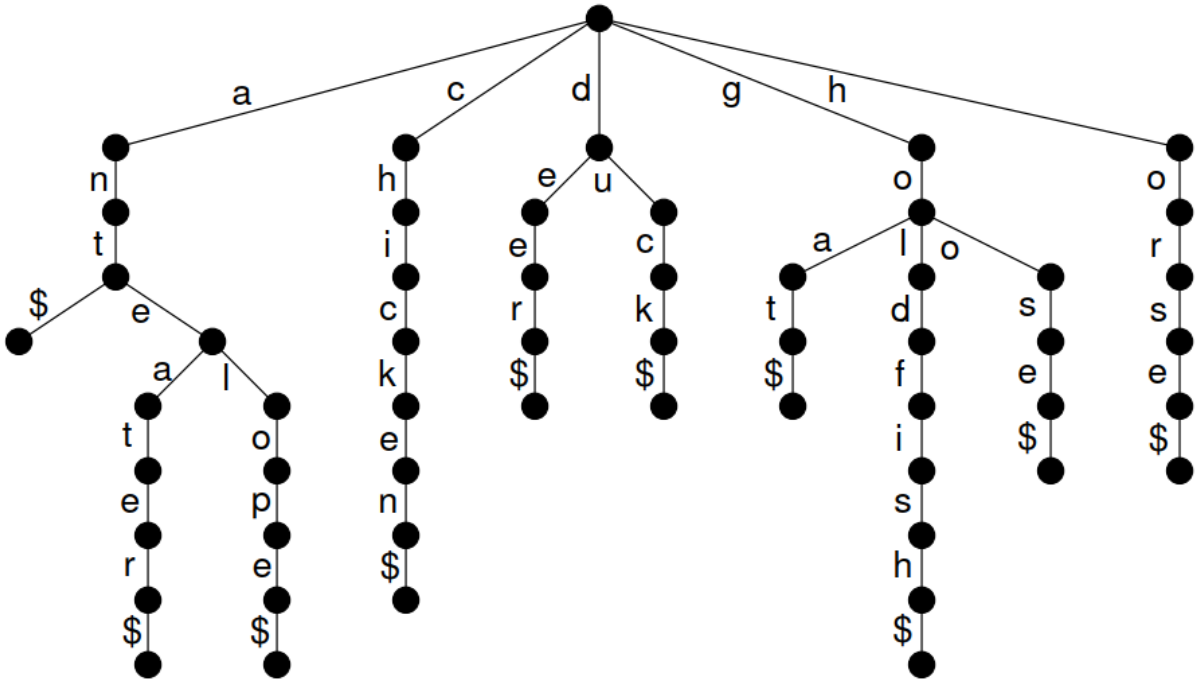
CS 211 PA #3: Text Autocomplete

In this assignment, you will build a trie (pronounced "try") to perform auto complete-like behavior on a string. Chapter 13.1 in the book has a great discussion on how Tries work. I highly suggest starting the assignment by first reading the book. Wikipedia also has a pretty good article on Tries. While there are multiple trie-based dictionary implementation strategies, we will take the most straightforward approach. Each node in the trie will represent a single character in a word. Trie nodes that have the '$' sentinel value as a child indicate the end of a word. Words are constructed by performing a pre-order traversal through our trie, stopping at the sentinel value. Here's an example trie pulled from the book:

