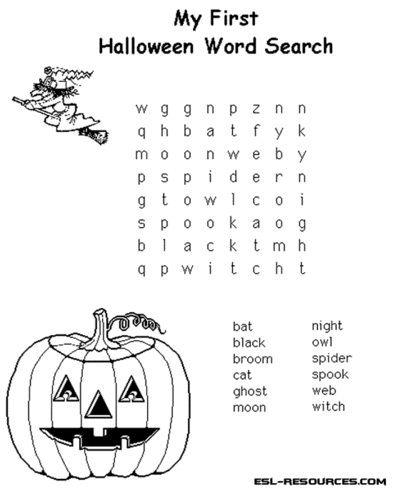
Zearn Developer Candidate Programming Project

Background
A word search is a puzzle consisting of letters arranged in a grid, containing several hidden words written in any direction. The "player" is given a list of words to find inside the puzzle. Here is an example of a word search:



Problem
We have a number of potential word search puzzles already made. Each puzzle is stored in a CSV file and has a corresponding word list stored in a TXT file (with each word on a separate line). The problem is that some of the word list files contain some words that aren't in the corresponding puzzle! This is a big problem because the "players" would be looking for these extra words forever!

What we want you to do is to create a program that replaces these word lists with correct ones. It should take in 3 command line arguments, the puzzle CSV's filename, the word list text file's filename and the filename of an output file. It should output a new text file that contains all the words in the original text file that are actually contained within the puzzle.

Attached to these instructions, you should find some example input files:

- *puzzle13.csv*
- *wordlist13.csv*

After running your program (example in ruby: `ruby wordsearch.rb puzzle13.csv wordlist13.csv output13.txt`) you would get an output like the attached *output13.txt*

Assumptions / Requirements

- Feel free to use whatever programming language you want. Just be sure to tell us how to build/run your program.
- The puzzle CSVs will always contain the same number of characters per row (always full rectangles for the puzzle).
- Only capital letters are used.
- A single character in the puzzle could be part of multiple hidden words.
- The puzzle could be any dimensions. The word list could be any size length. There could be any number of words from the word list hidden in the puzzle.
- Words may appear in the puzzle horizontally, vertically or diagonally
- Words may also appear in the puzzle written 'backwards' as well as forwards (CAT may appear as TAC)
- If words appear multiple times in a puzzle, your program should be able to tell us how many times it appears. For example, if the puzzle includes the word "cat" 3 times, the output file should say "cat (3)"