Kuncheng Feng

AI Assignment: Practice with the Basics, Evaluators, and Property

This assignment features a basic Lisp interaction exercise, two function definitions involving numeric computation, two function definitions that make good use of evaluators, and a property list representation for pieces in the game of Quarto.

Task 1: Simple Things

1). PRELIMINARY TASK: Start a Lisp process.

2). DISTANCE-BETWEEN-TWO-POINTS

```
[]> (setf x1 1)
1
[]> (setf y1 1)
1
[]> (setf x2 5)
5
[]> (setf y2 4)
4
[]> (setf point1 (list x1 y1))
(1 1)
[]> (setf point2 (list x2 y2))
(5 4)
[]> (setf diff-x (- (first point2) (first point1)))
4
[]> (setf diff-y (- (second point2) (second point1)))
3
[]> (setf distance (sqrt (+ (expt diff-x 2) (expt diff-y 2))))
5
[]> distance
5
```

3). REFERENCING X

(a) (A B X C D)

```
[]> (setf 3a '(a b x c d))
(A B X C D)
[]> (car (cdr (cdr 3a)))
X
```

(b) (A B (X) C D)

```
[]> (setf 3b '(a b (x) c d))
(A B (X) C D)
[]> (car (car (cdr (cdr 3b))))
X
```

(c) ((A (B X C D)))

```
[]> (setf 3c '((a (b x c d))))
((A (B X C D)))
[]> (car (cdr (car (cdr (car 3c)))))
X
```

4). A FEW META-FORMS

```
[]> (setf colors '(red orange yellow green blue indigo violet))
(RED ORANGE YELLOW GREEN BLUE INDIGO VIOLET)
[]> (quote colors)
COLORS
[]> 'colors
COLORS
[]> colors
(RED ORANGE YELLOW GREEN BLUE INDIGO VIOLET)
[]> (describe 'colors)

COLORS is the symbol COLORS, lies in #<PACKAGE COMMON-LISP-USER>, is
accessible in 1
package COMMON-LISP-USER, a variable, value:
(RED ORANGE YELLOW GREEN BLUE INDIGO VIOLET).

 #<PACKAGE COMMON-LISP-USER> is the package named COMMON-LISP-USER.
It has 2 nicknames
CL-USER, USER.
 It imports the external symbols of 2 packages COMMON-LISP, EXT and
exports no symbols,
but no package uses these exports.

(RED ORANGE YELLOW GREEN BLUE INDIGO VIOLET) is a list of length 7.
```

```
[]> (describe colors)

(RED ORANGE YELLOW GREEN BLUE INDIGO VIOLET) is a list of length 7.

[]> (type-of 'colors)
SYMBOL
[]> (type-of colors)
CONS
[]> (typep colors 'cons)
T
[]> (typep colors 'list)
T
[]> (typep colors 'symbol)
NIL
[]> (typep 'colors 'cons)
NIL
[]> (typep 'colors 'symbol)
T
```

5). B5 DECK

```
[]> (setf clubs '((10 c) (j c) (q c) (k c) (a c)))
((10 C) (J C) (Q C) (K C) (A C))
[]> clubs
((10 C) (J C) (Q C) (K C) (A C))
[]> (setf diamonds '((10 d) (j d) (q d) (k d) (a d)))
((10 D) (J D) (Q D) (K D) (A D))
[]> diamonds
((10 D) (J D) (Q D) (K D) (A D))
[]> (setf hearts '((10 h) (j h) (q h) (k h) (a h)))
((10 H) (J H) (Q H) (K H) (A H))
[]> hearts
((10 H) (J H) (Q H) (K H) (A H))
[]> (setf spades '((10 s) (j s) (q s) (k s) (a s)))
((10 S) (J S) (Q S) (K S) (A S))
[]> spades
((10 S) (J S) (Q S) (K S) (A S))
[]> (setf deck (append clubs diamonds hearts spades))
((10 C) (J C) (Q C) (K C) (A C) (10 D) (J D) (Q D) (K D) (A D) (10 H)
(J H) (Q H) (K H) (A H) (10 S) (J S) (Q S) (K S) (A S))
[]> deck
((10 C) (J C) (Q C) (K C) (A C) (10 D) (J D) (Q D) (K D) (A D) (10 H)
(J H) (Q H) (K H) (A H) (10 S) (J S) (Q S) (K S) (A S))
[]> (setf partitioned-deck (list clubs diamonds hearts spades))
(((10 C) (J C) (Q C) (K C) (A C)) ((10 D) (J D) (Q D) (K D) (A D))
((10 H) (J H) (Q H) (K H) (A H)) ((10 S) (J S) (Q S) (K S) (A S)))
```

```
[]> partitioned-deck
(((10 C) (J C) (Q C) (K C) (A C)) ((10 D) (J D) (Q D) (K D) (A D))
((10 H) (J H) (Q H) (K H) (A H)) ((10 S) (J S) (Q S) (K S) (A S)))
```

Task 2: Distance Between Two Points

Code

```
(defun distance (x1 y1 x2 y2 &aux diff-x diff-y result)
      (setf diff-x (- x2 x1))
      (setf diff-y (- y2 y1))
      (setf result (sqrt (+ (expt diff-x 2) (expt diff-y 2))))
      result)
```

Demo

```
[]> (load "perimeter_of_triangle.txt")
;; Loading file perimeter_of_triangle.txt ...
;; Loaded file perimeter_of_triangle.txt
T
[]> (distance 1 1 5 4)
5
[]> (distance 0 0 12 5)
13
[]> (distance 18 19 72 63)
69.656296
```

