Programming assignment 6: Scrabble in terminal

Handing in the assignment: Students hand in a single archive (**ZIP**/**RAR**/**7Z**) containing all their code (*.py*) and **storage files** on *Canvas*. Make sure all files are placed so that the program runs correctly when your main python file is run directly, and that the python file that should be run is named descriptively.
This assignment is **not** tested with automatic tests but run by humans and is mainly judged on working functionality. Points may nonetheless be deducted for:

- redundant or unnecessarily repeated code,
- messy code,
- code that is difficult to understand and not explained in comments.

Scrabble is a board game where players compete with each other to earn the highest point total. Players earn points by placing letters together so they make words. The players then earn points based on the letters used for that word.

Here are some good resources for those who are not familiar with the game:
https://www.youtube.com/watch?v=K1KgvZwwJqo
https://en.wikipedia.org/wiki/Scrabble

This assignment is different from previous ones in that there is not a specific input and output. The programs will be tested by running them and trying them, rather than running them against scripted tests. This means students have more freedom in the design and implementation of their programs. Making a game like this look good in a text-based terminal view can be tricky. It is OK if the game board is reprinted each time and the previous board just scrolls up in the terminal (no points for clearing the screen, although students can experiment with that if they wish) but students need to find nice ways of taking input from the user, i.e. where to start a word, is it vertical or horizontal, etc.

Students can use any built-in data structures and functionality in Python but if it is necessary to install packages that are imported in the project this needs to be stated in a README.txt file in the code folder.

Descriptions and weights of functionality on next page...

You can implement the following parts in any order you see fit but it is recommended to finish the basic game before proceeding to the advanced. Each part is graded on both **functionality** and **program structure**, but mainly on *functionality*.

Basic game

- 5% Two players enter their names before the game starts
- 5% Define a full bag of letters and map the letters to the correct values
- 5% Display which letters players have access to
- 5% Display which players turn it is
- 10% Display a 15x15 grid
- 10% Players can choose three options on their turn.
    - Play a word with letters they have drawn
        - Players can add words vertically and horizontally on the grid
    - Swap any number of letters with the same number of letters from the bag
    - Pass their turn
- 10% Display the words played on the grid
- 10% Calculate score for a word played

Advanced game

- 5% When a player plays a word using all 7 of their letters that player gets an extra 50 points added to their total score
- 5% Add modifiers to certain tiles on the grid
    - For example:
        - Double letter
        - Triple word
        - See wikipedia link for the modifier setup on the grid
- 5% Letter modifiers get calculated into players score when word is created on these tiles
- 5% Word modifiers get calculated into players score when word is created on these tiles
- 10% Detect when the game should end
    - The game ends when all letters have been drawn from the bag and one player has finished all their letters
    - Or when all players pass twice in a row
- 10% Allow more than 2 players to play (official scrabble rules state it can be played with 2-4 people)
- 10% Checking words (see assignment page for a dictionary file)
    - Check the word played with a dictionary
        - If the word played is invalid the current player forfeits their turn
    - Check all words connected to the played word with a dictionary
        - If any words connected to the played word are invalid the current player forfeits their turn