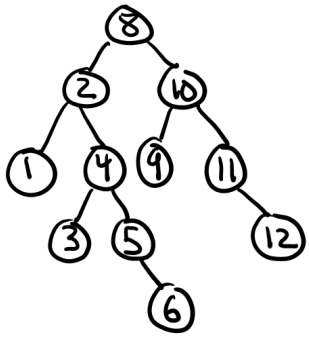
b. (2 points) Which nodes are the leaves?
- 1,3,6,9,12

c. (2 points) Which nodes are siblings? (a,b) -> a and b are siblings
- (2,10), (1,4), (3,5), (8,11)

d. (2 points) What is the depth of each node? A:2 -> A has a depth of 2
- 7:0, 2:1, 10:1, 1:2, 4:2, 8:2, 11:2, 3:3, 5:3, 9:3, 12:3, 6:4

e. (2 points) What is the height of each node? A:2 -> A has a height of 2
- 7:4, 2:3, 1:0, 4:2, 3:0, 5:1, 6:0, 10:2, 8:1, 9:0, 11:1, 12:0

(2) (2 points) Show the result of deleting the root.



Problem 2 (9 points)

(1) (6 points) Write efficient functions that take only a pointer to the root of a binary tree, T, and compute

a. The number of nodes in T.

```
int Numnodes(root) {
    if(root == nullptr) {
        return 0;
    }
    int l = Numnodes(root->left);
    int r = Numnodes(root->right);
    return 1 + r + l;
}
```

b. The number of leaves in T.

```
int Numleaves(root) {
    if (root == nullptr) {
        return 0;
    }
    if (node->left == nullptr && node->right == nullptr) {
        return 1;
    }
    else {
        return Numleaves(root->left) + Numleaves(root->right);
    }
}
```

c. The number of full nodes (nodes with 2 children) in T.

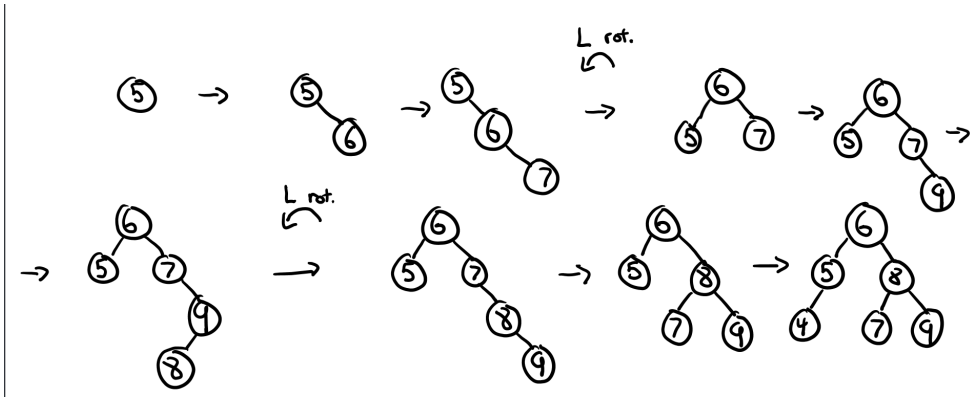
```
int fullNodes(root) {
    int count = 0;
    int leftside = 0;
    int rightside = 0;
    if (root == nullptr) {
        return 0;
    }
    if (root->left != nullptr && root->right != nullptr) {
        count++;
    }
    leftside = fullNodes(root->left);
    rightside = fullNodes(root->right);
    count = count + leftside + rightside;
    return count;
}
```

