CS214 Homework 2

Due: March 8th, 2pm. Submit solutions to CMS, `http://cms.csuglab.cornell.edu` .
Problem 3: SCRABBLE. 40 points. Turn in `scrabble.py` .

Introduction

This assignment should get you comfortable with data structures and file handling in Python.

Problem 3: SCRABBLE

In the famous word game SCRABBLE®, players take turns placing some number of character tiles on a 15×15 board to form new words. The tiles must be placed all in the same row, or in the same column (and not both) so that they all belong to the same word, and at least one of the new tiles is adjacent to another that is already on the table. The new tiles can only be played if all horizontal and vertical sequences they connect to form permissible words, as governed by a special SCRABBLE dictionary. Valid words are always spelled left-to-right or up-to-down, never backwards. After putting down the tiles, players draw new tiles from a bag until they have they have 7 tiles, or until the bag is empty. Note that a single character word is always valid, and that the connectedness requirement makes the game hard. There is a face value on every tile, for instance the letter '*A*' gives you 1 point, whereas '*Q*' gives you 10 points. Your score is the aggregate point value of all new words that formed with your play. Special squares on the board multiply your score. The goal of the game is to have the highest score. If you haven't played the game, please read up on Wikipedia or try the Scrabulous application on Facebook.

This is the highest scoring game of SCRABBLE that has been recorded in North America in a competition.

Task

In this assignment we will write a move legality evaluator for SCRABBLE in Python. Given a particular board configuration and proposed placement of new tiles, you need to verify whether the move is legal, in the sense that all tiles have been placed according to the rules, and that all new words are permissible according to the dictionary. We will ignore scores and blank tiles altogether.

Your program should be called *scrabble.py* and accept two arguments: a board configuration file, and the next move file, respectively.

File formats

A *board configuration file* consists of 15 lines with 15 characters each. Each character corresponds to the letter occupying that square, and we use '.' to denote an empty tile. For example, the board corresponding to the game above looks like:

```
G..BLEEP......Q
I..A........C.U
BUNT........O.I
E..E...J....RAX
.UNDERDOG.NO.NO
OP.....U.ZA...T
.......SCAMSTER
F......T......Y
LADYLIKE.......
AW....ADO......
TA...ES.MITE..V
F..........HAIR
I..........I.O
S..........VROW
HEIR..........S
```

The upper left corner has the coordinate A1 and the lower right is O15.

The *new move file* contains somewhere between 1 and 7 lines, each of which has two columns separated by a tab. The first column is a coordinate (like K10), and the second is a character (such as *Q*). For example,

```
F2	N
F3	T
F4	E
```

would produce the word '*ENTER*' vertically on the board above (the initial '*E*' and final '*R*' are already in place). The moves have no particular order, so the F4 move could precede e.g. F2.

We will use the "*SOWPODS*" SCRABBLE dictionary, the standard for international tournaments. It is simply a list of valid words, one word per line. Please download *sowpods.txt* from *http://www.scrabulous.com/sowpods_dictionary.php*. This file is in DOS format, so be careful as to remove carriage returns (use e.g. `col < sowpods.txt > fixed.txt`). Your script can assume this file exists in the current directory.

Output

The output of your script should be one of the following.

- "BAD START: *word*" if some word formed on the initial board configuration is not permissible. Your program should print the invalid word it detected, and then stop (if there are multiple possibilities, return any).
- "BAD MOVE" if the new move characters are not all in the same row or column, or if they belong to different word parts (i.e. there are some empty squares between the tiles being placed). You must also check whether previously occupied squares are being overwritten, and whether the new sequence connects with previous tiles on the board (i.e. some new tile must be adjacent to one that is already in place).
- "BAD WORD: *word*" if you detect an invalid word after placing the new tiles. You should print the word, and halt execution. For instance, N11 *M* and N13 *S* on the board above produces the valid words 'ISO' and 'MISO' but 'MV' is not a legal word.
- "VALID" otherwise, that is if this is a legal move that produces permissible SCRABBLE words. Given the two input files above, your script should say "VALID".

Criteria

Your solution will be checked automatically, so please make sure you don't print debugging information to `stdout` . Test your code and be unafraid to ask questions if you run into problems.

Along with correctness, the grade for this assignment will depend on your design (modularity), the readability of your code (helpful comments, explanation of data structures, intuitive variable names, program logic, etc.), and efficiency. Write short routines that do a clear, designated task. Think before you write.

Hint: You can reuse the routine to verify legality of the initial board to check the validity of the new words in the end. Check all horizontal words separately from vertical ones.

Hint: If you store the board in a two-dimensional array, it may pay off to use *sentinels*. A sentinel in this case is an extra layer of empty squares around the board (making it 17×17) that serves to get rid of the "*is this square out of bounds?*" checks when you are identifying a word, since you would never continue matching a word once a blank square has been found.

Hint: Use a *dict* to look up words in the dictionary (hence the name...)