Assignment 10: Recursion: WordBrain Solver

EC602 Design by Software

Fall 2017

Contents

1 Introduction

This assignment is a natural situation for recursion to be used.

1.1 Due Date

This assignment is due 2016-12-15 at midnight.

1.2 Submission

You can submit here: **wordbrainsolver** submit link

1.3   Group Size

For this assignment, the maximum group size is 4.

1.4   Points

This assignment is worth 20 points, 10 points for each of the C++ and python parts.

1.5   About `wordbrain`

The smartphone app *wordbrain* is a puzzle game. The game presents users with a grid of letters, ranging in size 2x2 up to 5x5 and even larger.

Here is an example:

The user is given the pattern of word lengths. For this puzzle, the first word to be solved for is 7 letters, then the second word is 4 letters, and the final word is 5 letters.

The words must be spelled with adjacent letters, and as correct words are formed, the remaining letters drop down.

The game play is easier to understand if you have the app, so please download it and try it.

2   The assignment: wordbrainsolver

Your program will be given the filenames of two word lists, via the command-line, like this:

`wordbrainsolver firstwordlist secondwordlist`

or

`python wordbrainsolver.py firstwordlist secondwordlist`

The `firstwordlist` is a file containing a small word list, and the `secondwordlist` is a file containing a more comprehensive word list.

The reason for two word lists is that the puzzle answer is more likely to use basic words than exotic / rare words, so the result of the small word list will be much faster, and if it finds an answer, usually the correct answer.

Your program will read from standard input a description of a puzzle to solve, in the following format:

```
nchn
iaaw
bpom
atsn
**** ***** *** ****
```

The first lines represent the word grid, and the line with asterisks represent the words to solved for. [Any letters specified in this last line are already known.]

Your program will then print out all possible solutions to the puzzle using the first word list as the set of possible words.

If there are none, then all possible solutions using the second word list will be printed.

The program should output a period '.' on a new line at the conclusion of each puzzle solution, whether the puzzle had a solution or not.

The output must be printed in sorted order. For this puzzle, using the provided word lists, the following should be printed:

```
snow panic man bath
.
```

Your program must be called `wordbrainsolver.cpp` or `wordbrainsolver.py`

2.1 Example

The result of running

```
wordbrainsolver small_word_list.txt large_word_list.txt <puzzles.txt >solutions.txt
```

or

```
python wordbrainsolver.py small_word_list.txt large_word_list.txt <puzzles.txt >solutions.t
```

when `puzzles.txt` is

```
hee
oqr
sua
*** ******
yson
elnn
hnca
olab
***** ***** ******
nchn
iaaw
bpom
atsn
```

```
**** ***** *** ****
vanmo
ipveo
toarr
tsmed
miipb
**** ******* ******* *******
vanmo
ipveo
toarr
tsmed
miipb
p*** ******* ******* *******
yson
elnn
hnca
olab
***** holly ******
```

is that the output is redirected to the file `solutions.txt` and that output should
be

```
hoe square
.
banes holly cannon
hones bally cannon
honey balls cannon
.
snow panic man bath
.
opts bedroom vampire vitamin
post bedroom vampire vitamin
pots bedroom vampire vitamin
stop bedroom vampire vitamin
.
post bedroom vampire vitamin
pots bedroom vampire vitamin
.
banes holly cannon
.
```

The program should exit on an empty line of input or EOF.

2.2 Extra words

The following words have been added to both word lists as they appear frequently in the game:

- tv
- keyring

2.3 Required imports

You must import sys in the following way for the python program:

```
from sys import argv
```

This will prevent errors resulting from using `sys.stdin` instead of `input()`.

2.4 Downloads

- small_word_list.txt
- large_word_list.txt
- puzzles.txt
- solutions.txt

2.5 Hints: suggested imports

One solution we have uses the following imports:

```
import numpy as np
from collections import Counter
from sys import argv
```

3 Evaluation

Your program will be graded on

- correctness 40%
- efficiency 30%
- elegance 20%
- style 10%

The details will be announced as soon as possible.