

Joseph Scollo

Abstract:

This project illustrates various examples of racket and lisp processing,
As lists, list references, and lambda functions.

Task 1: Lambda

Demo for Task 1a – Three ascending integers

```
Welcome to DrRacket, version 8.7 [cs].  
Language: racket, with debugging; memory limit: 2048 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) '() ))))10)  
'(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: 2048 MB.  
> ((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")  
>
```

Demo for Task 1c – Random Number Generator

```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 2048 MB.
> ((lambda (x y) (random x y)) 3 5)
3
> ((lambda (x y) (random x y)) 3 5)
3
> ((lambda (x y) (random x y)) 3 5)
3
> ((lambda (x y) (random x y)) 3 5)
4
> ((lambda (x y) (random x y)) 3 5)
4
> ((lambda (x y) (random x y)) 3 5)
4
> ((lambda (x y) (random x y)) 3 5)
3
> ((lambda (x y) (random x y)) 3 5)
4
> ((lambda (x y) (random x y)) 3 5)
3
> ((lambda (x y) (random x y)) 3 5)
4
> ((lambda (x y) (random x y)) 11 17)
11
> ((lambda (x y) (random x y)) 11 17)
16
> ((lambda (x y) (random x y)) 11 17)
14
> ((lambda (x y) (random x y)) 11 17)
12
> ((lambda (x y) (random x y)) 11 17)
11
> ((lambda (x y) (random x y)) 11 17)
15
> ((lambda (x y) (random x y)) 11 17)
13
> ((lambda (x y) (random x y)) 11 17)
15
> ((lambda (x y) (random x y)) 11 17)
11
> ((lambda (x y) (random x y)) 11 17)
16
>
```




Task 2: List References and Constructors

Demo

 Untitled 6 - DrRacket*
File Edit View Language Racket Insert Scripts
Untitled 6 ▾ (define ...) ▾ ➡ 
1 | #lang racket

'(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

```

> (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)
> (car(cdr(cdr(cdr animals))))
  animals: undefined;
cannot reference an identifier before its definition
> (caddr pitches)
'fa
> (list-ref pitches 3)
'fa
> (cons a(cons b(cons c '())))
 ;%app: missing procedure expression.
probably originally (), which is an identifier
> (cons a(cons b(cons c '())))
'(alligator pussycat chimpanzee)
> (list a b c)
'(alligator pussycat chimpanzee)
> (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 |
3 | (define (sampler)
4 |   (display "(?): ")
5 |   (define the-list (read))
6 |   (define the-element
7 |     (list-ref the-list (random(length the-list))))
8 |   )
9 |   (display the-element)
10 |   (display "\n")
11 |   (sampler)
12 | ) |
```

Demo

```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 2048 MB.
> (sampler)
(?: (red orange yellow green blue indigo violet)
violet
(?: (red orange yellow green blue indigo violet)
orange
(?: (red orange yellow green blue indigo violet)
indigo
(?: (red orange yellow green blue indigo violet)
red
(?: (red orange yellow green blue indigo violet)
orange
(?: (red orange yellow green blue indigo violet)
violet
(?: (aet ate eat eta tae tea)
tea
(?: (aet ate eat eta tae tea)
eat
(?: (aet ate eat eta tae tea)
eat
(?: (aet ate eat eta tae tea)
eta
(?: (aet ate eat eta tae tea)
ate
(?: (aet ate eat eta tae tea)
eta
(?: (0 1 2 3 4 5 6 7 8 9)
7
(?: (0 1 2 3 4 5 6 7 8 9)
5
(?: (0 1 2 3 4 5 6 7 8 9)
2
(?: (0 1 2 3 4 5 6 7 8 9)
8
(?: (0 1 2 3 4 5 6 7 8 9)
9
(?: (0 1 2 3 4 5 6 7 8 9)
2
(?: .. user break
```

Task 4 – Playing Cards

Code

```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 2048 MB.
> (define c1 '(7 C))
> (define c2 '(Q H))
> c1
'(7 C)
> c2
'(Q H)
> (rank c1)
7
> (suit c1)
'C
> (rank c2)
'Q
> (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 S))
#f
```

Demo

```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 2048 MB.
> (define c1 '(7 C))
> (define c2 '(Q H))
> c1
'(7 C)
> c2
'(Q H)
> (rank c1)
7
> (suit c1)
'C
> (rank c2)
'Q
> (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 S))
#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)
(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)
' (K H)
> (pick-a-card)
' (8 C)
> (pick-a-card)
' (A H)
> (pick-a-card)
' (5 C)
> (pick-a-card)
' (3 C)
> (pick-a-card)
' (K S)
> g
```
