CS 0445 Spring 2022
Recitation Exercise 6

Goal:

Recursive backtracking algorithms are tricky and can sometimes be difficult to understand for those don't have a lot of experience with recursive programming. The goal of this exercise is for you to fully understand the FindWord backtracking algorithm discussed in lecture – how it is designed and how it works and how its execution progresses.

Introduction:

In Lecture 13 we introduced the idea of backtracking and in Lecture 15 we discussed some backtracking algorithms and looked at their implementations. One example discussed was the FindWord.java program, which searched for words within a 2-D grid of letters. Before completing this exercise, look over the FindWord.java handout and run it to see how it works. If you wish, uncomment the println() statement at the beginning of the findWord() method call to better see how the recursion and backtracking are working as it runs. Also review the discussion of FindWord.java in Lecture 15.

Details:

The FindWord.java program as given will search a grid of characters in 4 directions: right, down, left and up. In this exercise you will add the diagonal directions to the recursive search so that the algorithm will now consider a total of 8 directions: right, down, left, up, right down, right up, left down and left up. This addition will clearly allow more valid words to be found in a given board. This in fact is not a difficult addition to the method. If you look at the findWord() method as given, and you note the 4 recursive calls that are made, it should be fairly clear how to add the 4 additional calls needed for this version of the algorithm.

Once you have made the additions to the findWord() method compile and run the program. Use the file findWord1.txt as your input file, or make up an input file yourself. Try it with several words to see whether they are found or not. Compare the output to the original FindWord.java handout (prior to your modifications). Make sure you understand how these additional calls impact the behavior of the algorithm and how many cases are considered.