Peter Pan u0680482

Zhi Wang u0761669

Assignment 4 Write-up

Because we have two classes to support our Trie class, we have written to test cases for each class respectively. The “TrieUnitTests.cpp” file tests the Trie class at a mostly high level, and the “NodeUnitTests.cpp” file tests the *node* class at a lower level. The idea behind this is so that if there is a problem, the programmer can more easily identify which class might be causing the problem. Also, we chose not to list all of our tests here in this write-up because it would be a relatively long list. Instead we chose to give a brief summary of the idea and purpose behind each test.

Beginning with the *node* class, we start with a few methods that verify the functionality of the *refCount* variable. The purpose of this variable is to count the number of nodes that are created in the Trie. There are various tests to test adding nodes in a straight forward fashion as well as adding them in them to child nodes that are already created. There is also a test to check if the destructor properly destroys the node and all it’s children in a recursive manner when it is not needed anymore. Additionally, are a couple methods to test the copy constructor and the assignment operator. Following that, there are several methods that test the *addChildNode* method checking for problems that may arise such as adding an invalid character or when adding duplicate nodes that already exist. The methods *getChildNode* and *childNodeExists* are also tested for invalid characters as well as for their basic functionality of retrieving a child node and checking if the node exists .

In the tests for the *Trie* class we follow a similar pattern. The first set of test check the functionality of the *addWord* method. The *addWord* method accepts a string, but does not return a value, instead essentially adds a word to the root node. A couple of the testing methods test basic functionality of just inserting a word and making sure it’s there. One of the methods inserts a word and checks to see if a prefix of that word would count as another word, which according to the specs, it would not. Some of the methods also test inserting an empty string, and adding an entire dictionary file to the Trie just to make sure it can handle something of that size. After that, several methods test the *isWord* method. For this method, a test also double checks that it can handle newline characters, making sure that even if a word has a newline character at the end, it is still considered a word. Other methods check the functionality of passing in an empty string or a word that does not exist in the Trie. A few more methods test the *allWordWithPrefix* method. Just like the previous two methods before it, this method should accept words with newlines without a problem. Additionally, a couple tests test the functionality of the method by checking the number of words that are returned by the method. The reaming methods test the functionality of the destructor, copy constructor, and the assignment operator.