ICS 361 Assignment 2: State Space Representations and Search (100 pts)

Due: Monday, October 5

Your task here is to implement general-purpose (i.e. not puzzle-specific) versions of the breadth-first and depth-first search strategies, and apply them to three different puzzles.

Part A (20 points)

Implement versions of both **breadth-first** and **depth-first** search strategies, using *open* and *closed* lists. Your search code must be **independent** of the application problem (i.e. you should use the same search code for all three problems in Part B, although you may use a depth limit of twenty on the 8-puzzle problem). You are welcome to start with existing code you find online, but you *must* cite your source, indicate clearly which code is yours and which is from someone else, and write all of the comments in your own words.

Add printing functions to your searches so that the first node on the open list is printed in each iteration. Also add printing functions so that when the goal is reached, the length of the open and closed lists as well as the solution path is printed.

This example (from Nancy Reed) shows depth-, breadth-, and best-first (not on this assignment) searches applied to the Missionaries and Cannibals river-crossing problem. Nodes (*problem independent*) are represented by a 3-tuple consisting of a state, the node's parent state, and the depth of the node in the search tree. States (*problem specific*) are represented by a 3-tuple with the number of Ms then number of Cs on the start side and the 3rd part of the state shows the side where the boat is currently located. Both Ms and Cs can paddle the boat. The maximum capacity of the boat is two and the minimum is one (at least one M or C must be in the boat to paddle). The goal is to move all Ms and Cs to the opposite side while making sure that Cs do not outnumber Ms on either side of the river. If that should happen, Ms would be eaten (not a legal state).

Part B (60 points)

Create state space representations of the following problems for use with the search strategies you implemented above:

- (15 pt) Farmer, Wolf, Goat and Cabbage (http://www.mathsisfun.com/puzzles/farmer-crosses-river.html)
- (15 pt) Water Jugs (http://www.math.tamu.edu/~dallen/hollywood/diehard/diehard.htm)
- (30 pt) 8 puzzle (http://www.tilepuzzles.com/default.asp?p=12)

FWGC and Water Jugs puzzles have fixed beginning and ending states. The 8-puzzle, however can start in any configuration, and has two commonly-defined goal states (but any state could be defined as the goal state).

To solve any 8-puzzle problem, you must be able to easily specify the start and goal states (e.g. read them in from a file, load lisp definitions, or enter them from the console). For your transcript, use the following start and goal states:

Start:

```
-------
|2|8|3|
-------
|1|6|4|
-------
|7| |5|
-------
```

Goal:

```
-------
|1|2|3|
-------
|8| |4|
-------
|7|6|5|
-------
```

Create meaningful comments in all files with a description of the problem and descriptions of each of the variables and functions used in the file. Load your code in to Lisp and save the output of solving each of the puzzles [UHUnixName]2OUT.txt (add A, B, C if more than one file).

Note: you will use the code from this assignment for the next assignment.

Part C (20 points)

For each puzzle:

- Describe a successful strategy for solving the puzzle, in clearly written English (a paragraph or two for each puzzle).
- How many different states are possible in the search space for each puzzle?

Turn in the following (all via Laulima):

- All code, appropriately commented.
- Transcripts of the solutions for each puzzle.
- A text file with the answers to Part C.