Programming Challenge: Missionaries and Cannibals SSPS

This programming challenge involves the coding of a state space problem solver for the classic Missionaries and Cannibals problem.

Why do it?

This programming challenge will:

1. Further acquaint you with the object-oriented features of Lisp realized in **CLOS**.
2. Afford you an opportunity to write a **state space problem solver** for the classic "missionaries and cannibals" problem.
3. Provide you with an experience in performing simple **breadth first search**.
4. Engage you a **stepwise refinement** exercise in Lisp programming.

Tasks

1. Establish a preliminary **solution document** for this assignment, consisting of a title, a short learning abstract, and four sections: the first section to house the source code; the secton section for a "quiet" demo; the third section in which you will place two things, an "expand/explore" node demo and a graphic representing the search tree created en route to the solution; the fourth section in which to place an "unexplored list / explored list" demo.
2. **Working by analogy** with the state space problem solving program presented in class for the water jug problem, **refine** the accompanying state space problem solving code (redacted code) which forms the basis of a breadth first search for a solution to the problem. **Add the source code to the first section of your solution document.**
3. Run the program once without tracing anything, so that the only output will be the solution. **Place the demo in the second section of your solution document.**
4. Run the program tracing just the e-node. Then, draw the state space tree corresponding to the output of your program. Color feast states red, previous states brown, and the initial state, the goal state, and the path from the initial state to the goal state blue. **Add both the demo and the graphic to the third section of your solution document.**
5. Run the program once more with "ux" as the trace indicator, in order to trace the unexplored list and the explored list. **Save the demo to the fourth section of your solution document.**
6. Post your solution document to your web work site!

Notes

1. The constraint that you **refine** the given code means that you must use it as is, without alteration. The idea is that you simply add code to what is given so that it all works together to solve the problem.
2. The Water Jug program conflated the notions of state and node. The two concepts are separated in the Missionaries and Cannibals problem. The latter design is conceptually nicer!

