CSC 416

Programming Challenge 4

Jacob Peck

A program in LISP to allow the user to attempt to interactively solve the Missionaries and Cannibals problem. Checks for feast-state, success, and move inapplicability.

Listing of `mc.l`:

```
; mc.l - Missionaries and Cannibals I

; runs the program
(defun mc ()
  (establish-world)
  (init-move-list)
  (make-moves)
)

; takes input moves, tests for applicibilty, and also for exit cases
(defun make-moves ()
  (display-world)
  (cond
    ((goalp)
      (write-line "good work!")
    )
    ((feast-state-p)
      (write-line "yummy yummy yummy, I got Good in my tummy!!")
    )
    (t
      (let ((m (read)))
        (if (applicable-p m)
          (let () (perform-move m) (make-moves))
          (let () (write-line "move inapplicable"))
        )
      )
    )
  )
)

; performs the move
(defun perform-move (move)
  (if (equal (current-bank) *left-bank*)
    (move-lr move)
    (move-rl move)
  )
  (setf *move-list* (cons move *move-list*))
)

; move from left to right
(defun move-lr (ml)
  (if (null ml) (return-from move-lr))
  (move-lr-1 (first ml))
  (move-lr (rest ml))
)

; helper function, moves 1 from left bank to right
(defun move-lr-1 (s)
  (setf *right-bank* (cons s *right-bank*))
  (setf *left-bank* (remove s *left-bank* :count 1))
)

; move from right to left
```

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

; helper function, moves 1 from right bank to left
(defun move-rl-1 (s)
  (setf *left-bank* (cons s *left-bank*))
  (setf *right-bank* (remove s *right-bank* :count 1))
)

; initialize the left and right banks
(defun establish-world ()
  (setf *left-bank* '(c c c m m m b))
  (setf *right-bank* '())
)

; initalize the move list
(defun init-move-list ()
  (setf *move-list* '())
)

; test for goal condition
(defun goalp ()
  (null *left-bank*)
)

; return the current bank
(defun current-bank ()
  (cond
    ((evenp (length *move-list*))
      *left-bank*
    )
    (t
      *right-bank*
    )
  )
)

; tests for move validity
(defun applicable-p (m)
  (cond
    ((and (<= (count 'c m) (count 'c (current-bank)))
          (<= (count 'm m) (count 'm (current-bank)))
          (member 'b m)
          (<= (length m) 3))
      t
    )
    (t
      nil
    )
  )
)
```

```
; tests for a feast state (more cannibals than missionaries on either side)
(defun feast-state-p ()
  (cond
    ((or (and (> (count 'c *left-bank*) (count 'm *left-bank*))
              (> (count 'm *left-bank*) 0))
         (and (> (count 'c *right-bank*) (count 'm *right-bank*))
              (> (count 'm *right-bank*) 0)))
      t
    )
    (t
      nil
    )
  )
)

; prints the solution
(defun display-solution ()
  (display-solution-1 *move-list*)
)

; helper function, prints one item of the move list
(defun display-solution-1 (l)
  (cond
    ((null l)
      t
    )
    (t
      (format t "~A" (car l)) (terpri)
      (display-solution-1 (cdr l))
    )
  )
)

; prints the current state of the world
(defun display-world ()
  (format t "*left-bank*     ~A" *left-bank*) (terpri)
  (format t "*right-bank*    ~A" *right-bank*) (terpri)
)
```

Listing of `mc-session.text`:

```
$ clisp

<...snip...>

[1]> (load "mc.l")
;; Loading file mc.l ...
;; Loaded file mc.l
T
[2]> (mc)
*left-bank*     (C C C M M M B)
*right-bank*    NIL
(m b)
*left-bank*     (C C C M M)
*right-bank*    (B M)
yummy yummy yummy, I got Good in my tummy!!
"yummy yummy yummy, I got Good in my tummy!!"
[3]> (mc)
*left-bank*     (C C C M M M B)
*right-bank*    NIL
(m c b)
*left-bank*     (C C M M)
*right-bank*    (B C M)
(c c b)
move inapplicable
"move inapplicable"
[4]> (mc)
*left-bank*     (C C C M M M B)
*right-bank*    NIL
(c c b)
*left-bank*     (C M M M)
*right-bank*    (B C C)
(c b)
*left-bank*     (B C C M M M)
*right-bank*    (C)
(c c b)
*left-bank*     (M M M)
*right-bank*    (B C C C)
(c b)
*left-bank*     (B C M M M)
*right-bank*    (C C)
(m m b)
*left-bank*     (C M)
*right-bank*    (B M M C C)
(c m b)
*left-bank*     (B M C C M)
*right-bank*    (M C)
(m m b)
*left-bank*     (C C)
*right-bank*    (B M M M C)
(c b)
*left-bank*     (B C C C)
*right-bank*    (M M M)
```

```
(c c b)
*left-bank*     (C)
*right-bank*    (B C C M M M)
(c b)
*left-bank*     (B C C)
*right-bank*    (C M M M)
(c c b)
*left-bank*     NIL
*right-bank*    (B C C C M M M)
good work!
"good work!"
[5]> (display-solution)
(C C B)
(C B)
(C C B)
(C B)
(M M B)
(C M B)
(M M B)
(C B)
(C C B)
(C B)
(C C B)
T
[6]> (bye)
Bye.
```