ECE 250 Project 3
Spell-Checking with Trie

1. Overview of Classes

I've implemented three classes to create the Trie spell-checking function.

Class tree_node

- The tree_node class generates a node that stores a char value which represents a letter of the alphabet. Each node generated has children which contain the next letter of a word

Member Variables:

1. value (char) - stores a letter of the alphabet
2. isLeaf (bool) – states whether the node is a leaf (true) or not (false)
3. isWord (bool) – states whether the node is the end of a word (true) or not (false)
4. children (vector<TreeNode*>) – collection of child nodes (one for each letter of the alphabet) for each node

No Member Functions

Class trie

- The trie class consists of multiple nodes that point to the next node to form words. This class contains functions that can add or remove words into the Trie, this allows the Trie to be used as a spellchecker. There are also additional functions that return certain characteristics of the Trie.

Member Variables:

1. root (TreeNode*) – points to the first node of the Trie and takes on the value '*'
2. current (TreeNode*) – points to the current node when traversing through the Trie
3. size (int) – the number of words in the trie

Member Functions:

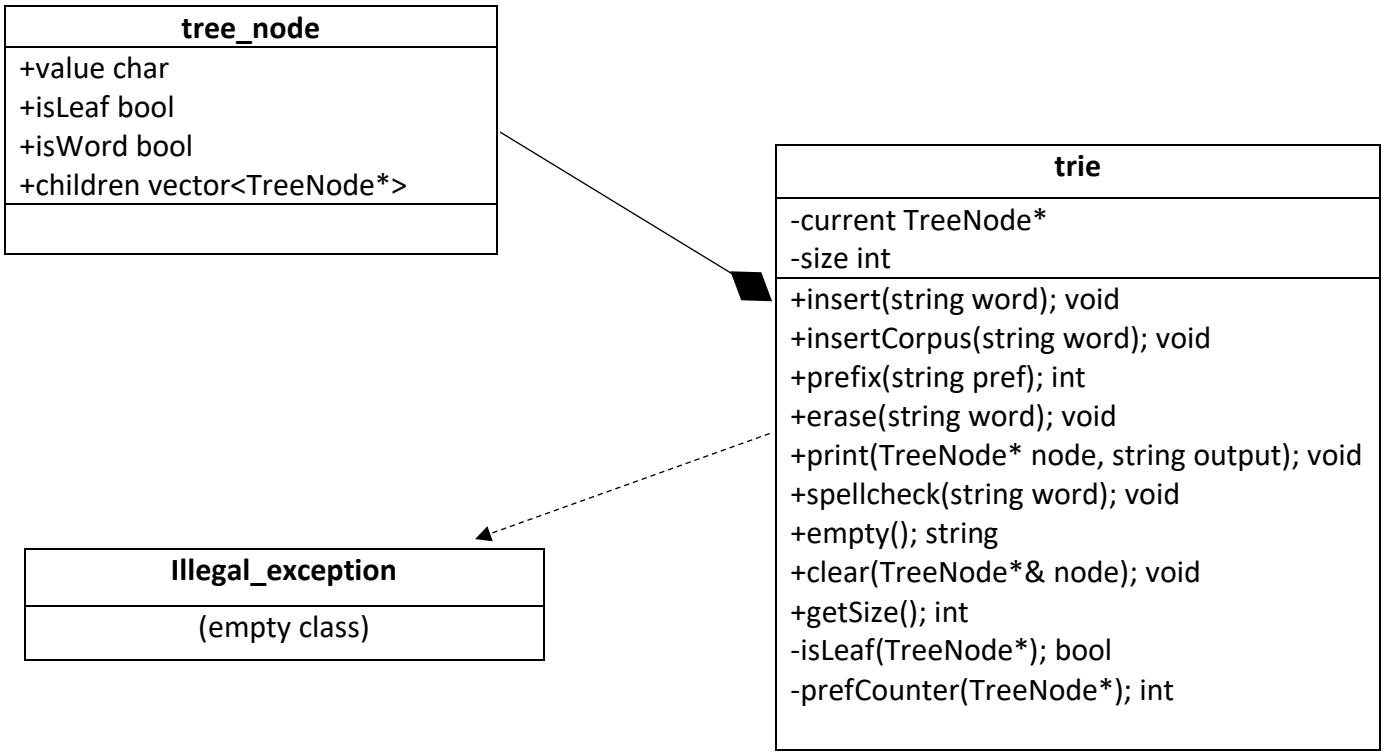
1. insert – Inserts the word into the trie and increments the size variable by 1, unless the word already exists in the trie.
2. insertCorpus – Inserts all the words from the "corpus.txt" file into the trie, excluding all duplicate words and incrementing the size variable each time a new word is inserted.
3. prefix – Takes in a prefix and returns the number of words in the trie that start with that prefix.
4. erase – If the word exists in the trie, the erase function removes the word and decrements the size variable by one.
5. print – Prints all the existing words in alphabetical order from the trie.
6. spellcheck – Takes in a word, and checks if it exists in the Trie. Print "correct" if the word exists, else offer suggestions. When the word is not spelt correctly, it will print suggestions starting with the first letter with an error.
7. empty – Checks if the trie is empty and returns either "empty 0" if the trie is empty or "empty 1" if the trie is not empty.
8. clear – Deletes all the nodes in the trie except for the root.
9. getSize – Returns the size variable which holds the number of words in the trie.
10. isLeaf – Checks to see if a node has any children. Returns true if no children are found and returns false if there are children.

11. prefCounter – Takes in a node from the prefix function to count the number of words with that prefix.

Class illegal_exception

- An empty class that is thrown when an error occurs with invalid input

2. UML Class Design



3. Constructors/Destructors and Design Decisions

tree_node Class:

The constructor for this class passes one parameter and sets the node's letter value. The variables isLeaf and isWord are both set to null when the node is created. The vector children is resized to 26, to account for the 26 letters in the alphabet. During run time, isLeaf and isWord may be toggled to true true, depending on whether the node is a leaf or the last letter of a word.

trie class:

The constructor for this class creates an empty trie, where TreeNode* root is defined as a new TreeNode* with char '*' and the current node is set to a nullptr. The rest of the trie is populated at run time

For the destructor, the clear function is used to delete all nodes stored in the trie except for the root. Then at the end of the code, the root is deleted. Deleting all the nodes ensures that memory leaks will not occur.

No operators were overloaded in my code, as there was no need.

4. Runtime Implementation

For the insert and erase function a runtime of O(n) is implemented, where n is the number of characters in the word being removed or inserted. This is the case since it must traverse through the entire word to add or remove the chars to and from the trie. Member functions prefix, print, spellcheck and clear require a runtime of O(N) where N is the number of words in the trie. For prefix, print and spellcheck, these commands must traverse through the whole trie to find words. For the clear function, the whole trie must be traversed in order to delete every node in the trie. The runtime O(1) is implemented for functions getSize and empty. The getSize function returns the number of words in the trie. Since the size variable is incremented and decremented when the insert or erase function is called, calling getSize only needs to return the value of the size variable. Furthermore, the empty function only needs to check if the size of the trie is 0 or not (by calling getSize), thus executing the code instantly.