Given, redacted code for you to work from

```
;--------------------------------------------------------------------------------
; GENERAL INFORMATION
;
; FILE:  mc_ssps.l
; DATE:  Fall 2020
; LINE:  State Space Solver for the Missionaries and Cannibals Problem
;--------------------------------------------------------------------------------

;--------------------------------------------------------------------------------
; PROGRAM DESCRIPTION
;
; This program is a state space problem solver for a classic missionaries and
; cannibls problem.  An explicit state space tree is grown in concert with breadth
; first search for a solution.
;--------------------------------------------------------------------------------

;--------------------------------------------------------------------------------
; REPRESENTATIONAL NOTES
;
; Banks are represented as a 3-slot class consisting of
;   missionaries, cannibals, and boat.
;
; States are represented as a 2-slot class consisting of
;   left-bank and right-bank.
;
; Operators are represented as a 3-slot class consisting of
;   a name, a precondition, and a description.
;
; Nodes are represented as a 4-slot class consisting of
;   a name, a state, a parent node, and a move (state space operator)
;--------------------------------------------------------------------------------

;--------------------------------------------------------------------------------
; MODELING A BANK

  ( defclass bank ()
    (
      ( missionaries :accessor bank-missionaries :initarg :missionaries )
      ( cannibals :accessor bank-cannibals :initarg :cannibals )
      ( boat :accessor bank-boat :initarg :boat )
    )
  )

;------------------------------------------------------------------
; MODELLING A STATE

  ( defclass state ()
    (
      ( left-bank :accessor state-left-bank :initarg :left-bank )
      ( right-bank :accessor state-right-bank :initarg :right-bank )
    )
  )

  ( defmethod display ( ( s state ) )
    ( display ( state-left-bank s ) )
    ( display ( state-right-bank s ) )
    nil
  )

;------------------------------------------------------------------
; MODELLING A NODE

  ( defclass node ()
    (
      ( name :accessor node-name :initarg :name )
      ( state :accessor node-state :initarg :state )
      ( parent :accessor node-parent :initarg :parent )
      ( operator :accessor node-operator :initarg :operator )
    )
  )

  ( defmethod display ( ( n node ) )
    ( format t "~A " ( node-name n ) )
    ( if ( not ( rootp n ) )
      ( let ()
        ( format t "~A " ( node-name ( node-parent n ) ) )
        ( display ( node-operator n ) )
      )
    )
    ( terpri )
    ( display ( node-state n ) )
    nil
  )

;------------------------------------------------------------------------------
; MODELING A STATE SPACE OPERATOR

  ( defclass operator ()
    (
      ( name :accessor operator-name :initarg :name )
      ( precondition :accessor operator-precondition :initarg :precondition )
      ( description :accessor operator-description :initarg :description )
    )
  )
```

```
;--------------------------------------------------------------------------------
; THE MAIN PROGRAM - argument values of e u x eu ex ux eux will cause tracing

    ( defmethod mc ( ( trace-instruction symbol ) )
      ( setf *trace-instruction* trace-instruction )
      ( establish-operators )
      ( setup )
      ( solve )
    )

;--------------------------------------------------------------------------------
; SOLVE PERFORMS BREADTH FIRST SEARCH

  ( defmethod solve ( &aux kids e-node )
    ( if ( member *trace-instruction* '(u eu ux eux) ) ( display-the-unexplored-list ) )
    ( if ( member *trace-instruction* '(x ex ux eux) ) ( display-the-explored-list ) )
    ( cond
      ( ( null *unexplored* )
        ( format t ">>> THERE IS NO SOLUTION.~%" )
        ( return-from solve NIL )
      )
    )
    ( setf e-node ( pop *unexplored* ) )
    ( if ( member *trace-instruction* '(e ex eu eux) ) ( display-the-e-node e-node ) )
    ( cond
      ( ( goalp ( node-state e-node ) )
        ( format t "~%>>> GOT A SOLUTION!" )
        ( display-solution e-node )
      )
      ( ( feast-state-p ( node-state e-node ) )
        ( solve )
      )
      ( ( exploredp ( node-state e-node ) )
        ( solve )
      )
      ( t
        ( push e-node *explored* )
        ( setf kids ( children-of e-node ) )
        ( setf *unexplored* ( append *unexplored* kids ) )
        ( solve )
      )
    )
    nil
  )

  ( defmethod exploredp ( ( s state ) )
    ### best to use member with two key word args -- :key and :test
  )

;------------------------------------------------------------------
; THE SETUP

  ( defmethod setup ( &aux root lb rb istate )
    ;; establish root node
    ( setf lb ( make-instance 'bank :missionaries '(m m m) :cannibals '(c c c) :boat 'b ) )
    ( setf rb ( make-instance 'bank :missionaries '() :cannibals '() :boat nil ) )
    ( setf istate ( make-instance 'state :left-bank lb :right-bank rb ) )
    ( setf root ( make-instance 'node :state istate :name 'root ) )
    ;; initialize list of unexplored nodes
    ( setf *unexplored* ( list root ) )
    ;; initialize list of explored nodes
    ( setf *explored* () )
    ; get ready to create good names
    ( setf *ng* ( make-instance 'name-generator :prefix "N" ) )
  )

;------------------------------------------------------------------
; GENERATING CHILDREN

  ( defmethod children-of  ( (e-node node) &aux kids )
    ###
  )

  ( defmethod child-of ( (  n node ) ( o operator ) &aux c )
    ( setf new-node ( make-instance 'node ) )
    ( setf ( node-name new-node ) ( next *ng* ) )
    ( setf ( node-parent new-node ) n )
    ( setf ( node-operator new-node ) o )
    ( setf c ( copy-state ( node-state n ) ) )
    ( apply-operator o c )
    ( setf ( node-state new-node ) c )
    new-node
  )

;------------------------------------------------------------------
; MODELLING A NAME-GENERATOR

  ( defclass name-generator ()
    ( ( prefix :accessor name-generator-prefix :initarg :prefix :initform "name" )
      ( nr :accessor name-generator-nr :initform 0 )
    )
  )

  ( defmethod next ( ( ng name-generator ) )
    ( setf ( name-generator-nr ng ) ( + 1 ( name-generator-nr ng ) ) )
    ( concatenate 'string
      ( name-generator-prefix ng )
      ( write-to-string ( name-generator-nr ng ) )
    )
  )
```

