Programming Challenge: M&C Interactive Problem Solver

This document presents a programming assignment that pertains to the classic Missionaries and Cannibals problem in recreational mathematics. This assignment affords you an opportunity to do just a little bit of list processing as you refine some given code. Moreover, this assignment paves the way for subsequent programming assignments that focus on this problem.

The Problem

Three missionaries and three cannibals, along with one boat that fits at most two people (and requires at least one person for operation), are on the left bank of a river. The most salient thing about missionaries and cannibals in "cohabitation" is that if ever the cannibals in any one spot (left bank, right bank, on the boat) outnumber the missionaries, the outnumbered missionaries will be consumed – eaten! The goal of this problem is to get all six individuals safely across the river from the left bank to the right bank.

Solution?

Can you solve the problem? Give it a try. Feel free to draw a picture of the situation and use the picture in your thinking. You will need to be able to solve the problem in order to demo your program, so you might as well solve it now.

Tasks

1. **Refine** the accompanying program. That means that you must refrain from changing my code. Rather, simply add code that my code references. As you do this:
   - Maintain precisely three global variables, `*left-bank*` and `*right-bank*`, which collectively represent the state of the world, and `*move-list*` which is used to record the history of moves made en route to the goal.
   - Initialize the state of the world in the `establish-world` procedure. Initialize the move history in the `init-move-list` procedure.
2. When your program is ready, type `( mc )` and demonstrate its workings. Be sure to illustrate **inapplicability**, **feastivity**, and **success** in a sample session.

Please do bear in mind that it is required that your program be consistent with both the code that I have supplied and the accompanying sample session.

Code To Be Refined

```
( defun mc ()
  ( establish-world )
  ( init-move-list )
  ( make-moves )
)

( defun make-moves ()
  ( display-world )
  ( cond
    ( ( goalp )
      ( write-line "Good work!" )
      nil
    )
    ( ( feast-state-p )
      ( write-line "Yummy yummy yummy, I got Good in my tummy!!" )
      nil
    )
    ( t
      ( let ( m )
        ( format t ">>> " ) ( setf m ( read ) )
        ( if ( applicable-p m )
             ( let () ( perform-move  m ) ( make-moves ) )
             ( let () ( write-line "Move inapplicable" ) nil )
        )
      )
    )
  )
)

( defun perform-move ( move )
  ( setf *move-list* ( snoc move *move-list* ) )
  ( if ( equal ( current-bank ) *left-bank* )
    ( move-lr move )
    ( move-rl move )
  )
)

( defun move-lr ( ml )
  ( if ( null ml ) ( return-from move-lr ) )
  ( move-lr-1 ( first ml ) )
  ( move-lr ( rest ml ) )
)

( defun move-rl ( ml )
  ( if ( null ml ) ( return-from move-rl ) )
  ( move-rl-1 ( first ml ) )
```

```
  ( move-rl ( rest ml ) )
)
```

Redacted Demo

```
[] ( mc )
*left-bank*     (M M M C C C B)
*right-bank*    NIL
>>> ( m b )
*left-bank*     (M M C C C)
*right-bank*    (B M)
Yummy yummy yummy, I got Good in my tummy!!
NIL

[] ( mc )
*left-bank*     (M M M C C C B)
*right-bank*    NIL
>>> ( m c b )
*left-bank*     (M M C C)
*right-bank*    (B C M)
>>> ( c c b )
Move inapplicable
NIL

[] ( mc )
*left-bank*     (M M M C C C B)
*right-bank*    NIL
>>> ( c m b )
*left-bank*     (M M C C)
*right-bank*    (B C M)
>>> ...
*left-bank*     (C B M M M C)
*right-bank*    (C)
...

*left-bank*     NIL
*right-bank*    (B C C C M M M)
good work!
"good work!"

[] ( display-solution )
( c m b )
...
```

Post Your Work

Reference your solution document for this assignment from your AI Work Site. Your document should have a nice title, a short learning abstract, a section for your demo, and a section for your

source code.

Due Date

Friday, September 23, 2022