Assignment 4 Testing Outline

In order to accomplish both black box testing and white box testing, we have three different test cases. They are the normalTest(), the noLadderTest(), and the invalidInputTest(). As well as making sure the program methods are behaving properly, these tests will also allow us to step through all statements in the program.

The normalTest() consists of a valid word pair where a word ladder exists between them. This will allow us to step through the computeLadder(), makeLadder(), getDifferenceIndex(), and compareLetters() methods. For black box testing, a valid word ladder must be created. This word ladder will be validated with the validateResult() method. The word pairs chosen will be based on our goal to accomplish white box testing as well. There is a conditional in the computeLadder() method that is reached using word pairs of the same word. This will give us full coverage of the code.

The noLadderTest() consists of a valid word pair where no word ladder exists between. This test steps through computeLadder(), makeLadder(), getDifferenceIndex(), and compareLetters() methods. For black box testing, the valid behavior is to not have created a word ladder. Multiple words pairs that fit these criteria will be used to test this. In the case of white box testing, in this test case the makeLadder() should always return false. Using multiple inputs and stepping through the code will give us full coverage.

The invalidInputTest() consists of word pair where either word cannot be found in the dictionary or the input itself is not in the proper format. This test steps through computeLadder(). For black box testing, the proper behavior is that no word ladder is created and an InvalidArgumentException is thrown. We will use many invalid inputs to make this behavior is always executed. For white box testing, this covers the conditionals in the computeLadder() method that the previous two test cases do not reach.

These three test cases test the three behaviors of this assignment. They also provide full coverage of the program's methods and code.