1. **Eight and fifteen puzzle [60 pts]**
   In this question you will compare the effectiveness of various search methods in solving the eight and fifteen puzzles.

   Write a program that solves the 8 puzzle using each of the following search methods:
   
   - Breadth-first search;
   - Depth-first search;
   - Iterative deepening search;
   - Bidirectional search;
   - A* with the following heuristics:
     - # of mispalced tiles ($h_1$);
     - Sum of the Manhattan distances of each tile from its correct position ($h_2$);
     - A nonadmissible heuristic of your design ($h_3$). Describe your heuristic and explain why it’s nonadmissible.

   In writing your program use the `search.py` module which is provided on the book website. Your module, named `eight_puzzle.py` will contain a class `EightPuzzle` which is a subclass of `search.Problem`. For testing purposes, include in your module a function called `eight_puzzle_search` that receives as input the following two parameters:

   - `start_state`: a python list of length 9 providing the initial state, where element `i` is either a number between 1 and 8, or the symbol '.', denoting the blank square. In this notation the goal state can be represented as `['.', 1, 2, 3, 4, 5, 6, 7, 8]`.
   - A search method: a string which is one of the following: 'BFS', 'DFS', 'IDS', 'BID', 'h1', 'h2', 'h3'. The latter three define the heuristic to use in conjunction with A* search.

   The function returns a list of actions that when applied to the initial state lead to the goal state. Each action is encoded as a single number between 1 and 9, signifying moving the tile at that position to the position of the blank tile.

   After you write your program, compare the performance of the different search methods on 300 random instances of the 8 puzzle. The methods should be compared in term of the number of nodes expanded during the search. Note that not all 8 puzzle instances are solvable, so in order to generate solvable random instances, start with the goal state and apply random moves to it (say 100). This analysis should be submitted as part of the written part of the assignment.
Extra Credit [10 pts] Make your program work for the 15 puzzle and make a similar analysis as for the 8 puzzle. Which search methods can handle this problem? Explain!

2. Bidirectional heuristic search [20 pts]
In class we discussed uninformed bidirectional search. Bidirectional search can be combined with heuristic search. Ira Pohl presented an algorithm for doing so in his Ph.D. thesis in 1969. Dennis De Champeaux and Lenie Sint describe the shortcoming of Ira Pohl’s version and proposed an improved way of incorporating heuristic functions in a 1977 paper (link on the assignments page). What issue in Pohl’s algorithm is their algorithm trying to address? How do they address this issue?

3. Weighted evaluation functions [20 pts]
Consider a best-first search algorithm with the following evaluation function:

\[ f(n) = (2 - w)g(n) + wh(n) \]

where \( w \) is a real-valued parameter, \( g(n) \) is the cost of getting to the node \( n \), and \( h(n) \) is an admissible heuristic function. For what values of \( w \) is this algorithm guaranteed to be optimal? What kind of search does this perform when \( w = 0 \)? When \( w = 1 \)? When \( w = 2 \)?

Submission Prepare a module called eight_puzzle.py and submit it via checkin. The answers to the written part are to be submitted in class.