Racket Programming Assignment #3: Lambda and Basic Lisp

<><><><><><><><><><><><><><><><><><><><><><><><>

Learning Abstract: In this lab I got a better understanding of lambda functions and the other two basic list processing operations in Lisp. I also figured out how to revise my code to make it work better.

<><><><><><><><><><><><><><><><><><><><><><><><>

Task 1a:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ((lambda (x) ( cons x ( cons ( + x 1 ) (cons ( + x 2 ) '()))))5)
'(5 6 7)
> ((lambda (x) ( cons x ( cons ( + x 1 ) (cons ( + x 2 ) '()))))0)
'(0 1 2)
> ((lambda (x) ( cons x ( cons ( + x 1 ) (cons ( + x 2 ) '()))))108)
'(108 109 110)
> 
```

Task 1b:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ((lambda (a b c ) (cons c ( cons b ( cons a '() )))) 'red 'yelllow 'blue)
'(blue yelllow red)
> ((lambda (a b c ) (cons c ( cons b ( cons a '() )))) 10 20 30)
'(30 20 10)
> ((lambda (a b c ) (cons c ( cons b ( cons a '() )))) "Miss Scarlet" "Colonel Mustard" "Professor 
Plum")
'("Professor Plum" "Colonel Mustard" "Miss Scarlet")
> 
```

Task 1c:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
5
>  ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
5
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
4
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
4
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
3
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
5
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
5
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
3
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
5
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 3 5)
3
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
16
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
15
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
12
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
15
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
15
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
16
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
15
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
14
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
12
> ( ( lambda ( x y ) (random x y ) ( random x ( + y 1 ) ) ) 11 17)
15
>
```

Task 2:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( define colors '(red blue yellow orange) )
> colors
'(red blue yellow orange)
> 'colors
'colors
> (quote colors)
'colors
> ( car colors)
'red
> (cdr colors)
'(blue yellow orange)
> (car (cdr colors) )
'blue
> (cdr (cdr colors) )
'(yellow orange)
> (cadr colors)
'blue
> (cddr colors)
'(yellow orange)
> (first colors)
'red
> (second colors)
'blue
> (third colors)
'yellow
> (list-ref colors 2 )
'yellow
> (define key-of-c '(c d e) )
> (define key-of-g '(g a b) )
> (cons key-of-c key-of-g )
'((c d e) g a b)
> (list key-of-c key-of-g)
'((c d e) (g a b))
> (append key-of-c key-of-g )
'(c d e g a b)
> (define pitches '(do re mi fs so la ti) )
> (car (cdr (cdr (cdr animals) ) ) )
 animals: undefined;
 cannot reference an identifier before its definition
> (cadddr pitches)
'fs
> (list-ref pitches 3)
'fs
> (define a 'alligator)
> (define b 'pussycat)
> (define c 'chimpanzee)
> (cons a (cons b (cons c '())))
'(alligator pussycat chimpanzee)
> (list a b c)
'(alligator pussycat chimpanzee)
> (define x '(1 one))
> (define y '(2 two))
> (cons (car x) (cons (car (cdr x))y))
'(1 one 2 two)
> (append x y)
'(1 one 2 two)
>
```

Task 3a:


Untitled
(define ...)
Check Syntax
Debug
Macro Stepper
Run
Stop
#lang racket
(define (sampler)
(display "(?): ")
(define the-list (read))
(define the-element
(list-ref the-list (random (length the-list)))
)
(display the-element) (display "\n")
(sampler)
)
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (sampler)
(?): (red orange yellow green blue indigo violet)
violet
(?): (red orange yellow green blue indigo violet)
green
(?): (red orange yellow green blue indigo violet)
green
(?): (red orange yellow green blue indigo violet)
indigo
(?): (red orange yellow green blue indigo violet)
violet
(?): (red orange yellow green blue indigo violet)
yellow
(?): (aet ate eat eta tae tea)
tea
(?): (aet ate eat eta tae tea)
aet
(?): (aet ate eat eta tae tea)
ate
(?): (aet ate eat eta tae tea)
eta
(?): (aet ate eat eta tae tea)
ate
(?): (aet ate eat eta tae tea)
tea
(?): (0 1 2 3 4 5 6 7 8 9)
2
(?): (0 1 2 3 4 5 6 7 8 9)
7
(?): (0 1 2 3 4 5 6 7 8 9)
6
(?): (0 1 2 3 4 5 6 7 8 9)
6
(?): (0 1 2 3 4 5 6 7 8 9)
9
(?): (0 1 2 3 4 5 6 7 8 9)
2
(?): . . user break
read: illegal use of `.`
>

