CSC461 Programming Languages – Fall 2016
Programming Assignment #4: Functional Programming in Lisp
Farmer, Wolf, Goat, Cabbage Puzzle

A farmer with his wolf, goat, and cabbage arrive at the bank of a river. A boat at the river's edge is only large enough for the farmer and one of his possessions. The farmer cannot leave the wolf alone with the goat, or the goat alone with the cabbage. Write a program in Lisp to help the farmer cross the river with all of his possessions intact.

Initiate the search by typing (FWGC) at the Lisp prompt. Output each state in the solution path in a nicely-formatted sequence, similar to this:

```
> (fwgc)

  Left Bank        Right Bank       Action
  ---------        ----------       ------
  (F W G C)        -                *start state*
  (W C)            (F G)            farmer takes goat across
  (F W C)          (G)              farmer returns alone
  ...              ...              ...
  (F G)            (W C)            farmer returns alone
  -                (F W G C)        farmer takes goat across
                                    *** problem solved! ***
```

This is a classic Artificial Intelligence (AI) problem. 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 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 in different ways. 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 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 Farmer, Wolf, Goat, and Cabbage Problem. Code a recursive DFS function to implement the search strategy. You must use this approach to receive full credit on the assignment.
- 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.
- 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.
- Submit your program by midnight on the due date (Wednesday November 30) 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 by email, and submit your program via email attachment.
- You must work alone on this assignment. Please do not share code with your classmates.