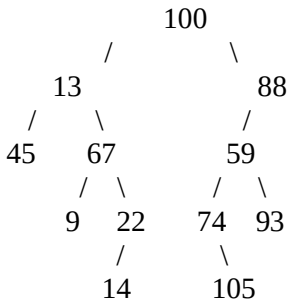
COP 3502 Quiz #4 Version A (Binary Search Trees, AVL Trees, Tries) Solutions

1) (6 pts) Provide the Preorder, Inorder and Postorder traversals of the following binary tree:



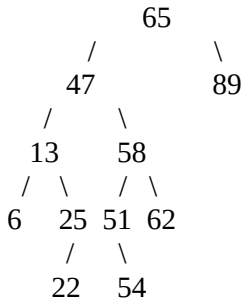
Pre-Order: 100, 13, 45, 67, 9, 22, 14, 88, 59, 74, 105, 93

In-Order: 45, 13, 9, 67, 14, 22, 100, 74, 105, 59, 93, 88

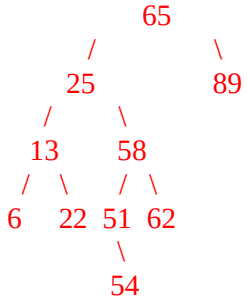
Post-Order: 45, 9, 14, 22, 67, 13, 105, 74, 93, 59, 88, 100

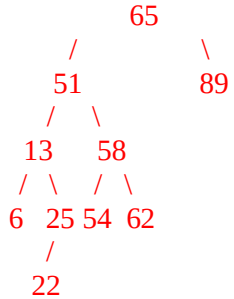
Grading: 2 pts for each correct traversal, 1 pt if more than 1/2 the values are right, if answers are all traversals but named incorrectly (they put pre-order for post-order or something like that), then -2 total.

2) (5 pts) Show the result of deleting 47 from the binary search tree shown below. (Note: There are two possible right answers.)



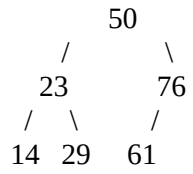
We can either replace 47 with the max on its right (25), or the min on its left (51). Here is the result of both (so both correct answers):





Grading: 2 pts valid replacement, 2 pts for patch below, 1 pt for copying rest

3) (9 pts) In a binary search tree, consider adding all the values at an even depth from the root, and subtracting all the values at an odd depth from the root. For example, for the tree shown below:



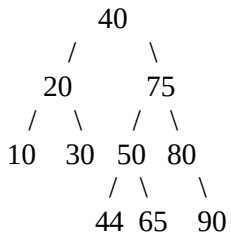
The desired sum would be 50 - 23 - 76 + 14 + 29 + 61 = 10.

Write a recursive function that calculates this adjusted sum for a binary search tree, given a pointer to its root. **(Hint: If I am a tree rooted at 50 above, then the contribution to my sum is the NEGATIVE of the corresponding sum for each of my subtrees.)** Please use the struct and function prototype given below:

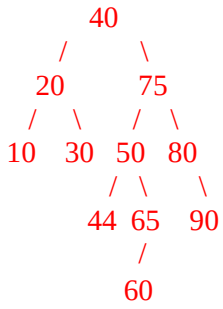
```
typedef struct bintreenode {
    int data;
    struct bintreenode* left;
    struct bintreenode* right;
} bintreenode;

int getAdjustedSum(bintreenode* root) {

    if (root == NULL) return 0;

    return root->data - getAdjustedSum(root->left)
                      - getAdjustedSum(root->right);

}
```

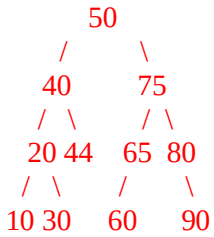
Grading: 2 pts base case, 1 pt return reg case, 2 pts for adding data, 2 pts for subtracting recursive call on left, 2 pts for subtracting recursive call on right

4) (5 pts) Show the result of inserting the value 60 into the AVL tree below. Put a box around your final answer.



When we do this, we get an imbalance at 40, the rebalancing (and final answer) is on the right:





Grading: 5 pts correct answer, 2 pts if shows insertion without any rebalance, partial if rebalance is attempted but incorrect (so give either 3 or 4 based on your discretion)

5) (10 pts) Complete the function below so that it counts the number of words in the trie pointed to by root and returns this value. Please use the struct definition and function prototype shown below:

```
typedef struct trie {
    int isWord;
    struct trie* next[26];
} trie;

int numWords(trie* root) {

    if (root == NULL) return 0;
    int res = root->isWord;
    for (int i=0; i<26; i++)
        res += numWords(root->next[i]);
    return res;
}
```

Grading: 2 pts base case, 2 pts for adding in isWord for root, 2 pts loop, 3 pts adding in each of the 26 recursive calls, 1 pt final return