UNIVERSITY OF WISCONSIN PARKSIDE

Department of Computer Science

CSCI 340/CIS 540 Data Structures/Algorithm Design

Homework #5: 50 points

Outcomes:

- Write programs that use Directed Graphs
- Write programs that use HashMap implementation
- Write programs that use Graph Traversal Technique – BFS

Scoring:

- If you do not submit a .java file containing your source code, your score will be zero.
- If you submit the source code that does not compile, your score will be zero.
- If you submit source code without the correct class name (WordLadder) or your program partially works, you will receive at most half credit for this assignment.
- Deductions will be made for not meeting the usual requirements:
    - The source code is not formatted according to the usual style guidelines, including commenting on each method to explain its purpose.

Create an IntelliJ Java Project called `WordLadder`. Write the comment lines to include the author name(s) and complete the project following the instructions.

Introduction:

A word ladder is a sequence of words each differing from the last by only modifying one letter. For example,

```
good -> food -> fool -> foal -> coal -> coil -> boil
```

is a word ladder of length 7 that takes us from `good` to `boil`.

The goal of word ladder games is to find the shortest sequence from a starting word to an ending word. For example, the above word ladder is not the shortest.

```
good -> food -> fool -> foal -> foil -> boil
```

is a better answer.

Your assignment:

You are to write a program that finds the shortest word ladder between two words. Your instructor has created multiple word files. Each contains words of a given length. words.3 contains three letter words. words.4 contains four letter words, etc. If the words are of different lengths, simply print an error and terminate. Otherwise read the appropriate file. If either of the input words is not in the file, print an error and terminate. If no word ladder can be found, print an appropriate message and terminate. If a word ladder can be found, print the word ladder in a format similar to that above.

This is really a graph search problem. Start by creating a graph of the words and legal transitions between them. Note that an adjacency matrix is not a good representation for this graph. words.4 contains 3264 words. The matrix would contain over 10 million entries, almost all of which are 0 (or infinity) for no edge.

A much better representation for the graph is to maintain a HashMap, where the key is the word and the value is the list of words it can transition to. For example, the graph containing the words: `car, can, cat, ran` would be

```
car  :  can  ->  cat
can  :  car  ->  cat  ->  ran
cat  :  car  ->  can
ran  :  can
```

This says that `car` may be transformed into either `can` or `cat`.
`can` may be transformed into `car, cat,` or `ran`.

Once you have built the graph, perform an appropriate search on it to find a ladder. Note that you are searching a graph, not a tree. As such, you must be careful not to follow cycles, or your program may run forever. A simple solution to the cycles problem is to mark each word when it is visited. If your program comes back to that word again, prune the branch and continue.

Test your project by finding word ladders between
a. spin to yarn
b. plays to games
c. cod to elk
d. wrinkle to bedbugs

Extras for Experts
It is generally not necessary to build the entire graph, as you will only search a small portion of it. You can try expanding the graph as you go. That is, search for the neighbors of `cat` only when you come to the word `cat` and need to process it.

Grading Rubric:

Requirement	Full credit	Partial credit
Implement the `WordLadder` Class using BFS (40 points)	Your program implements the the `WordLadder` Class as described.	Your program implements the class, but with some errors.
Format code (follow Style Guidelines) and output as specified (10 points)	Your output is formatted as specified, including proper spacing, spelling, and so on.	You did not follow some or all of the requirements for output.

What to Submit:
Electronically submit the zipped-up IntelliJ project file `WordLadder`. Do not use any packages.
Good software engineering is expected. Use lots of comments, appropriate indentation, etc. when writing your program.
Sample Run 1:
Enter the beginning word
hit
Enter the ending word
mat
hit -> hat -> mat

Sample Run 2:
Enter the beginning word
hunt
Enter the ending word
note
hunt -> hunk -> dunk -> dune -> done -> dote -> note

Sample Run 3:
Enter the beginning word
hunts
Enter the ending word
tents
hunts -> hints -> tints -> tents

Sample Run 4:
Enter the beginning word
table
Enter the ending word
graph
No word ladder found

Here is a possible main() tester method

```
public class WordLadder {

    public static void main(String[] args) throws Throwable {
        String start, end;        // the words on which the ladder is based
        Scanner in = new Scanner(System.in);

        // the words in the file, WordNode is class to represent nodes in our graph
        HashMap<String, WordNode> wordlist = new HashMap<String, WordNode>();

        // Read in the two words
        System.out.println("Enter the beginning word");
        start = in.next();

        System.out.println("Enter the ending word");
        end = in.next();

        // Check length of the words
        int length = start.length();
        if (length != end.length()) {
            System.err.println("ERROR! Words not of the same length.");
            System.exit(1);
        }

        // Read in the appropriate file of words based on the length of start
        readFile(wordlist, start);

        // Search the graph
        breadthFirstSearch(wordlist, start, end);
    }

    // Method definitions here…

}
```