Suggestive Demo

```
bash-3.2$ clisp
...

[1]> ( load "mc_ssps.l" )
;; Loading file mc_ssps.l ...
;; Loaded file mc_ssps.l
T
[2]> ( mc 'eux )

>>> UNEXPLORED LIST
ROOT
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL

>>> EXPLORED LIST

>>> E-NODE
ROOT
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL

>>> UNEXPLORED LIST
N5 ROOT MOVE-CM-LR
MISSIONARIES=(M M) CANNIBALS=(C C) BOAT=NIL
MISSIONARIES=(M) CANNIBALS=(C) BOAT=B
N4 ROOT MOVE-CC-LR
MISSIONARIES=(M M M) CANNIBALS=(C) BOAT=NIL
MISSIONARIES=NIL CANNIBALS=(C C) BOAT=B
N3 ROOT MOVE-MM-LR
MISSIONARIES=(M) CANNIBALS=(C C C) BOAT=NIL
MISSIONARIES=(M M) CANNIBALS=NIL BOAT=B
N2 ROOT MOVE-C-LR
MISSIONARIES=(M M M) CANNIBALS=(C C) BOAT=NIL
MISSIONARIES=NIL CANNIBALS=(C) BOAT=B
N1 ROOT MOVE-M-LR
MISSIONARIES=(M M) CANNIBALS=(C C C) BOAT=NIL
MISSIONARIES=(M) CANNIBALS=NIL BOAT=B

>>> EXPLORED LIST
ROOT
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL

>>> E-NODE
N5 ROOT MOVE-CM-LR
MISSIONARIES=(M M) CANNIBALS=(C C) BOAT=NIL
MISSIONARIES=(M) CANNIBALS=(C) BOAT=B

>>> UNEXPLORED LIST
N4 ROOT MOVE-CC-LR
MISSIONARIES=(M M M) CANNIBALS=(C) BOAT=NIL
```

```
MISSIONARIES=NIL CANNIBALS=(C C) BOAT=B
N3 ROOT MOVE-MM-LR
MISSIONARIES=(M) CANNIBALS=(C C C) BOAT=NIL
MISSIONARIES=(M M) CANNIBALS=NIL BOAT=B
N2 ROOT MOVE-C-LR
MISSIONARIES=(M M M) CANNIBALS=(C C) BOAT=NIL
MISSIONARIES=NIL CANNIBALS=(C) BOAT=B
N1 ROOT MOVE-M-LR
MISSIONARIES=(M M) CANNIBALS=(C C C) BOAT=NIL
MISSIONARIES=(M) CANNIBALS=NIL BOAT=B
N8 N5 MOVE-CM-RL
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL
N7 N5 MOVE-C-RL
MISSIONARIES=(M M) CANNIBALS=(C C C) BOAT=B
MISSIONARIES=(M) CANNIBALS=NIL BOAT=NIL
N6 N5 MOVE-M-RL
MISSIONARIES=(M M M) CANNIBALS=(C C) BOAT=B
MISSIONARIES=NIL CANNIBALS=(C) BOAT=NIL

>>> EXPLORED LIST
N5 ROOT MOVE-CM-LR
MISSIONARIES=(M M) CANNIBALS=(C C) BOAT=NIL
MISSIONARIES=(M) CANNIBALS=(C) BOAT=B
ROOT
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL

###
###
###

>>> UNEXPLORED LIST
N45 N40 MOVE-CC-LR
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B
N44 N40 MOVE-C-LR
MISSIONARIES=NIL CANNIBALS=(C) BOAT=NIL
