**EE42C Project 3 (Word Ladder) Test Plan**

Ram Muthukumar rm48763

Mohit Joshi msj696

Summary

Our initial procedure consisted of us working separately on modules for the design that we discussed and came up with prior to even starting the code. We tested the modules individually to make sure they worked before joining our work to form the finished product. Once we had the completed project, we moved to JUNIT testing, in which we again tested each component of the program for bugs. We tested many aspects of the design, such as correct initialization and word ladder formation, along without looking for duplicates. Perhaps we could have covered the relationship between each rung more in testing, since we did not test every rung in the huge ladders after they were constructed to check that each word is one letter away. This should not be a problem, however, because our design only allowed for words to be connected to neighbors that are actually in the dictionary and are indeed neighbors.

Tests

1.

a) Name: testInit()

b) Function: Test correct initialization of data structures and dictionary

c) Setup: no setup, only assumes dictionary file in directory

d) Output: none, passes if all used structures are not null

e) Pass/Fail: fails if dictionary is no populated and if bfsqueue, markedWords, and neighbors are null

f) Comments: quick test

2.

a) Name: bfsNormalTestPass()

b) Function: checks if the bfs method will pass a simple input

c) Setup: none

d) Output: the constructed word ladder between flood and front

e) Pass/Fail: passes if the ladder is not empty but less than 10 elements long

f) Comments: the outputted word ladder should be simple and short

3.

a) Name: bfsNormalTestFail

b) Function: Test if bfs fails for two words with no ladder between

c) Setup: none

d) Output: none

e) Pass/Fail: Passes if the word ladder constructed is null or empty

f) Comments: this function should pass for any words with no connection

4.

a) Name: bfsTestIfNeighbors

b) Function: Test if two neighbors are viewed as neighbors

c) Setup: pass two neighbor words into the bfs function

d) Output: none

e) Pass/Fail: Passes if the word ladder constructed is less than 3 elements

f) Comments: this function should pass by constructing a ladder with zero rungs and only two elements: the given inputs

5.

a) Name: noDuplicatesBFS

b) Function: Test if there are any duplicates in the ladder from bfs method

c) Setup: none

d) Output: none

e) Pass/Fail: Passes if the word ladder contains no duplicate elements

f) Comments: none

6.

a) Name: testParse()

b) Function: Test if the parse function gets two inputs and converts them to capital letters

c) Setup: pass a lower-case input into parse

d) Output: none

e) Pass/Fail: Passes if parse returns an ArrayList containing the input

f) Comments: none

7.

a) Name: dfsNormalTestPass()

b) Function: test if a ladder will be constructed with two words with a connection

c) Setup: pass two inputs with a connection to dfs method

d) Output: prints the ladder

e) Pass/Fail: Should pass as long as inputs have a connection

f) Comments: none

8.

a) Name: dfsNormalTestFail()

b) Function: test if no ladder will be constructed with inputs without a connection

c) Setup: pass two inputs with no connection to dfs

d) Output: none

e) Pass/Fail: Passes if there is no ladder constructed

f) Comments: none

9.

a) Name: noDuplicatesDFS()

b) Function: Test if there are duplicates in the ladder

c) Setup: pass two inputs with a connection

d) Output: none

e) Pass/Fail: Passes if no duplicates in the ladder

f) Comments: none

10.

a) Name: dfsTestIfNeighbors()

b) Function: Test if the dfs will return a zero rung ladder

c) Setup: pass two inputs that are neighbors

d) Output: print the ladder

e) Pass/Fail: Passes if parse ladder size < 2

f) Comments: none