3) (10 pts) DSN (Tries)

It's often useful to know how many words start with a particular prefix. Given a trie that stores a dictionary of valid words (**lowercase letters only**) as well as a prefix string, write a **non-recursive** function that calculates the number of words that begin with that prefix. To aid you in your solution, the struct that stores a trie node will not only store whether or not that node represents a word or not, but it will **also** store the total number of words stored within that subtree of the trie in a variable called numwords. You may assume that the TrieNode pointer passed to the function represents the root of the whole trie storing the dictionary of words. You may assume that root is NOT NULL and prefix has at least one lowercase letter in it.

```
#include <string.h>

typedef struct TrieNode {
    struct TrieNode *children[26];
    int flag; // 1 if the string is in the trie, 0 otherwise
    int numwords; // the total # of words stored in this sub-trie.
} TrieNode;

int numWordsWithPrefix(TrieNode* root, char* prefix) {

    // 1 pt var declarations.
    int i, len = strlen(prefix);

    // 1 pt loop
    for (i=0; i<len; i++) {

        // 3 pts NULL check
        if (root->children[prefix[i]-'a'] == NULL)

            // 1 pt return for this case.
            return 0;

        // 3 pts advancing pointer down trie.
        root = root->children[prefix[i]-'a'];
    }

    // 1 pt return value.
    return root->numwords;

}
```