Task 3: Perimeter of Triangle

Code

```
(defun perimeter (x1 y1 x2 y2 x3 y3 &aux d1 d2 d3 perimeter)
      (setf d1 (distance x1 y1 x2 y2))
      (setf d2 (distance x1 y1 x3 y3))
      (setf d3 (distance x2 y2 x3 y3))
      (setf perimeter (+ d1 d2 d3))
      perimeter)
```

Demo

```
[]> (load "perimeter_of_triangle.txt")
;; Loading file perimeter_of_triangle.txt ...
;; Loaded file perimeter_of_triangle.txt
T
[]> (perimeter 0 0 4 0 4 3)
12
[]> (perimeter 0 0 12 0 12 5)
30
[]> (perimeter 2 5 6 3 5 7)
12.200792
```

Task 4: LR Infix Calculator

Code

```
(defun calculator-lr (&aux val-1 val-2 val-3 op-1 op-2 temp result)
     (format t "Expression? ")
     (setf val-1 (read))
     (setf op-1 (read))
     (setf val-2 (read))
     (setf op-2 (read))
     (setf val-3 (read))
     (setf temp (apply op-1 (list val-1 val-2)))
     (setf result (apply op-2 (list temp val-3)))
     (format t "Result: ~A~%" result)
     (calculator-lr))
```

Demo

```
[]> (calculator-lr)
Expression? 12.3 + 4.5 * 9.6
Result: 161.28
Expression? 12.3 / 4.5 - 9.6
Result: -6.866667
Expression? 8 * 125 / 25
Result: 40
Expression? 77 + 23 * 4
Result: 400
…
```

Task 5: RL Infix Calculator

Code

```
(defun calculator-rl (&aux val-1 val-2 val-3 op-1 op-2 temp result)
      (format t "Expression? ")
      (setf val-1 (read))
      (setf op-1 (read))
      (setf val-2 (read))
      (setf op-2 (read))
      (setf val-3 (read))
      (setf temp (funcall op-2 val-2 val-3))
      (setf result (funcall op-1 val-1 temp))
      (format t "Result: ~A~%" result)
      (calculator-rl))
```

Demo

```
[]> (calculator-rl)
Expression? 12.3 + 4.5 * 9.6
Result: 55.5
Expression? 12.3 / 4.5 - 9.6
Result: -2.4117646
Expression? 77 + 23 * 4
Result: 169
…
```

Task 6: Representing Quarto Pieces using Property Lists

Code

```
; File: quarto.txt
; This file holds the code for csc 416 assignment 04 task 6

; Size :- Big / Small
; Color :- Red / Blue
; Style :- Hollow / Solid
; Shape :- Circle / Square

; Big reds
(setf (symbol-plist 'brhc) '(size big color red style hollow shape
circle))
```

```
(setf (symbol-plist 'brhs) '(size big color red style hollow shape
square))
(setf (symbol-plist 'brsc) '(size big color red style solid shape
circle))
(setf (symbol-plist 'brss) '(size big color red style solid shape
square))

; Big blues
(setf (symbol-plist 'bbhc) '(size big color blue style hollow shape
circle))
(setf (symbol-plist 'bbhs) '(size big color blue style hollow shape
square))
(setf (symbol-plist 'bbsc) '(size big color blue style solid shape
circle))
(setf (symbol-plist 'bbss) '(size big color blue style solid shape
square))

; Small reds
(setf (symbol-plist 'srhc) '(size small color red style hollow shape
circle))
(setf (symbol-plist 'srhs) '(size small color red style hollow shape
square))
(setf (symbol-plist 'srsc) '(size small color red style solid shape
circle))
(setf (symbol-plist 'srss) '(size small color red style solid shape
square))

; Small blues
(setf (symbol-plist 'sbhc) '(size small color blue style hollow shape
circle))
(setf (symbol-plist 'sbhs) '(size small color blue style hollow shape
square))
(setf (symbol-plist 'sbsc) '(size small color blue style solid shape
circle))
(setf (symbol-plist 'sbss) '(size small color blue style solid shape
square))

(setf *pieces* '(brhc brhs brsc brss bbhc bbhs bbsc bbss srhc srhs
srsc srss sbhc sbhs sbsc sbss))

(defun display-random-piece (&aux number name size color style shape)
     (setf number (random (length *pieces*)))
     (setf name (nth number *pieces*))
     (setf size (get name 'size))
     (setf color (get name 'color))
```

```
      (setf style (get name 'style))
      (setf shape (get name 'shape))
      (list name size color style shape)
)
```

Demo

```
[1]> (load "quarto.txt")
;; Loading file quarto.txt ...
;; Loaded file quarto.txt
T
[2]> *pieces*
(BRHC BRHS BRSC BRSS BBHC BBHS BBSC BBSS SRHC SRHS SRSC SRSS SBHC
SBHS SBSC SBSS)
[3]> (symbol-plist 'brsc)
(SIZE BIG COLOR RED STYLE SOLID SHAPE CIRCLE)
[4]> (symbol-plist 'sbhs)
(SIZE SMALL COLOR BLUE STYLE HOLLOW SHAPE SQUARE)
[5]> (display-random-piece)
(BRHC BIG RED HOLLOW CIRCLE)
[6]> (display-random-piece)
(BRHS BIG RED HOLLOW SQUARE)
[7]> (display-random-piece)
(BBHC BIG BLUE HOLLOW CIRCLE)
[8]> (display-random-piece)
(BRSS BIG RED SOLID SQUARE)
[9]> (display-random-piece)
(SBHS SMALL BLUE HOLLOW SQUARE)
[10]> (display-random-piece)
(BBHS BIG BLUE HOLLOW SQUARE)
[11]> (bye)
Bye.
```

Task 7: Web work site

Make an “entry” for this assignment on the website