3) (10 pts) DSN (Tries)

Write a recursive function that takes the root of a trie, *root*, and a single character, *alpha*, and returns the number of strings in the trie that contain that letter. You may assume that letter is a valid lowercase alphabetic character ('a' through 'z').

Note that we are *not* simply counting how many times a particular letter is represented in the trie. For example, if the trie contains only the strings "apple," "avocado," and "persimmon," then the following function calls should return the values indicated:

```
countStringsWithLetter(root, 'p') = 2
countStringsWithLetter(root, 'm') = 1
```

The TrieNode struct and function signature are given below. You may assume that the variable numwords accurately stores the number of valid words stored (# of nodes within the trie with flag set to 1) in all trie structs.

```
typedef struct TrieNode {
    struct TrieNode *children[26];
    int numwords;
    int flag; // 1 if the string is in the trie, 0 otherwise
} TrieNode;

int countStringsWithLetter(TrieNode *root, char alpha) {




}
```