Name: _____
Obtain copies of files

- puzz8_midterm.py
- astar_midterm.py

You will add one function to the code in these file.

TH-1: ***Complete function puzz8_eval_fct_B (_,_) in file puzz8_midterm.py*****.** Given a puzzle state and a goal state, this function is to compute the sum of the horizontal and vertical displacements for all tiles in the puzzle state relative to the goal state.

Test your function puzz8_eval_fct_B() by calling it for provided puzzles `puzzD`, `puzzE`, and `challenge.`

Submit: (1) A hardcopy of your completed file puzz8_midterm.py, and (2) a typescript of your testing the new function for the three puzzle states (do show_puzz8 to display each states, then output the evaluation function value).

TH-2: ***Complete function `extract_path(_,_)` in file astar_midterm.py.*** This function takes two arguments, the last state from `open` (which matches the goal state) and list of nodes `closed.` Its purpose is to extract the path between the starting state and the goal state from the nodes stored on `closed.`

When you examine the provided implementation of A* closely, you will notice that the paths from start to a node ('Node') are no longer stored explicitly in a data member of Node. However, 'Node' has been supplied with a new data member, Node.theparent. For the Node that contains the starting state, .theparent is None. For each other Node object, (call it X), .theparent is set to the Node object (call it P) from which X was computed as one of the successors of P. In this manner, each Node object contains information about its own state and also has a reference to the parent Node from which it was generated. I.e., computing the successors of P produced Node X (and possible others); Node X will remember its "parent" by setting its .parent to P.

All nodes on closed thus come with information about their "parent" … Think about this and what it will take to reconstruct the path from the starting state to any such node on closed. Program function `extract_path()` to do the reconstructing …

The implementation of function extract_path() is all you need to do to arrive at a running program that will iterate over all 8-puzzle examples in puzz8_midterm.py, and run A* first with the original evaluation function (#tiles-out-of-place), and then with the evaluation function from TH-1 (sum of horizontal and vertical displacements). At the end of all runs, you will see some simple statistics displayed. You should find that the new evaluation function outperforms #tiles-out-of-place.

Submit: (1) A hardcopy of your completed file astar_midterm.py, and (2) a typescript of your running the new variant of A* with both evaluation functions.

Submit the requested files and script files for TH-1 and TH-2 even if your code is running perfectly. You should minimally be able to load the files into the Python interpreter. Any run-time errors that you should appear in the typescript, and they will help grading and earning of partial credit.

Do not submit any other implementations of the A* algorithm as they are not tuned to work with the completion of just two functions. Other than completing the functions for TH-1 and TH-2, do not make any modifications to the provided code (as given, the code should also work for Python 3 users … except for print statements, and … yes, you may modify these; be as conservative as possible).