# Data Structures Homework #2 Due: Sept. 21 (Sat. at 11:59 PM)

REUVC unscrabbles to ['CURVE']

TOHOP unscrabbles to ['PHOTO']

FNELEN unscrabbles to ['FENNEL']

EEESDC unscrabbles to ['SECEDE']

XYZ could not be unscrabbled!

OTP unscrabbles to ['OPT', 'POT', 'TOP']

RVOSTE unscrabbles to ['STOVER', 'STROVE', 'TROVES', 'VOTERS']

You are to write a program to help solve “JUMBLE” puzzles. A “JUMBLE” puzzle is a series of scrambled words which needs to be unscrambled to real words. For example, the follow is series of 6 scrambled words in a file named jumble.txt and the corresponding expected output file named solution.txt.

REUVC

TOHOP

FNELEN

EEESDC

XYZ

OTP

RVOSTE

The file hw2/dictionary.txt contains an alphabetical list of “all” real words that JUMBLE words can unscramble to. **A couple hints about how I** want you to solve this problem reasonably efficiently:

* sort the letters in each word to get a unique string for comparisons. For example FNELEN would become EEFLNN, and FENNEL would become a matching EEFLNN
* Before reading the JUMBLE words, preprocess the dictionary.txt file to create a Python dictionary whose keys are the sorted letters of words in the dictionary file with corresponding values being the list of “matching” dictionary words. For example in this dictionary will be entrys: 'EEFLNN':['FENNEL'], 'OPT':['OPT', 'POT', 'TOP'], 'EORSTV':['STOVER', 'STROVE', 'TROVES', 'VOTERS'], and 'AEKNSS':['SKEANS', 'SNAKES', 'SNEAKS'].

**Your program can assume an input file named** jumble.txt containing one jumble word per line, and it should generate an output file named solution.txt formatted similar to the above example.

**When you write your program, be sure you:**

* think about the functional-decomposition (top-down) design **before you start to write code**! You will need to turn in a design document **design.doc** (or design.pdf, or design.txt, or design.rtf). This document describes the design of your program including a **functional-decomposition diagram showing parameters and returned values**, and text describing each function (see lab 1 description and program lab1/diceOutcome.py for examples)
* use meaningful variable names with good style (i.e., useCamelCase)
* use comments (""" Multi-line Comment """) at the start of the program **and** immediately after each function definition describing what they do (see lab 1 diceOutcomes.py program for an example)
* use a main function (see lab1 diceOutcomes.py program) located at the top of program with a call to it at the bottom of the file to start execution
* do not use global variables, except for global constants where appropriate with good style (ALL\_CAPS\_AND\_UNDERSCORES). Put your global constants after your initial comments describing the program and before your main function definition so they can be found and changed easily in future versions of your program. (see DIE\_SIDES in lab 1 diceOutcomes.py program for an example)

**On eLearning (Course Content | Unit #1 | Homework #2 subfolder) , submit a single .zip file, hw2.zip containing the following:**

* **jumbleSolver.py** (your Python program)
* **dictionary.txt** (my supplied dictionary file)
* **design.docx** (or hand-written and scanned as design.pdf, or photo design.jpg, or design.rtf) a document describing the design of your program including a **functional-decomposition diagram showing parameters and returned values**, and text describing each function (see lab 1 description)

(If you miss the deadline, you can still submit it without a late penalty. However, I there will be a homework 3, etc. and you don’t want to get too far behind!)