Racket Programming Assignment #3: Lambda and Basic Lisp

Learning Abstract:

This assignment's first two assignments demonstrate interactions with lambda functions and list processing techniques in lisp. The second objective involves developing more programs that demonstrate the "basic list processing" demonstrated in lesson 6. The final job allows me to experiment with alternative color interpretations inside the racket context. Finally, the fourth job expands on the card code from lesson 6.

Task 1 - Lambda:

Task 1a - Three ascending integers:

```
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 - Make list in reverse order:

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

Task 1c - Random number generator:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 3 5)
5
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 3 5)
3
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 3 5)
4
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 3 5)
4
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 3 5)
5
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 3 5)
4
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 3 5)
3
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 11 17)
13
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 11 17)
16
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 11 17)
15
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 11 17)
12
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 11 17)
16
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 11 17)
11
> ((lambda (x y) (+ x(random ( -(+ y 1) x)))) 11 17)
17
>
```

Task 2 - List Processing References and Constructors:

```
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)
> ( cadr colors )
'blue
> ( cddr colors )
'(yellow orange)
> ( first color )
```

```
> ( 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 fa so la ti) )
> ( car ( cdr ( cdr ( cdr pitches ) ) ) )
'fa
> ( cadddr pitches )
'fa
> ( list-ref pitches 3 )
'fa
> ( 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 3 - Little Color Interpreter:

Task 3a - Establishing the Sampler code from Lesson 6:

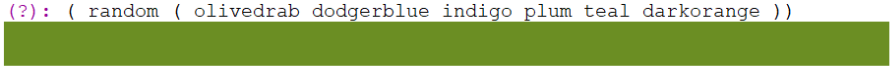
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
>  ( sampler )
(?): ( red orange yellow green blue indigo violet )
violet
(?): ( red orange yellow green blue indigo violet )
violet
(?): ( red orange yellow green blue indigo violet )
red
(?): ( red orange yellow green blue indigo violet )
red
(?): ( red orange yellow green blue indigo violet )
violet
(?): ( red orange yellow green blue indigo violet )
yellow
(?):  ( aet ate eat eta tae tea )
eta
(?): ( aet ate eat eta tae tea )
ate
(?): ( aet ate eat eta tae tea )
ate
(?): ( aet ate eat eta tae tea )
aet
(?): ( aet ate eat eta tae tea )
tae
(?):  ( aet ate eat eta tae tea )
ate


(?):  ( 0 1 2 3 4 5 6 7 8 9 )
9
(?): ( 0 1 2 3 4 5 6 7 8 9 )
0
(?): ( 0 1 2 3 4 5 6 7 8 9 )
6
(?): ( 0 1 2 3 4 5 6 7 8 9 )
3
(?): ( 0 1 2 3 4 5 6 7 8 9 )
7
(?): ( 0 1 2 3 4 5 6 7 8 9 )
5
```

Task 3b - Color Thing Interpreter:

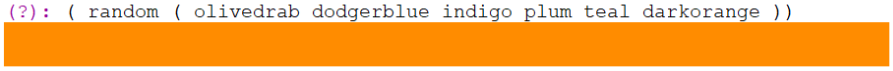
```
#lang racket

( require 2htdp/image )

( define ( color-thing )
( display "(?): " )
( define the-list ( read ) )
   ( define c1 ( car the-list ))
   ( define c2 ( cadr the-list ))

( define the-element
( list-ref c2 ( random ( length c2 ) ) )
)

   ( define (rectangles color ) ( rectangle 600 30 "solid" color))



( define ( all-colors n )
   ( define colors ( list-ref c2 ( - n 1 )))
   ( display ( rectangles colors ))
   ( display "\n" )
   ( cond
      (( eq? 1 n) ( display "\n" ))
      ( else
       ( all-colors ( - n 1 ))
       )))

   ( define (option c1)
      ( cond
         (( eq? c1 'random )
          ( display (rectangles the-element)) (display "\n" ))
         (( eq? c1 'all-colors )
          ( all-colors ( length c2 )))
         ( else
           ( display ( rectangles ( list-ref c2 ( - c1 1 ))))( display "\n"))))

   (option c1 )
   ( color-thing)
   )
```

Language: racket, with debugging; memory limit: 128 MB.

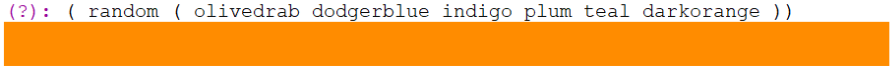
```
> ( color-thing )
(?): ( random ( olivedrab dodgerblue indigo plum teal darkorange ))
```



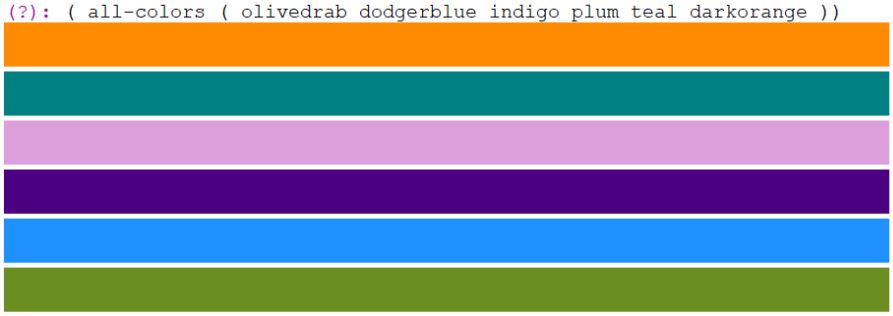
```
(?): ( random ( olivedrab dodgerblue indigo plum teal darkorange ))
```



