CSC461 Programming Languages – Spring 2015
Programming Assignment #2: Functional Programming in Lisp
Missionaries and Cannibals Puzzle

Three missionaries and three cannibals find themselves on one bank of a piranha-infested river with a canoe that holds only two persons. How can they safely cross the river in such a way that the number of missionaries on either bank is never less than the number of cannibals? Write a program in Lisp to help everyone cross the river safely.

This classic AI problem assumes exactly three missionaries and three cannibals, but your program will allow the user to specify any number of missionaries and cannibals. The program is run by typing:

```
(m-c m c)
```

where *m* is the number of missionaries and *c* is the number of cannibals. Note that not all problem instances are solvable. It should be possible to run this program more than once in a Lisp session.

Program output is a nicely formatted set of statements describing the current state (how many cannibals and missionaries are on each bank, and where the canoe is) and the last move made. For consistency, let us assume that the goal is to move everyone from the left bank to the right bank. Here is sample output for *(m-c 3 3).* You do not have to use exactly the same output format.

```
3 Missionaries and 3 Cannibals:

left bank       right bank      canoe       last move
---------       ----------      -----       ---------
3 M, 3 C        0 M, 0 C        left        start position
2 M, 2 C        1 M, 1 C        right       move 1 M, 1 C left to right
    .               .             .             .
    .               .             .             .
    .               .             .             .
```

The high-level symbolic approach to problem-solving in AI is known as the state space approach, and involves graph search. In this approach, successor states are generated from the start state. One or more of these successors are selected for exploration in the next iteration, new successors are generated, and eventually a solution path is determined from the start state to the goal state in the state space graph.

A variety of search strategies have been developed to explore the state space. Exhaustive search strategies eventually explore all possible successor states en route to finding a solution path. Depth-first search (DFS) is an exhaustive search technique that is most easily described recursively:

```
if ( current_state == goal_state )
      return SUCCESS;

for each unexplored successor of the current state
      if ( DFS( successor ) == SUCCESS )
            return SUCCESS;

return FAILURE;
```

DFS explores potential solution paths as deeply as possible, until it reaches a goal (success), or hits a dead end (no more unexplored successor nodes) and must backtrack. Recursion is an elegant way to handle the backtracking. Once a goal state is reached, the path can be traced back to the start state to recover the sequence of solution steps.

Notes

- Write a Lisp program that uses the state space approach to solve the generalized version of the Missionaries and Cannibals Problem. Code a recursive DFS function to implement the search strategy. You must use this approach to receive full credit on the assignment.
- Work in teams of two students on this program, with the same team assignments as HW#3.
- When you are finished writing, testing, and debugging your program, submit your source code using the *Submit It!* link on the MCS Department Website. Usage is self-explanatory: enter your name, choose the instructor (Weiss) and click "Select Instructor", choose the appropriate course (CSC461), browse to the filename you wish to submit, and click "Upload". Multi-file programs should be packaged in a zip or tar archive for submission. Teams should make one joint submission, not individual submissions for each team member.
- To receive full credit, your code must be readable, modular, nicely formatted, and adequately documented, as well as complete and correct. It must build and run successfully under the current CLisp interpreter (version 2.49, available on the course website) under Windows and Linux. If your program does not run correctly, indicate why. This will make it easier to give you partial credit.
- Submit your program by the due date (Monday April 27) in order to receive credit for this assignment. Late programs will not be accepted for partial credit unless prior arrangements have been made with the instructor. If you have any problems with the submit program, report them to your instructor and submit your program by email instead.