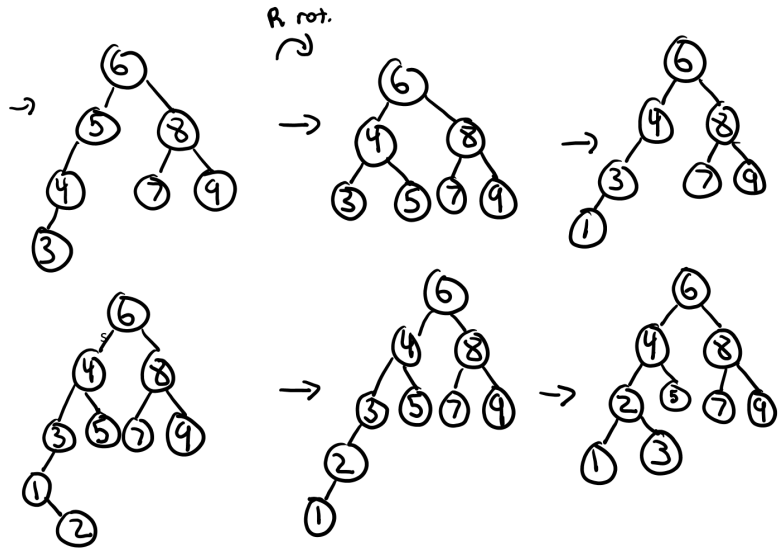
(2) (3 points) What are the time complexities (use big-O notation) of your routines?

- All of the time complexities are O(n).

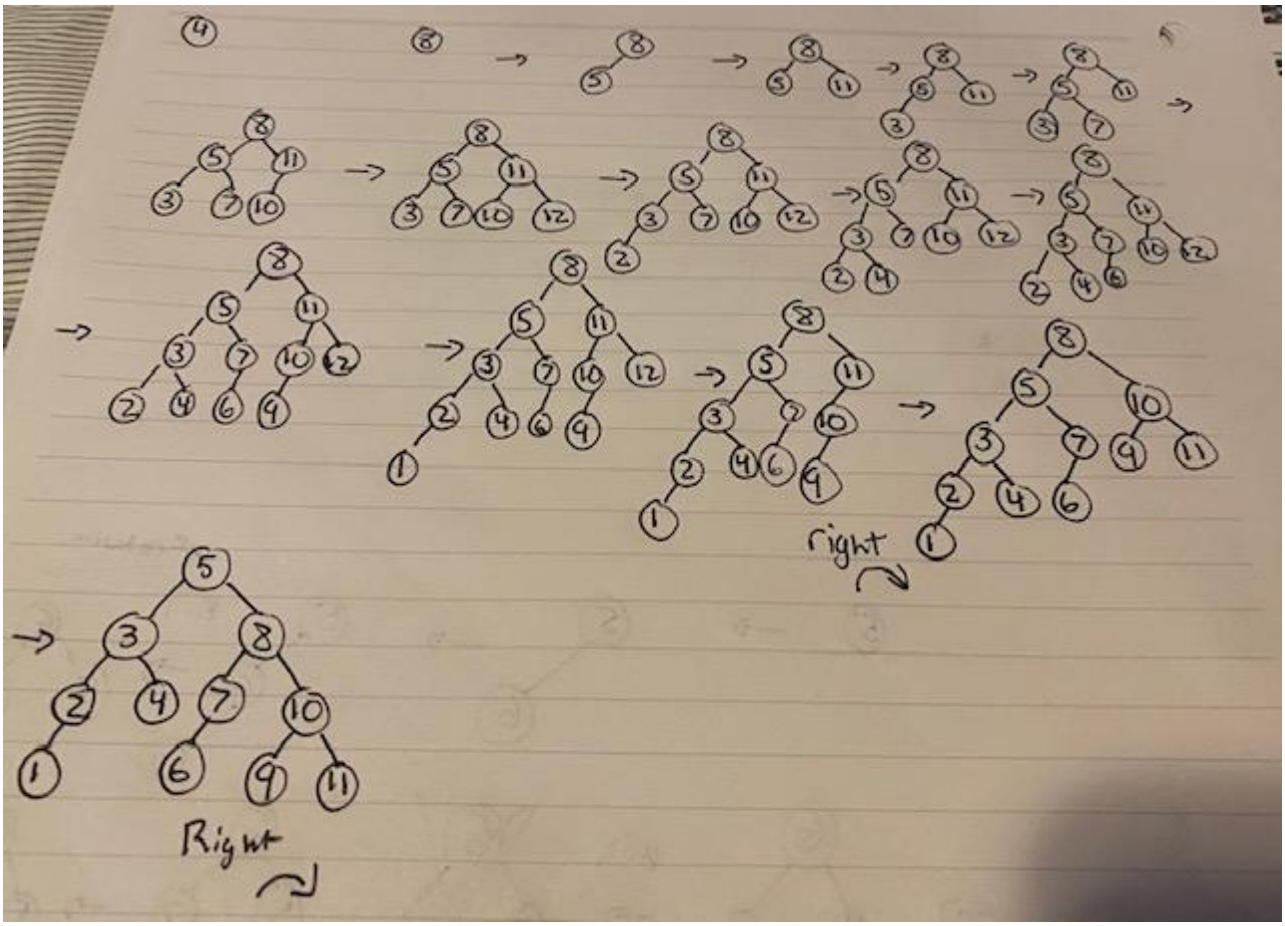
Problem 3 (2 points)