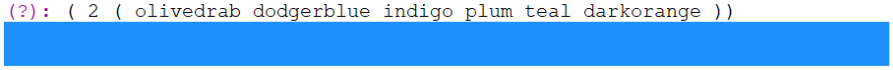
```
(?): ( random ( olivedrab dodgerblue indigo plum teal darkorange ))
```



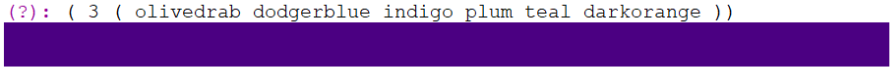
```
(?): ( all-colors ( olivedrab dodgerblue indigo plum teal darkorange ))
```



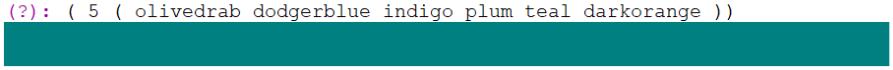
```
(?): ( 2 ( olivedrab dodgerblue indigo plum teal darkorange ))
```



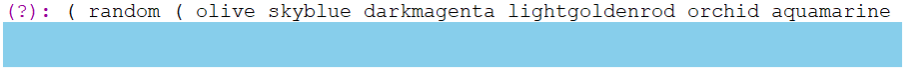
```
(?): ( 3 ( olivedrab dodgerblue indigo plum teal darkorange ))
```



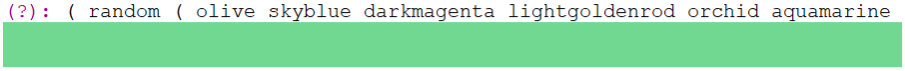
```
(?): ( 5 ( olivedrab dodgerblue indigo plum teal darkorange ))
```



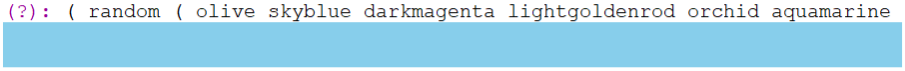
```
(?): ( random ( olive skyblue darkmagenta lightgoldenrod orchid aquamarine ))
```



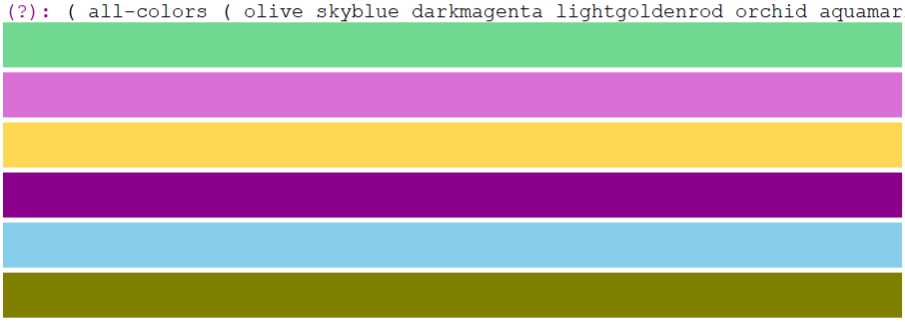
```
(?): ( random ( olive skyblue darkmagenta lightgoldenrod orchid aquamarine ))
```



```
(?): ( random ( olive skyblue darkmagenta lightgoldenrod orchid aquamarine ))
```



```
(?): ( all-colors ( olive skyblue darkmagenta lightgoldenrod orchid aquamarine ))
```



```
(?): ( 2 ( olive skyblue darkmagenta lightgoldenrod orchid aquamarine ))
```

```
(?): ( 3 ( olive skyblue darkmagenta lightgoldenrod orchid aquamarine ))
```

```
(?): ( 5 ( olive skyblue darkmagenta lightgoldenrod orchid aquamarine ))
```

Task 4 - Two Card Poker:

Task 4a - Establishing Card code from Lesson 6 :

```
#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 'X )
( ranks 'J )
( ranks 'Q )
( ranks 'K )
( ranks 'A )
)
)
( define ( pick-a-card )
( define cards ( deck ) )
( list-ref cards ( random ( length cards ) ) )
)
( define ( show card )
( display ( rank card ) )
( display ( suit card ) )
)
( define ( rank card )
( car card )
)
( define ( suit card )
( cadr card )
)
( define ( red? card )
( or
( equal? ( suit card ) 'D )
( equal? ( suit card ) 'H )
)
)
( define ( black? card )
( not ( red? card ) )
)
( define ( aces? card1 card2 )
( and
( equal? ( rank card1 ) 'A )
( equal? ( rank card2 ) 'A )
)
)
```

```
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 ) )
': undefined;
 cannot reference an identifier before its definition
> ( aces? '( A C ) '( A S ) )
#t
> ( aces? '( K S ) '( A C ) )
#f
> ( 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)
(X C) (X D) (X H) (X 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-a-card )
'(X C)
>  ( pick-a-card )
'(8 D)
> ( pick-a-card )
'(3 C)
> ( pick-a-card )
'(4 H)
> ( pick-a-card )
'(6 H)
> ( pick-a-card )
'(4 S)
>
```