

CSC 344 Programming Languages

Learning Abstract: This assignment introduces us to Lambda and Basic Lisp in Racket. This assignment helped me gain a deeper understanding of the Lambda function in Racket as well as Lisp which overall helps improve my problem solving skills.

```

Welcome to DrRacket, version 8.6 [cs].
Language: Determine language from source; 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)
> ( (lambda (x y z) (list z y x) ) 'red 'yellow 'blue)
'(blue yellow red)
> ( (lambda (x y z) (list z y x) ) '10 '20 '30)
'(30 20 10)
> ( (lambda (x y z) (list z y x) ) "Professor Plum" "Colonel Mustard" "Miss Scarlet")
'("Miss Scarlet" "Colonel Mustard" "Professor Plum")
> |

```

```

Welcome to D-BugKit, version 8.6 [cs]
Language: racket, with debugging: memory limit: 128 MB
> ((lambda (x y) (random x (+ y 1) )) 3 5)
5
> ((lambda (x y) (random x (+ y 1) )) 3 5)
4
> ((lambda (x y) (random x (+ y 1) )) 3 5)
4
> ((lambda (x y) (random x (+ y 1) )) 3 5)
5
> ((lambda (x y) (random x (+ y 1) )) 3 5)
5
> ((lambda (x y) (random x (+ y 1) )) 3 5)
4
> ((lambda (x y) (random x (+ y 1) )) 3 5)
3
> ((lambda (x y) (random x (+ y 1) )) 3 5)
4
> ((lambda (x y) (random x (+ y 1) )) 3 5)
3
> ((lambda (x y) (random x (+ y 1) )) 3 5)
5
> ((lambda (x y) (random x (+ y 1) )) 11 17)
15
> ((lambda (x y) (random x (+ y 1) )) 11 17)
12
> ((lambda (x y) (random x (+ y 1) )) 11 17)
17
> ((lambda (x y) (random x (+ y 1) )) 11 17)
17
> ((lambda (x y) (random x (+ y 1) )) 11 17)
13
> ((lambda (x y) (random x (+ y 1) )) 11 17)
15
> ((lambda (x y) (random x (+ y 1) )) 11 17)
14
> ((lambda (x y) (random x (+ y 1) )) 11 17)
14
> ((lambda (x y) (random x (+ y 1) )) 11 17)
13
> ((lambda (x y) (random x (+ y 1) )) 11 17)
13
> ((lambda (x y) (random x (+ y 1) )) 11 17)
16
>

```

Task 2 Demos:

```

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
> (car colors)
'red
> (cdr colors)
'(blue yellow orange)
> (car (cdr colors) )
'blue
> (cdr (cdr colors) )
'(yellow orange)
> (cadr colors)
'blue
> (caddr 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 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) )
> (caddr pitches)
'fa
> (car (cdr (cdr pitches) ) )
pitches: undefined;
cannot reference an identifier before its definition
> (car (cdr (cdr 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 ) ) )
car: arity mismatch;
the expected number of arguments does not match the given number
expected: 1
given: 2
> (cons (car x (cons (car (cdr x) ) y ) ) x)
car: arity mismatch;
the expected number of arguments does not match the given number
expected: 1
given: 2
> (cons (car x ) (cons (car (cdr x) ) y ) )
'(1 one 2 two)
> (append x y)
'(1 one 2 two)

```

Task 3 Code and Demo:

```
Untitled 6 - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help
Untitled 6v (define ...)
1 #lang racket
2 (define (sampler)
3   (display "(?): " )
4   (define the-list (read) )
5   (define the-element
6     (list-ref the-list (random (length the-list) ) )
7   )
8   (display the-element ) (display "\n")
9   (sampler) )
10
11
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)
orange
(?): (red orange yellow green blue indigo violet)
red
(?): (red orange yellow green blue indigo violet)
green
(?): (red orange yellow green blue indigo violet)
orange
(?): (aet ate eat eta tae tea)
tae
(?): (aet ate eat eta tae tea)
tae
(?): (aet ate eat eta tae tea)
eat
(?): (aet ate eat eta tae tea)
eta
(?): (aet ate eat eta tae tea)
tae
(?): (aet ate eat eta tae tea)
tae
(?): ( 0 1 2 3 4 5 6 7 8 9 )
8
(?): ( 0 1 2 3 4 5 6 7 8 9 )
1
(?): ( 0 1 2 3 4 5 6 7 8 9 )
5
(?): ( 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 )
0
```

Task 4 Demo and Code:

```
1 #lang racket
23 (ranks 'Q)
24 (ranks 'K)
25 (ranks 'A)
26 )
27 )
28
29 (define (pick-a-card)
30   (define cards (deck) )
31   (list-ref cards (random (length cards) ) )
32 )
33
34 (define (show card)
35   (display (rank card) )
36   (display (suit card) )
37 )
38
39 (define (rank card)
40   (car card)
41 )
42
43 (define (suit card)
44   (cadr card)
45 )
46
47 (define (red? card)
48   (or
49     (equal? (suit card) 'D)
50     (equal? (suit card) 'H)
51   )
52 )
53
54 (define (black? card)
55   (not (red? card) )
56 )
57
58 (define (aces? card1 card2)
59   (and
60     (equal? (rank card1) 'A)
61     (equal? (rank card2) 'A)
62   )
63 )
64
1 #lang racket
2 (define (ranks rank)
3   (list
4     (list rank 'C)
5     (list rank 'D)
6     (list rank 'H)
7     (list rank 'S)
8   )
9 )
10
11 (define (deck)
12   (append
13     (ranks 2)
14     (ranks 3)
15     (ranks 4)
16     (ranks 5)
17     (ranks 6)
18     (ranks 7)
19     (ranks 8)
20     (ranks 9)
21     (ranks 'X)
22     (ranks 'J)
23     (ranks 'Q)
24     (ranks 'K)
25     (ranks 'A)
26   )
27 )
28
29 (define (pick-a-card)
30   (define cards (deck) )
31   (list-ref cards (random (length cards) ) )
32 )
33
34 (define (show card)
35   (display (rank card) )
36   (display (suit card) )
37 )
38
39 (define (rank card)
40   (car card)
41 )
42
43 (define (suit card)
```

```

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
> (rank c2)
Q
> (suit c1)
C
> (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
> (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 2
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) 2
(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 2
D) (A H) (A S))
>

```

```

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (pick-a-card)
'(5 D)
> (pick-a-card)
'(X C)
> (pick-a-card)
'(4 D)
> (pick-a-card)
'(4 C)
> (pick-a-card)
'(8 D)
> (pick-a-card)
'(A S)
>

```