

Assignment: **Racket Assignment 4 – Lambda and Basic Lisp**

LEARNING ABSTRACT

The project is aimed to introduce us to Lambda and Basic Lisp in Racket programming language. I was able to gain a basic understanding on the Lambda function in racket, as well as Lisp too. This project helped me to develop a deeper appreciation for the power and elegance of Lisp and Lambda functions, as well as to improve my problem-solving abilities and programming skills.

Task 1: Lambda

Demo for Task 1a – Three ascending integers

Welcome to [DrRacket](#), version 8.7 [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)
>
```

Demo for Task 1b - Make list in reverse order

Welcome to [DrRacket](#), version 8.7 [cs].

Language: [racket](#), with [debugging](#); memory limit: 128 MB.

```
> ( ( lambda ( a b c ) ( list c b a ) ) 'red 'yellow 'blue )
'(blue yellow red)
> ( ( lambda ( a b c ) ( list c b a ) ) 10 20 30 )
'(30 20 10)
> ( ( lambda ( a b c ) ( list c b a ) )
    "Professor Plum" "Colonel Mustard" "Miss Mustard" )
'("Miss Mustard" "Colonel Mustard" "Professor Plum")
> |
```

Demo for Task 1c – Random number generator

Welcome to [DrRacket](#), version 8.7 [cs].

Language: [racket](#), with [debugging](#); memory limit: 128 MB.

```
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
3
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
5
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
4
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
3
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
4
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
3
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
3
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
3
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 3 5 )
4
>
```

```

> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
11
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
14
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
11
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
14
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
13
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
11
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
17
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
16
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
17
> ( ( lambda ( a b ) ( random a ( + b 1 ) ) ) 11 17 )
11
> |

```

Task 2: List Processing Referencers and Constructors

Demo


Welcome to [DrRacket](#), version 8.7 [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
> ( 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 pitches '( do re mi fa so la ti ) )
> ( car ( cdr ( cdr ( cdr animals ) ) ) )
 animals: undefined;
cannot reference an identifier before its definition
> ( caddr 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: The Sampler Program

Code

```

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 | )

```

Demo

Welcome to [DrRacket](#), version 8.7 [cs].

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

```
> ( sampler )
```

```
(?): ( red orange yellow green blue indigo violet )
```

```
indigo
```

```
(?): ( red orange yellow green blue indigo violet )
```

```
red
```

```
(?): ( red orange yellow green blue indigo violet )
```

```
yellow
```

```
(?): ( red orange yellow green blue indigo violet )
```

```
yellow
```

```
(?): ( red orange yellow green blue indigo violet )
```

```
yellow
```

```
(?): ( red orange yellow green blue indigo violet )
```

```
green
```

```
(?): ( aet ate eat eta tae tea )
```

```
aet
```

```
(?): ( aet ate eat eta tae tea )
```

```
eat
```

```
(?): ( aet ate eat eta tae tea )
```

```
tea
```

```
(?): ( aet ate eat eta tae tea )
```

```
tae
```

```
(?): ( aet ate eat eta tae tea )
```

```
eat
```

```
(?): ( aet ate eat eta tae tea )
```

```
tae
```

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

```
0
```

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

```
1
```

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

```
2
```

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

```
5
```

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

```
5
```

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

```
4
```

```
.....
```

Task 4: Playing Cards

Code

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

Demo

```
Welcome to DrRacket, version 8.7 [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
> ( 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) 2
(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) 2
(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) 2
(K D) (K H) (K S) (A C) (A D) (A H) (A S))
> ( pick-a-card )
'(A D)
> ( pick-a-card )
'(8 D)
> ( pick-a-card )
'(6 S)
> ( pick-a-card )
'(9 H)
> ( pick-a-card )
'(4 H)
> ( pick-a-card )
'(3 S)
>
```