Insert 5, 6, 7, 9, 8, 4, 3, 1, 2 into an initially empty AVL tree. This should require all 4 types of rotations.





Problem 4 (3 points)
Insert 8, 5, 11, 3, 7, 10, 12, 2, 4, 6, 9, 1 into an initially empty AVL tree, then delete 12. This should require 2 rotations.

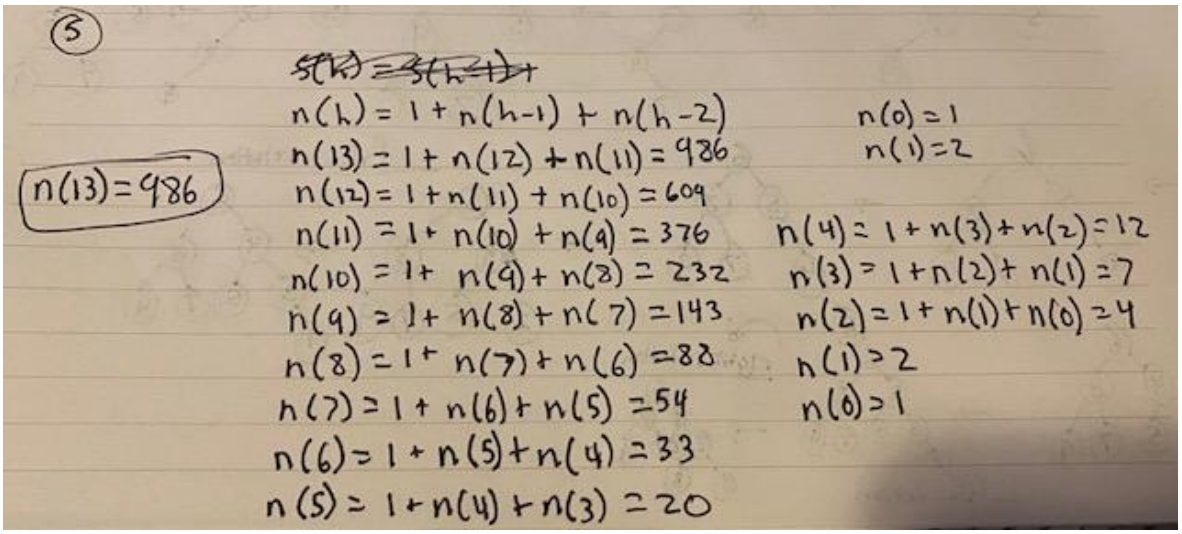


Problem 5 (2 points)

Compute the minimum number of nodes in an AVL tree of height 13.

You do not need to find a closed form solution. Just work out what the value is for height = 13.

Hint: use a recurrence relation.



Problem 6 (3 points)

Show that every AVL tree can be colored as a red-black tree. Are all red-black trees AVL?

- Every AVL tree can be colored as a red-black tree, however, not all red-black trees are AVL trees. To color an AVL tree as a red-black tree, we must start at the root of the AVL tree and recursively color its left and right subtrees in a way that maintains balance and the color constraints. At each step, we can use the height difference between the left and right subtrees to determine whether the current node should be colored red or black, and we can adjust the colors of its children accordingly.

Problem 7 (2 points)

Insert 5, 6, 7, 9, 8, 4, 3, 1, 2 into an initially empty red-black tree.