HW#1 --- last modified Friday, 01-Sep-2017 15:31:46 PDT.

Solution set.

Due date: Sep 18

Files to be submitted:
Hw1.zip

Purpose: To gain experience with problem solving agents and the $A^\star$ algorithm

Related Course Outcomes:

The main course outcomes covered by this assignment are:

LO1 -- By code or by hand find solution nodes in a state space using the $A^\star$ algorithm.

Specification:

Nario is an upwardly mobile video game character. When presented a n x m, text-based maze, Nario always strives to climb to the top floor using the A*-algorithm. Below is an example of the kind of maze Nario might appear in:

```
..........
=.======.=
..=.......
====.=====
........=.
=.=====.==
=.........
==.=======
.....@....
```

The meaning of this Nario maze is as follows:

- In the above Nario, is represented by the @.
- An obstacle is represented by =
- All other squares in the n x m board have a '.' symbol.
- In one turn, Nario can move one square to the left, right or up of his current location provided there are no obstacles in the square he is going to move to.
- Nario can move off the edge of the board to the left or right and will wrap-around to the other side of the board, again, provided there is no obstacle blocking this move.
- Input boards will always place Nario somewhere on the bottom row of the board.

For this homework you will write a Python program to get Nario to the top of the maze. Here are some guidelines for your program:

- Your program will be run from the command line with a line like:

  ```
  python nario.py file_name heuristic
  ```

  Here *file_name* is the name of some text file with a Nario board. *heuristic* can be one of the three values *manhattan*, *euclidean*, *my_own*.

- Your program should use the A* algorithm to determine a sequence of moves to take Nario from the bottom row of the board to the top row. If the input board was:

```
...
==.
@..
```

  your output should look like:

```
...
==.
.@.

...
==.
..@

...
==@
...

..@
==.
...
```

- If there is no path to the top row, your program should output, NO PATH.
- Your A* algorithm should use the heuristic specified by the command-line argument. I.e., if *manhattan* is specified, then the Manhattan distance of Nario from his current position to the top row, ignoring obstacles should be used.
- If the *my_own* heuristic is chosen, then the A* algorithm should use a heuristic of your own invention, different from Manhattan or Euclidean distance.
- Your program should draw using text characters a sequence of intermediate boards to take Nario from the bottom row to the top row.
- Your Python code should conform to the Pep8 coding guidelines and should work if Python 2.7.* is being used.

Your program will

Point Breakdown

PEP 8 coding guidelines followed.	1pt
Code is split into reasonable function, classes, etc and is well commented.	1pt
Program reads in input mazes correctly.	1pt
Can find in Python code an implementation of $A^\star$ search.	1pt
Can find in Python code an implementations of each of the three mentioned heuristics (1pt each).	3pts
Output corresponds to spec above and should exactly match on examples above.	1pt
Program works on all my tests cases where there is a solution.	1pt
Program works on all my tests cases where there is a no solution.	1pt
Total	10pts