CPE 101 (Spring 2018) Project 4-Due 5/21

Word Puzzle

Group of TWO

You have more freedom in this project. Your code must pass all of the tests to get full credit. Your grade will be based on the date, which your code passes all the tests. To get 100% credit for this project, you must submit your files in PolyLearn by 11:00 PM on Monday, 5/21/18.
Monday (5/21) - 100%
Tuesday (5/22) - 80%
Wednesday (5/23) – 60%

Objectives:

- Practice on lists and strings.
- Opportunity to test your understanding of all of the different concepts:
    - Conditional logic, loops,
    - lists, and functions

Required File Header:

All students are required to have the following header comment at the top of their source files. Note that the stuff to the right of the colon in red is information for an imaginary example student. (Please put YOUR appropriate information. Also, your header comment is not expected to have red text.) All program files require this header.

```
# Project 4 – Word Puzzle
# Name: Members Name
# Instructor: S. Einakian
# Section:
```

Resources:

- Your instructor, via office hours, email, Piazza, etc.
- Free tutoring Sunday-Thursday, 7-9 PM, 14-302
- Your TA

Program Description

In this project, you will implement a program, which locates words in common word search puzzles (all puzzles are 10x10). A sample puzzle is shown below:

```
WAQHGTTWZE
CBNSZQQELS
AZXWKWIIML
LDWLFXXSAV
PONDTMVUXN
OEDSDYQPOB
LGQCKGMMIT
YCSLOACAZM
XVDMGSXCYZ
UUIUNIXFNU
```

The above puzzle contains the words (shown in bold): UNIX, CALPOLY, SLO, and CAMPUS. Words can appear in the puzzle running **up, down, forward, or backward**. You do not need to check diagonals.

Input and Output

Input:

- Your program should read in a **100 character long string** from user via standard keyboard input.
- These strings are available (at the first line) in the input files called: puzzleAndWords0, puzzleAndWords1, puzzleAndWords2
- Using this user input, you should create a 10x10 puzzle which you can work with, in order to find the words in the puzzle.
- Your program should also at the same time read in **words** from user via standard keyboard input.
- These words are available (at the second line) in the input files called: puzzleAndWords0, puzzleAndWords1, puzzleAndWords2)
- These are the words that are to be found in the puzzle by your program.

Output:

- Your program should be tested as below:

  Python3 wordFinder.py < puzzleAndWords > output0
  - You may review the following files to know what the "expected output" should look like of your program:

  output0, output1, output2

- Remember that the requirement for your project is to print the output to screen, not to a file.
- Output files mentioned above are ONLY provided to you for comparison purposes. So, you can compare your own output (**prints to the screen**) with the "expected output".
- This comparison will help you verify whether or not your program is working as expected.
- You may access the **input** as well as **output** files (that are mentioned in steps above) via command given below or through PolyLearn:
- While being present in the location where you would like to copy, run the following command:

  Copy the zip file from PolyLearn to your working directory of Project4.

  unzip Project4.zip

Sample output of program (print to the screen, not to a file) is shown as below:

```
Puzzle:

WAQHGTTWEE
CBMIVQQELS
AZXWKWIIIL
LDWLFXPIPV
PONDTMVAMN
OEDSOYQGOB
LGQCKGMMCT
YCSLOACUZM
XVDMGSXCYZ
UUIUNIXFNU

UNIX: (FORWARD) row: 9 column: 3
CALPOLY: (DOWN) row: 1 column: 0
GCC: word not found
SLO: (FORWARD) row: 7 column: 2
COMPILE: (UP) row: 6 column: 8
VIM: (BACKWARD) row: 1 column: 4
TEST: word not found
```

Submission

- **Use the same name for your files**
- You must have total of three files in your submission:
    1. **funcs.py** → This file must include the actual functions definitions/implementation.

    2. **funcsTests.py** → This file must include at least two unit tests for each function, which contain the assert statements testing the functions developed in funcs.py.

    3. **wordFinder.py** → This file must include the main function, which calls all the functions developed within funcs.py.

    4. Readme.txt → This file must explain responsibilities of each member

NOTE:
Some of the suggested functions are given in a file funcs.py. You can create as many functions as you need. You can even ignore the suggested functions. But, you are not allow to write the whole project in One Function!