Temitope Emokpae 10/26/2022

CSC 344

Racket Programming Assignment #4: RLP and HoFs

Learning Abstract

This racket programming assignment goes in-depth with recursive lists and the practice of higher-order functions. It also involves using the 'map', 'filter', and 'foldr' functions.

Problem 1 - Generate Uniform List

Code:

```
#lang racket
( define ( generate-uniform-list number object )
   ( cond
      ( ( = number 0 )
        '()
        )
      ( ( > number 0 )
        ( cons object ( generate-uniform-list ( - number 1 ) object ) )
        )
      )
   )
```

Demo:

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( generate-uniform-list 5 'kitty )
'(kitty kitty kitty kitty kitty)
> ( generate-uniform-list 10 2 )
'(2 2 2 2 2 2 2 2 2 2)
> ( generate-uniform-list 0 'whatever )
'()
> ( generate-uniform-list 2 '( racket prolog haskell rust ) )
'((racket prolog haskell rust) (racket prolog haskell rust))
>
```

Problem 2 - Association List Generator

Code:

```
#lang racket
( define ( a-list list1 list2 )
   ( cond
      ( ( = ( length list1 ) 0 )
        '()
        )
      ( ( > ( length list1 ) 0 )
        ( cons ( cons ( car list1 ) ( car list2 ) ) ( a-list ( cdr list1 ) ( cdr list2 ) ) )
        )
      )
   )
```

Demo:

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( a-list '( one two three four five ) '( un deux trois quatre cinq ) )
'((one . un) (two . deux) (three . trois) (four . quatre) (five . cinq))
> ( a-list '() '() )
'()
> ( a-list '( this ) '( that ) )

'((this . that))
> ( a-list '( one two three ) '( ( 1 ) ( 2 2 ) ( 3 3 3 ) ) )
'((one 1) (two 2 2) (three 3 3 3))
>
```

Problem 3 - Assoc

Code:

```
( define ( assoc object a-list )
   ( cond
      ( ( = ( length a-list ) 0 )
        '()
        )
      ( ( eq? ( car ( car a-list ) ) object )
        ( car a-list )
        )
        ( else
          ( assoc object ( cdr a-list ) )
          )
        )
      )
```

Demo:

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( define al1
     ( a-list '( one two three four ) '( un deux trois quatre ) )
     )
> ( define al2
     ( a-list '( one two three ) '( ( 1 ) ( 2 2 ) ( 3 3 3 ) ) )
     )
> al1
'((one . un) (two . deux) (three . trois) (four . quatre))
> ( assoc 'two al1 )
'(two . deux)
> ( assoc 'five al1 )
'()
> al2
'((one 1) (two 2 2) (three 3 3 3))
> ( assoc 'three al2 )
'(three 3 3 3)
> ( assoc 'four al2 )
'()
>
```

Problem 4 - Rassoc

Code:

```
( define ( rassoc object list )
   ( cond
      ( ( empty? list )
        '()
        )
      ( ( eq? object ( cdr ( car list ) ) )
        ( car list )
        )
      ( else
        ( rassoc object ( cdr list ) )
        )
      )
   )
```

Demo:

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( define al1
     ( a-list '(one two three four ) '(un deux trois quatre ) )
     )
> ( define al2
     ( a-list '(one two three) '( (1) (2 2) ( 3 3 3 ) ) )
     )
> al1
'((one . un) (two . deux) (three . trois) (four . quatre))
> ( rassoc 'three al1 )
'()
> ( rassoc 'trois al1 )
'(three . trois)
> al2
'((one 1) (two 2 2) (three 3 3 3))
> ( rassoc '(1) al2 )
'()
> ( rassoc '(3 3 3) al2 )
'()
> ( rassoc 1 al2 )
'()
>
```

Problem 5 - Los->s

Code:

```
( define ( los->s list )
   ( cond
      ( ( empty? list )
        ""
        )
      ( ( empty? ( cdr list ) )
        ( car list ) )
      ( else
        ( string-append ( car list ) " " (los->s ( cdr list ) ) )
        )
      )
   )
```

Demo:

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( los->s '( "red" "yellow" "blue" "purple" ) )
"red yellow blue purple"
> ( los->s ( generate-uniform-list 20 "-" ) )
"- - - - - - - - - - - - - - - - - - - -"
> ( los->s '() )
""
> ( los->s '( "whatever" ) )
"whatever"
>
```

Problem 6 - Generate list

Code:

```
#lang racket
( require 2htdp/image)
( define ( generate-list number name )
   ( cond
      ( ( = number 0 )
        '()
        )
      ( ( > number 0 )
        ( cons ( name ) ( generate-list ( - number 1 ) name ) )
        )
      )
   )

( define ( roll-die ) ( + ( random 6 ) 1 ) )
( define ( dot )
   ( circle ( + 10 ( random 41 ) ) "solid" ( random-color ) )
   )

( define ( random-color )
   ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) )
   )
( define ( rgb-value )
   ( random 256 )
   )

( define ( sort-dots loc )
   ( sort loc #:key image-width < )
   )
```

Demo 1:

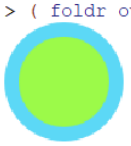
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( generate-list 10 roll-die )
'(5 6 6 2 1 3 4 2 3 6)
> ( generate-list 20 roll-die )
'(5 2 1 5 6 5 1 5 5 6 6 2 5 2 2 3 1 4 4 5)
> ( generate-list 12
    ( lambda () ( list-ref '( red yellow blue ) ( random 3 ) ) )
    )
'(blue blue red red yellow blue red yellow yellow yellow yellow red)
>
```

Demo 2:

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
> ( define dots ( generate-list 3 dot ) )
> dots
```