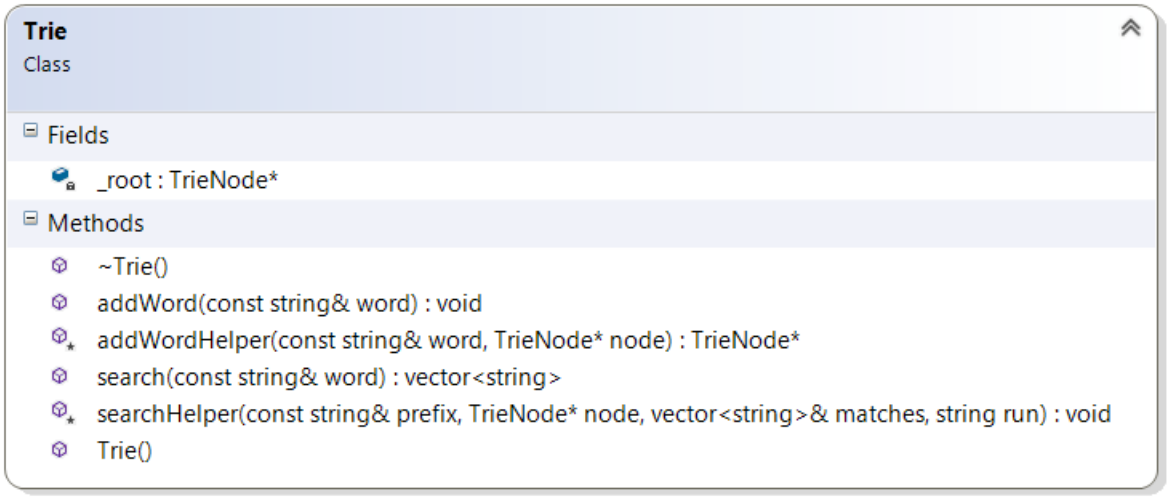


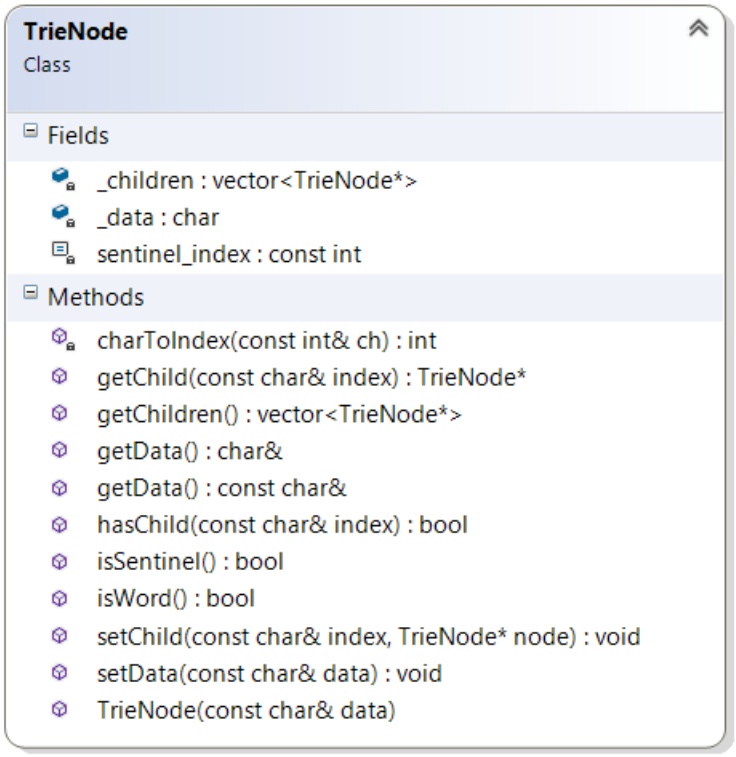
Note that each branch represents a possible character sequence in a word. Thus, in the 'a' subtree, we are aware of the words "ant", "anteater", and "antelope."

For this assignment, your program will prompt the user for a list of known words in a dictionary. From this, you must construct a trie. Next, the user will supply prefixes to words. Your program will supply "autocomplete" suggestions for the supplied prefix. So, for example, if the user entered "ante", your program would return the words "anteater" and "antelope" but not "ant".

UML Diagram

Provided below is the full UML class diagram for this assignment.





Starter Code

Included with this document is a set of starter code. Your solution must be based on this starter code. In order to successfully complete this assignment, you will need to implement the missing functions located in the file Trie.h. Note that in order to fully implement missing functionally, it may be necessary to specify one or more additional helper functions. The names and purposes of these extra functions are left to your discretion.

Possible Strategy for Getting Started

1. Start by implementing the addWord function. Note that I used a recursive helper function to make the work easier.
    a. If you go the recursive route (highly recommended), what information is required at each stage of the recursive call. What are the bases cases?
2. After implementing the addWord function, next implement the search function. Again, I used recursion to solve this problem.
    a. Remember, we search for words using an in-order traversal.
    b. Note that it might be helpful to pass a reference to the final vector of strings to the recursive call.
    c. Words are added to the final result whenever we encounter a node that has the sentinel as a child.

Header Comment, and Formatting

1. Be sure to modify the file header comment at the top of your script to indicate your name, student ID, completion time, and the names of any individuals that you collaborated with on the assignment.
2. Remember to follow the basic coding style guide. A basic list of rules is included with this document.

Reflection Essay

In addition to the programming tasks listed above, your submission must include an essay that reflects on your experiences with this homework. This essay must be at least 350 words long. Note that the focus of this paper should be on your reflection, ***not*** on structure (e.g. introductory paragraph, conclusion, etc.). The essay is graded on content (i.e. it shows deep though) rather than syntax (e.g. spelling) and structure. Below are some prompts that can be used to get you thinking. Feel free to use these or to make up your own.

- Describe a particular struggle that you overcame when working on this programming assignment.
- Conversely, describe an issue with your assignment that you were unable to resolve.
- Provide advice to a future student on how he or she might succeed on this assignment.
- Describe the most fun aspect of the assignment.
- Describe the most challenging aspect of the assignment.
- Describe the most difficult aspect of the assignment to understand.
- Provide any suggestions for improving the assignment in the future.

Deliverables

You must upload your program and reflection as a ZIP file through Canvas no later than midnight on Monday, March 5, 2018.

PA #3 Checkins

During lab on 2/26, you must demonstrate a completed version of the search function. The search function is provided a prefix (e.g. 'a'). It will return all words that begin with the supplied prefix. For example, passing a prefix on 'a' would return "apple" but not "dog".

Both the walk and search functions will be helpful in solving this assignment.

Grading Criteria

Your assignment will be judged by the following criteria:

Reflection essay (10pts)

- Your reflection meets the minimum requirements as specified earlier in this document.

PA Checkin (10pts)

- You successfully implement the search function prototype found in the supplied main.cpp starter code.

Style (20pts)

- Your project contains good structure and implements the required classes. Your program intelligently uses classes when appropriate and generally conforms to good OOP design (i.e. everything isn't slapped into main).

Dynamic Memory Management (5pts)

- You dynamically allocate memory in your program.
- You correctly clean up all dynamically allocated memory in your program.

Text Prediction (40 pts)

- Your program correctly displays all possible matches for a given input word sequence.

Grade Distribution

Your final grade for the assignment will be determined based on the number of points earned. Note that failing to use good design (e.g. objects, appropriate data structures), regardless of score earned, may result in a lower overall grade.

Score	Points Required
100	90
90	85
80	75
70	60
60	50
50	40
40	30
25	20