Task 3b:

Untitled (define ...) Check Syntax Debug Macro Stepper Run Stop

```
1  #lang racket
2  (require 2htdp/image)
3
4  ( define ( color_thing )
5  ( display "(?): " )
6  (define the_list (read))
7     (define condition (first the_list))
8     (define list (second the_list))
9     (define len (length list))
10    (cond
11      ((eq? condition 'random)
12       (define new (random len))
13       (draw (list-ref list new)) (display "\n")
14      )
15    ((eq? condition 'all) (all list))
16    (else (draw ( list-ref list (- condition 1 ))) (display "\n"))
17    )
18 (color_thing)
19 )
20 (define (all col)
21   (cond
22     ((eq? (length col) 1 ) (draw (car col)) (display "\n"))
23     (else
24      (draw (car col)) (display "\n")
25      (all (cdr col))
26      )))
27 (define (draw color) (display (rectangle 500 20 "solid" color)))
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (color_thing)
(?): (random (olivedrab dodgerblue indigo plum teal darkorange) )

(?): (random (olivedrab dodgerblue indigo plum teal darkorange) )

(?): (random (olivedrab dodgerblue indigo plum teal darkorange) )

(?): (all (olivedrab dodgerblue indigo plum teal darkorange) )

(?): (2 (olivedrab dodgerblue indigo plum teal darkorange) )

(?): (3 (olivedrab dodgerblue indigo plum teal darkorange) )

(?): (5 (olivedrab dodgerblue indigo plum teal darkorange) )
```


Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (color_thing)
(?): (random (plum silver aqua salmon maroon tomato))
(?): (random (plum silver aqua salmon maroon tomato))
(?): (random (plum silver aqua salmon maroon tomato))
(?): (all (plum silver aqua salmon maroon tomato))
(?): (2 (plum silver aqua salmon maroon tomato))
(?): (3 (plum silver aqua salmon maroon tomato))
(?): (5 (plum silver aqua salmon maroon tomato))
eof

Task 4a:


Untitled
(define ...)
Debug
Macro Stepper
Run
Stop
#lang racket
(define (ranks rank)
(list
(list rank 'C)
(list rank 'D)
(list rank 'H)
(list rank 'S)))
(define (deck)
(append
(ranks 2)
(ranks 3)
(ranks 4)
(ranks 5)
(ranks 6)
(ranks 7)
(ranks 8)
(ranks 9)
(ranks 'Joker)
(ranks 'J)
(ranks 'Q)
(ranks 'K)
(ranks 'A)))
(define (pick-card)
(define cards (deck))
(list-ref cards (random (length cards))))
(define (show card)
(display (rank card))
(display (suit card)))
(define (rank card)
(car card))

```
28
29 (define (show card)
30   (display (rank card))
31   (display (suit card)))
32
33 (define (rank card)
34   (car card))
35
36 (define (suit card) (cadr card))
37
38 (define (red? card)
39   (or
40    (eq? (suit card) 'D)
41    (eq? (suit card) 'H)))
42
43 (define (black? card)
44   (not (red? card)))
45
46 (define (aces? card1 card2)
47   (and
48    (eq? (rank card1) 'A)
49    (eq? (rank card2) 'A)))
50
51
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (define c1 '(7 C))
> (define c2 '(Q H))
> c1
'(7 C)
> c2
'(Q H)
> (rank c1)
7
> (suit c1)
'C
> (rank c2)
'Q
> (suit c2)
'H
> (red? c1)
#f
> (red? c2)
#t
> (black? c1)
#t
> (black? c2)
#f
> (aces? '(A C) '(A S))
#t
> (aces? '(K S) '(A C))
#f
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (ranks 4)
'((4 C) (4 D) (4 H) (4 S))
> (ranks 'K)
'((K C) (K D) (K H) (K S))
> (length (deck))
52
> (display (deck))
((2 C) (2 D) (2 H) (2 S) (3 C) (3 D) (3 H) (3 S) (4 C) (4 D) (4 H) (4 S) (5 C) (5 ↵
D) (5 H) (5 S) (6 C) (6 D) (6 H) (6 S) (7 C) (7 D) (7 H) (7 S) (8 C) (8 D) (8 H)  ↵
(8 S) (9 C) (9 D) (9 H) (9 S) (Joker C) (Joker D) (Joker H) (Joker S) (J C) (J D) ↵
(J H) (J S) (Q C) (Q D) (Q H) (Q S) (K C) (K D) (K H) (K S) (A C) (A D) (A H) (A  ↵
S))
> (pick-card)
'(K S)
> (pick-card)
'(8 S)
> (pick-card)
'(2 H)
> (pick-card)
'(6 H)
> (pick-card)
'(6 H)
> (pick-card)
'(5 H)
>
```

