Part 1 The first part of this assignment is to compare several heuristics for solving the 8-puzzle using A* search. You do not have to implement A* for this. You can use existing code. The class web page contains a pointer to A* code for the 8-puzzle written in C++ using the STL. There is other code available on the web. (Of course, you are welcome to write your own.) But you will probably need to write the code for at least some of the heuristics you test.

Test the following three heuristics:

Number of tiles out of place

Manhattan distance: This is the sum of the (horizontal and vertical) distances of each tile from its goal position.

Nilsson's sequence: \( h(n) = M(n) + 3S(n) \) where \( M(n) \) is the Manhattan distance heuristic and \( S(n) \) is a “sequence score” computed as follows: for each noncentral square, add two if it is not followed by its proper successor, otherwise add zero. (This assumes the goal state has a blank in the middle. Also, this heuristic is not admissible.)

Hand in the following for part 1:

1) A copy of the A* code used for your experiments. Indicate the source of your code and describe any changes you made to it.
2) For at least 30 instances of the 8-puzzle, plot the average number of nodes expanded as a function of solution length for each heuristic.
3) At least a half-page write-up describing what you learned from these experiments.

Part 2: The second part of the assignment is implement the IDA* algorithm and use it to solve the 8 puzzle. You must write the code for the IDA*algorithm yourself. However, you can use code for the 8-puzzle game (the board representation, function for computing successor nodes, etc.) from other sources, including the code used for the experiments of part 1.

Hand in the following for part 2:

1) A documented listing of the code you wrote to implement IDA*.
2) For at least 30 instances of the 8-puzzle, plot the average number of nodes expanded during the last iteration of IDA* as a function of solution length, using the Manhattan distance heuristic.
3) At least a half-page write-up describing what you learned from implementing IDA*.

Node ordering: Your implementation of IDA* will run much faster if you expand the children of each node in increasing order of the heuristic estimate. This technique is called node ordering. Because IDA* stops as soon as it finds a solution, this can cause it to visit many fewer nodes during its last iteration. Although this will not make earlier iterations any faster, most of the running time of IDA* is spent in the last iteration.

Extra credit:

1) You will get up to 10 points extra credit if you invent your own heuristic for the 8-puzzle, give a reasonable justification for it, and compare it to the other three heuristics tested in part 1.
2) You will get up to 20 points extra credit if your implementation of IDA* also solves the 15-puzzle. The 15-puzzle is much more difficult to solve than the 8-puzzle, but IDA* can do it if implemented efficiently. (By contrast, A* usually runs out of memory on most examples of the 15-puzzle.)