MISSIONARIES=(M M M) CANNIBALS=(C C) BOAT=B
N48 N39 MOVE-CM-LR
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B
N47 N39 MOVE-C-LR
MISSIONARIES=(M) CANNIBALS=NIL BOAT=NIL
MISSIONARIES=(M M) CANNIBALS=(C C C) BOAT=B
N46 N39 MOVE-M-LR
MISSIONARIES=NIL CANNIBALS=(C) BOAT=NIL
MISSIONARIES=(M M M) CANNIBALS=(C C) BOAT=B

>>> EXPLORED LIST
N39 N38 MOVE-M-RL
MISSIONARIES=(M) CANNIBALS=(C) BOAT=B
MISSIONARIES=(M M) CANNIBALS=(C C) BOAT=NIL
N40 N38 MOVE-C-RL
```

```
MISSIONARIES=NIL CANNIBALS=(C C) BOAT=B
MISSIONARIES=(M M M) CANNIBALS=(C) BOAT=NIL
N38 N34 MOVE-CC-LR
MISSIONARIES=NIL CANNIBALS=(C) BOAT=NIL
MISSIONARIES=(M M M) CANNIBALS=(C C) BOAT=B
N34 N30 MOVE-C-RL
MISSIONARIES=NIL CANNIBALS=(C C C) BOAT=B
MISSIONARIES=(M M M) CANNIBALS=NIL BOAT=NIL
N30 N27 MOVE-MM-LR
MISSIONARIES=NIL CANNIBALS=(C C) BOAT=NIL
MISSIONARIES=(M M M) CANNIBALS=(C) BOAT=B
N27 N21 MOVE-CM-RL
MISSIONARIES=(M M) CANNIBALS=(C C) BOAT=B
MISSIONARIES=(M) CANNIBALS=(C) BOAT=NIL
N21 N17 MOVE-MM-LR
MISSIONARIES=(M) CANNIBALS=(C) BOAT=NIL
MISSIONARIES=(M M) CANNIBALS=(C C) BOAT=B
N17 N15 MOVE-C-RL
MISSIONARIES=(M M M) CANNIBALS=(C) BOAT=B
MISSIONARIES=NIL CANNIBALS=(C C) BOAT=NIL
N15 N6 MOVE-CC-LR
MISSIONARIES=(M M M) CANNIBALS=NIL BOAT=NIL
MISSIONARIES=NIL CANNIBALS=(C C C) BOAT=B
N6 N5 MOVE-M-RL
MISSIONARIES=(M M M) CANNIBALS=(C C) BOAT=B
MISSIONARIES=NIL CANNIBALS=(C) BOAT=NIL
N2 ROOT MOVE-C-LR
MISSIONARIES=(M M M) CANNIBALS=(C C) BOAT=NIL
MISSIONARIES=NIL CANNIBALS=(C) BOAT=B
N4 ROOT MOVE-CC-LR
MISSIONARIES=(M M M) CANNIBALS=(C) BOAT=NIL
MISSIONARIES=NIL CANNIBALS=(C C) BOAT=B
N5 ROOT MOVE-CM-LR
MISSIONARIES=(M M) CANNIBALS=(C C) BOAT=NIL
MISSIONARIES=(M) CANNIBALS=(C) BOAT=B
ROOT
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL

>>> E-NODE
N45 N40 MOVE-CC-LR
MISSIONARIES=NIL CANNIBALS=NIL BOAT=NIL
MISSIONARIES=(M M M) CANNIBALS=(C C C) BOAT=B

>>> GOT A SOLUTION!
Move (C M B) from left bank to right bank.
Move (M B) from right bank to left bank.
###
Move (C C B) from left bank to right bank.
NIL
[3]>
```

Due Date:

Wednesday, November 9, 2022