Task 4b:

CODE:

```
|
|
|
|
|
```

```
52
53 (define (eqs? card1 card2)
54   (eq? (suit card1) (suit card2)))
55
56 (define (eqr? card1 card2)
57   (eq? (rank card1) (rank card2)))
58
59 (define (eqc? card1 card2)
60   (and
61    (eqr? card1 card2)
62    (eqs? card1 card2)))
63
64 (define (pick-2-cards)
65   (define c1 (pick-card))
66   (define c2 (pick-card))
67   (cond
68     ((eqc? c1 c2)
69      (pick-2-cards))
70     (else
71      (list c1 c2))))
72
73 (define (higher-rank c1 c2)
74   (define r1 (num (rank c1)))
75   (define r2 (num (rank c2)))
76   (cond
77     ((> r1 r2) (display (rank c1)))
78     (else (display (rank c2)))))
79
80 (define (num r)
81   (cond
82     ((number? r) r)
83     ((eq? r 'Joker) 10)
84     ((eq? r 'J) 11)
85     ((eq? r 'Q) 12)
86     ((eq? r 'K) 13)
87     ((eq? r 'A) 14)))
88
89
90 (define (flush c1 c2)
91   (define s1 (suit c1))
92   (define s2 (suit c2))
93   (cond
94     ((eq? s1 s2) (display " flush!"))))
95
96 (define (straight c1 c2)
97   (define r1 (num (rank c1)))
98   (define r2 (num (rank c2)))
99   (cond
00     ((eq? (- r1 1) r2) (display " straight!"))
01     ((eq? (+ r1 1) r2) (display " straight!"))))
02
03 (define (classify-2-cards-ur l)
04   (display l) (display ":  ")
05   (define eleven (car l))
06   (define twelve (cadr l))
07   (higher-rank eleven twelve)
08   (display " high")
09   (straight eleven twelve)
10   (flush eleven twelve)
11   (display "\n"))
12
```

CLASSIFIER DEMO:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( classify-2-cards-ur ( pick-2-cards ) )
((K C) (8 S)):  K high
> ( classify-2-cards-ur ( pick-2-cards ) )
((3 S) (J H)):  J high
> ( classify-2-cards-ur ( pick-2-cards ) )
((7 C) (3 H)):  7 high
> ( classify-2-cards-ur ( pick-2-cards ) )
((7 H) (Joker S)):  Joker high
> ( classify-2-cards-ur ( pick-2-cards ) )
((7 H) (Joker S)):  Joker high
> ( classify-2-cards-ur ( pick-2-cards ) )
((J D) (K S)):  K high
> ( classify-2-cards-ur ( pick-2-cards ) )
((7 D) (J C)):  J high
> ( classify-2-cards-ur ( pick-2-cards ) )
((8 H) (7 H)):  8 high straight! flush!
> ( classify-2-cards-ur ( pick-2-cards ) )
((6 S) (7 C)):  7 high straight!
> ( classify-2-cards-ur ( pick-2-cards ) )
((K H) (5 S)):  K high
> ( classify-2-cards-ur ( pick-2-cards ) )
((2 S) (8 D)):  8 high
> ( classify-2-cards-ur ( pick-2-cards ) )
((9 D) (K S)):  K high
> ( classify-2-cards-ur ( pick-2-cards ) )
((J C) (6 D)):  J high
> ( classify-2-cards-ur ( pick-2-cards ) )
((5 S) (6 S)):  6 high straight! flush!
> ( classify-2-cards-ur ( pick-2-cards ) )
((3 H) (J H)):  J high flush!
> ( classify-2-cards-ur ( pick-2-cards ) )
((8 H) (7 S)):  8 high straight!
> ( classify-2-cards-ur ( pick-2-cards ) )
((J S) (4 C)):  J high
> ( classify-2-cards-ur ( pick-2-cards ) )
((5 S) (Joker S)):  Joker high flush!
> ( classify-2-cards-ur ( pick-2-cards ) )
((2 C) (6 D)):  6 high
> ( classify-2-cards-ur ( pick-2-cards ) )
((Q H) (A S)):  A high
>
```

PICK 2 CARDS DEMO:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (pick-2-cards)
'((7 S) (K C))
> (pick-2-cards)
'((5 S) (Q H))
> (pick-2-cards)
'((4 C) (A D))
> (pick-2-cards)
'((6 C) (K H))
> (pick-2-cards)
'((5 S) (J S))
>
```

