Homework - Thanksgiving XC

Version 0.1

COS125 - Instructor: Zachary Hutchinson

Due: Friday November 24th @12AM (midnight)

Submission Instructions

These must be followed or you will lose points.

1. Submit your coded solution as a .py file. Code in any other file format will be rejected.
2. Your .py file should be named: yourlastname_hwX.py. The 'X' in the filename is replaced by the homework number.
3. Your code must have as a comment at the top of the file: your name, the homework and the names of anyone from whom you received help. This includes students in the course as well as course staff. For example, if you went to Boardman 138 and received help from an MLA, put their name on your file. If a classmate helped you debug a bit of code, put their name down. This is for your benefit.
4. If you do not manage to squash all the bugs in your program, include a comment at the top of the file detailing the outstanding bugs. Acknowledging bugs is a sign of a mature programmer. Doing so will not eliminate point deductions but it might mitigate them. It shows you care about your work.

Problem Description

In class during the lecture on testing, you considered how to test a program that determined whether a sequence of moves given by a string of characters resulted in a specified final position. In class this was done using a 10x10 2D board. Now you will implement the actual program. Although we spoke about this with respect to how to test it, you do not need to write extensive tests for the program.

Requirements

Your program will take an input string from the command line (using the sys module) and report back whether a series of moves from a starting position on the board finishes at the end position. Positions are given in X,Y coordiantes.

Format of the input string:

```
<SX><SY><EX><EY><MOVES>
```

- SX: is the starting x-coordinate given by a single decimal character.
- SY: is the starting y-coordinate given by a single decimal character.
- EX: is the ending x-coordinate given by a single decimal character.
- EY: is the ending y-coordinate given by a single decimal character.
- MOVES: is a series of moves given by the characters: U, D, L, R. See the section on **Moves** for more information.

- U: causes the program to move up on the board (y-1)
- D: causes the program to down up on the board (y+1)
- L: causes the program to left up on the board (x-1)
- R: causes the program to right up on the board (x+1)

Example of running the program with an input string that starts at 0,0 and ends (I believe) at 1,1:

```
python hwxc.py 0011D9R5R4U8LLLLLLLL
```

It moves down 9 spaces. Right 5 space. Right again 4 spaces. Up 8 spaces. Finally, left 8 spaces.

Coordinates

Coordinates are given by a single decimal digit. An input string is required to have start and end coordinates.

Examples:

- 0099 would mean that the starting location is 0,0 (top left of the board) and the ending location is 9,9 (bottom right).
- 4253 places the start at 4,2 and the end at 5,3.

Moves

MOVES can consist of any number of moves. There is no limit. *There can be no moves,* in which case the input string consists of only a start and end.

Each move can be followed by a single numeric digit. This indicates the number of times the previous move should be repeated. For example: `D5` would mean from the current position, the program should move five spaces down on the board. This allows movement to be repeated up to 9 times using a single direction command. If a direction is not followed by a number, it is assumed to be 1. Zero cannot follow a direction, only 1-9.

A movement command which would move the program off the board causes the program to "wrap" around to the opposite side. For example, if the program was at 1,9 (the bottom of the board) and it encountered D as its next instruction, it would move to 1,0. Similarly, if it was at 0,5 and it encountered an L2, it would end at 8,5, having moved two spaces to the left.

Output

The program should output the start, end and final x,y coordinates. If final matches the end, the program should also alert the user that they match or not with either OK (for match) or KO (for no match).

Example (X,Y is replaced with the actual values):

```
Start: X,Y
End:   X,Y
```

```
    Final: X,Y
    OK or KO
```

Hints

- Modulus
- The program can move into and out of the end location. It only counts as a match with the end location if after the series of moves the program is at the end location. If it does not finish there it doesn't count.