CS302

Assignment 2: Word Ladder

Description

This program will demonstrate the true meaning behind the phrase *words can hurt.* A word ladder from a $startWord$ to an $endWord$ is a list of words $startWord \to w_1 \to w_2 \to w_3 \to ... \to endWord$ such that

- Every adjacent pair of words differ by a single letter
- Each word $startWord$, $endWord$, and each word w_i $1 \leq i \leq k$ is in a dictionary of given words

You may need to implement a few functions but you are required to implement the following function

- `bool wordLadder(string word, string wordFinal, const vector<string>& words, vector<string>& ladder)` - this will be a recursive function, word is the current word that is being processed in the word ladder (current word you're on in the word ladder), wordFinal is the word you are trying to reach, words is your dictionary of words, and ladder contains all the words that are currently in the word ladder. This will require recursive backtracking, i.e. when you're on a current word you need to try all of its adjacent words one at a time (one function call for each), and based on what is returned (`true`/`false`), you decide what to do next...

Contents of main

In main, you prompt for an input file that contains the dictionary. Then you read in two words (the start and end of the word ladder), you can assume both words are in the dictionary. If a solution exists, output the word ladder, otherwise output the solution does not exist. Make sure the output matches exactly with code grade (in terms of upper/lower cased letters)

Specifications

- Comment your code and functions
- You must implement the recursive backtracking function as defined
- Other functions are ok to implement as needed
- No other data structure besides an array or vector is allowed

Sample Run

You can see the sample output in code grade, again the input file given with this program with the list of words will be read in via command line arguments

Written Portion

In some word processor answer the following

1. Explain what would happen if we didn't maintain the ladder vector in the recursive function.

2. Does this guarantee the optimal word ladder for a given input? Why or why not?
3. When are you guaranteed to obtain an optimal solution? (If the optimal solution isn't always guaranteed)

Submission

Submit the source file and write up to code grade by the deadline

References

- Link to the top image can be found at $http : //pngimg.com/uploads/hello/hello_PNG1.png$