HIGHER RANK DEMO:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (higher-rank (pick-card) (pick-card))
>(higher-rank '(8 C) '(8 H))
8<#<void>
> (higher-rank (pick-card) (pick-card))
>(higher-rank '(7 D) '(3 C))
7<#<void>
> (higher-rank (pick-card) (pick-card))
>(higher-rank '(5 D) '(7 D))
7<#<void>
> (higher-rank (pick-card) (pick-card))
>(higher-rank '(A H) '(9 D))
A<#<void>
> (higher-rank (pick-card) (pick-card))
>(higher-rank '(6 H) '(A H))
A<#<void>
>
```

Task 4c:
CODE:

```
(define (BETTER r)
  (cond
    ((eq? r '2) (display "TWO"))
    ((eq? r '3) (display "THREE"))
    ((eq? r '4) (display "FOUR"))
    ((eq? r '5) (display "FIVE"))
    ((eq? r '6) (display "SIX"))
    ((eq? r '7) (display "SEVEN"))
    ((eq? r '8) (display "EIGHT"))
    ((eq? r '9) (display "NINE"))
    ((eq? r 'Joker) (display "JOKER"))
    ((eq? r 'J) (display "JACK"))
    ((eq? r 'Q) (display "QUEEN"))
    ((eq? r 'K) (display "KING"))
    ((eq? r 'A) (display "ACE"))))

    (define (higher-rank-better c1 c2)
      (define r1 (num (rank c1)))
      (define r2 (num (rank c2)))
      (cond
        ((> r1 r2) (BETTER (rank c1)))
        (else (BETTER (rank c2)))))
```

```
(define (classify-2-cards l)
  (display l) (display ":  ")
  (define eleven (car l))
  (define twelve (cadr l))
  (higher-rank-better eleven twelve)
  (display " high")
  (straight eleven twelve)
  (flush eleven twelve)
  (display "\n"))
```

BETTER CLASSIFIER:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (classify-2-cards (pick-2-cards))
((K C) (6 S)):  KING high
> (classify-2-cards (pick-2-cards))
((6 S) (2 D)):  SIX high
> (classify-2-cards (pick-2-cards))
((2 H) (4 S)):  FOUR high
> (classify-2-cards (pick-2-cards))
((5 S) (7 D)):  SEVEN high
> (classify-2-cards (pick-2-cards))
((K D) (5 D)):  KING high flush!
> (classify-2-cards (pick-2-cards))
((5 D) (K C)):  KING high
> (classify-2-cards (pick-2-cards))
((2 H) (4 C)):  FOUR high
> (classify-2-cards (pick-2-cards))
((6 C) (Q C)):  QUEEN high flush!
> (classify-2-cards (pick-2-cards))
((Q D) (8 C)):  QUEEN high
> (classify-2-cards (pick-2-cards))
((9 C) (5 H)):  NINE high
> (classify-2-cards (pick-2-cards))
((4 S) (5 C)):  FIVE high straight!
> (classify-2-cards (pick-2-cards))
((3 C) (Joker D)):  JOKER high
> (classify-2-cards (pick-2-cards))
((4 S) (3 C)):  FOUR high straight!
> (classify-2-cards (pick-2-cards))
((9 S) (3 D)):  NINE high
> (classify-2-cards (pick-2-cards))
((5 S) (9 S)):  NINE high flush!
> (classify-2-cards (pick-2-cards))
((7 C) (4 D)):  SEVEN high
> (classify-2-cards (pick-2-cards))
((5 H) (6 H)):  SIX high straight! flush!
> (classify-2-cards (pick-2-cards))
((2 H) (Q H)):  QUEEN high flush!
> (classify-2-cards (pick-2-cards))
((6 S) (K S)):  KING high flush!
> (classify-2-cards (pick-2-cards))
((Q C) (Joker H)):  QUEEN high
>
```