1) (10 pts) DSN (Dynamic Memory Management in C)

Consider a binary search tree, where each node contains some key integer value and data in the form of a linked list of integers. The structures are shown below: the tree nodes and list nodes are dynamically allocated. We are going to eventually upgrade the structure, and when we do so, all of the dynamically allocated memory will be deleted (including all of the linked lists). Write a function called `deleteTreeList` that will take in the root of the tree, freeing all the memory space that the tree previously took up. Your function should take 1 parameter: a pointer to the root. It should return a null pointer representing the now empty tree.

```
typedef struct listNode {
    int value;
    struct listNode * next;
} listNode;

typedef struct treeNode {
    struct treeNode * left;
    struct treeNode * right;
    int value;
    listNode * head;
} treeNode;

treeNode * deleteTreeList (treeNode * root)
{
    // Checking the tree is empty 1 pt
    if (root == NULL)
        return NULL; // returning NULL 1 pt

    // Loop through (iterative or recursive) list at the node 1 pt
    while (root->head != NULL) {
        listNode * tmp = root->head; // Prevent use after free 1 pt
        root->head = root->head->next;
        free(tmp); // free 1 pt
    }

    // Freeing both children 1 pt each
    root->right = deleteTreeList(root->right);
    root->left = deleteTreeList(root->left);

    // Freeing the tree's root 1 pt
    free(root); // If the root is free'd last 1 pt

    return NULL; // returning NULL at end 1 pt
}
```