**EE422C Project 3 (Word Ladder) Team Plan**

Jon Ambrose ja38435

Neil Abhishek Charles nc23227

Spring 2017

Responsibilities:

Neil initially completed the BFS search and findEdge method.

Jon will complete the DFS search method.

Jon and Neil implemented the printLadder function together.

Neil and Jon will implement the parse function. Jon will test the parse function.

Jon will test Neil's BFS method and Neil will test Jon's DFS method.

All testing, communication, and debugging is done either in person or via google hangouts video call.

BFS Methodology:

For the BFS we go through all the nodes in the dictionary Node arraylist to find the start node. We set it as head and add it to the queue. While the queue is not empty, check to see if the current head of the queue is the word. If it is not then for each child of head, if it is not visited, add it to the queue. And, if it is found, place the end node in an arraylist and go up its parent until the parent is null. Then reverse the arraylist to get the word ladder in order.

DFS Methodology:

DFS sets up a recursive call to DFSInner() to implement the DFS search algorithm. The function passes in the starting Node, the string the search is looking for, and a Node array that will hold the final Node if it is found. DFSInner base cases are if the starting Node has an empty adj list, return false, or if we find the Node we where looking for, return true. If one of those base cases are not met, DFSInner calls itself again on every Node in the starting adj list. This search is exhaustive but is guaranteed to return either the desired Node or an empty array. Once DFSInner completes, DFS will build the resulting array list using the final node and it's parental linkage. Finally it returns the result to the calling function.