Word Search (due 07 Sep 2020)

Given an *m* by *n* grid of letters, and a list of words, find the location in the grid where the word can be found. A word matches a straight, contiguous line of letters in the grid. The match could either be done horizontally (left or right) or vertically (up or down). It is not guaranteed that every word in the grid has the same orientation and direction. Moreover the word may not exist in the grid.

INPUT: You will read your input from a file called *word.in*.

- First line will have two integers - m, the number of lines in the grid and n, the number of characters in each line. The integers *m* and *n* will be separated by one or more spaces.
- There will be a single blank line.
- There will be *m* lines, where each line will have *n* characters, all in upper case, separated by a space.
- There will be a single blank line.
- There will be a single integer *k*, denoting the number of words that follow.
- There will be *k* lines. Each line will contain a single word in all uppercase.

For this assignment, we have provided a *template* for you to read the input file. After having read the input file you will get two variables *word_grid* and *word_list*. These variables are described here and in HackerRank:

- *word_grid* - a 2D list (matrix) of uppercase letters, of size *m* x *n* where *m* > 0 and *n* > 0. This represents the grid of characters that you will be searching in.
- *word_list* - a 1D list of size *k* where *k* > 0. These are the words you will search for. Words will be given in uppercase.

OUTPUT: You will print your output as shown in this file *word.out*. There will be *k* lines in your output. Each line will have the x and y coordinates of the **first** letter of the word separated by a space. Use zero based indexing, with the top left corner being 0, 0. If the word does not exist in the grid then print *-1 -1*.

Extra Credit (10 points): Some of the words that we will ask you to search do exist in the grid. But those words run along a diagonal either from left to right or from right to left. If you find those words then you can output the indices of the first letter of those words like you did before.

For this assignment you may work with a partner. Both of you must read the paper on Pair Programming and abide by the ground rules as stated in that paper. If you are working with a partner then only one of you will submit the code. Make sure that in the header in HackerRank that you have your name and UT EID and your partner's name and UT EID. If you are working alone then you will just have your name and your UT EID.

We have provided the following file Word.py as a template. You may **NOT** change the names of the functions but you may add as many helper functions as needed. You will follow the standard coding conventions in Python. There are two reasons for providing this template:

1. To help you get an idea on how to structure your code.
2. In case you would like to run this code on your own IDE instead of HackerRank's native IDE.

Submission: Here is the HackerRank Link to the assignment. Use the *HackerRank* platform to submit your code. If you have questions about how HackerRank works, do refer to this video.

Here are important things to bear in mind:

- You have multiple chances of editing and testing your code.
- Once you submit your code, you may not edit your code any further.
- HackerRank will NOT determine if your code is late or not. We will according to the late policies as specified in the syllabus.

We should receive your work by 11 PM on Monday, 07 Sep 2020. **You need to manually submit your code by this deadline.** HackerRank will **not** automatically submit your code for you. HackerRank will also give you more time than you actually have to complete the assignment. **Follow only OUR due dates.**

Grading: Grading will be assigned based on the number of test cases you pass. We provide you 3 of our test cases on HackerRank to show you how answers should be formatted. Passing all the sample test cases **does not** guarantee you will receive an 100. We have more hidden cases that we run your code through.

Thus it is encouraged that you **thoroughly test your code by writing your own test cases**, read through this document and the HackerRank requirements, and ask questions on Piazza if anything else remains unclear. Try to think as hard as you can about potential edge cases that could cause errors in your solution.