A prefix tree is a data structure that stores a collection of words in a tree structure. Each node in a prefix tree represents a single character in a word, and it has up to 26 children, one for each letter of the alphabet. Since words can appear inside other words, prefix tree nodes also keep track of whether a word ends after their character.

For example, the prefix tree containing the words be, bee, bed, best, and cat would look like the tree below (nodes that end words have a double circle):

Note that the root node doesn't have an associated letter; the first letter of the word is a child of the root node. Also, nodes will never have more than one child for the same letter: the words "be", "bee", "bed", and "best" all share the "b" node below the root.

Your task for this lab is to implement two member functions for a PrefixTree class. The first member function is addWord. This function accepts a string, and it adds nodes to the PrefixTree so that this word is included. For example, to add the word "car" to the tree above, you would need to add a new node containing "r" below the "a" node. To add "dog", you would need to add 3 new nodes ("d", "o", and "g") below the root node.

The second member function to implement is contains. This function accepts a string and returns whether that word has been added to the PrefixTree. The prefix tree above contains "be", but it doesn't contain "beer" or "bes".

You have been provided with some code to get you started.

- prefixtree.h
- prefixtree-base.cpp
- prefixtree-driver.cpp

The most relevant member functions are described below:

- PrefixTree::getRoot(): returns the root node of the PrefixTree
- PrefixTreeNode::getChild(char): returns the child of this node corresponding to the given letter (null pointer if that child doesn't exist)
- PrefixTreeNode::addChild(char): adds a child to this node that corresponds to the given letter
- PrefixTreeNode::isFinal(): returns whether this node is the end of a word
- PrefixTreeNode::setFinal(bool): changes whether this node ends a word

The provided driver file accepts 4 different commands. The "quit" command ends the program, while typing "add word" will add a word to the PrefixTree (replace "word" with what you want to add). Typing "search word" will print out whether the tree contains that word, and typing "print" will print out a representation of the PrefixTree. The print functionality prints children after their parent, and if a node has multiple children, they appear as a comma-separated list in parentheses. Letters that end a word are capitalized. For example, the prefix tree in the example above would be printed as

"(bE(D,E,sT),caT)".  The print function is already implemented, so you can use this to test your addWord function.

You should submit a source file that contains the definitions of the missing addWord and contains member functions of PrefixTree.