```
(list                                   )
> ( foldr overlay empty-image dots )
```



```
> ( sort-dots dots )
```

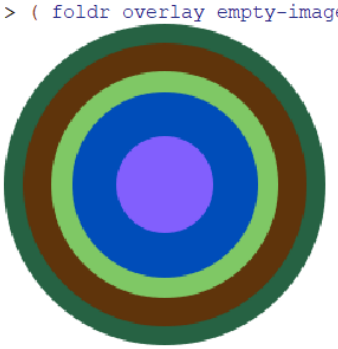
```
(list                                   )
> ( foldr overlay empty-image ( sort-dots dots ) )
```

```
>
```

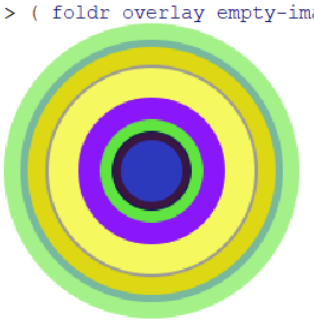
Demo 3:

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
> ( define a ( generate-list 5 big-dot ) )
> ( foldr overlay empty-image ( sort-dots a ) )
```



```
> ( define b ( generate-list 10 big-dot ) )
> ( foldr overlay empty-image ( sort-dots b ) )
```



```
>
```

Problem 7 - The Diamond

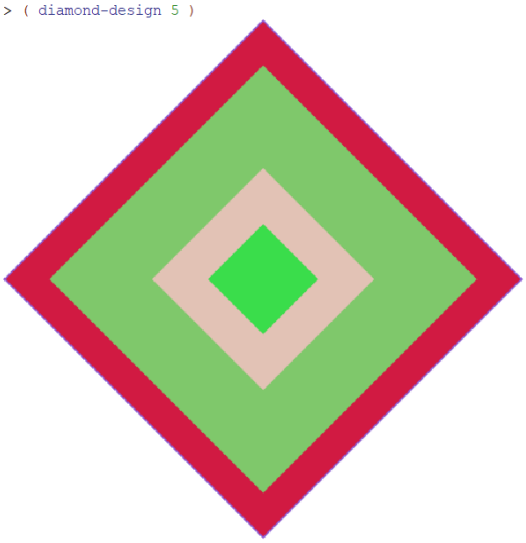
Code:

```
#lang racket
( require 2htdp/image)
( define ( generate-list number name )
   ( cond
      ( ( = number 0 )
        '()
        )
      ( ( > number 0 )
        ( cons ( name ) ( generate-list ( - number 1 ) name ) )
        )
      )
   )

( define ( diamond )
   ( rotate 45 ( square ( + 20 ( random 365 ) ) "solid" ( random-color ) ) )
   )

( define ( random-color )
   ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) )
   )

( define ( rgb-value )
   ( random 256 )
   )

( define ( sort-diamond loc )
   ( sort loc #:key image-width < )
   )

( define ( diamond-design number )
   ( define shape ( generate-list number diamond ) )
   ( foldr overlay empty-image ( sort-diamond shape ) )
   )
```

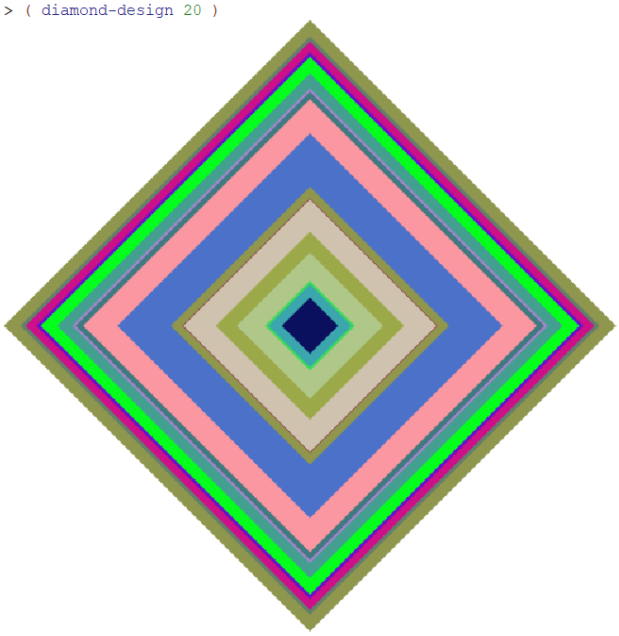
Demo:

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
> ( diamond-design 5 )
```



```
> ( diamond-design 20 )
```



```
>
```

Problem 8 - Chromesthetic renderings

Code:

```
#lang racket
( require 2htdp/image)
( define ( a-list list1 list2 )
   ( cond
      ( ( = ( length list1 ) 0 )
        '()
        )
      ( ( > ( length list1 ) 0 )
        ( cons ( cons ( car list1 ) ( car list2 ) ) ( a-list ( cdr list1 ) ( cdr list2 ) ) )
        )
      )
   )

( define pitch-classes '( c d e f g a b ) )
( define color-names '( blue green brown purple red yellow orange ) )
( define ( box color )
   ( overlay
     ( square 30 "solid" color )
     ( square 35 "solid" "black" )
     )
   )

( define boxes
   ( list
     ( box "blue" )
     ( box "green" )
     ( box "brown" )
     ( box "purple" )
     ( box "red" )
     ( box "gold" )
     ( box "orange" )
     )
   )

( define pc-a-list ( a-list pitch-classes color-names ) )
( define cb-a-list ( a-list color-names boxes ) )
( define ( pc->color pc )
   ( cdr ( assoc pc pc-a-list ) )
```

```
)

( define ( color->box color )
   ( cdr ( assoc color cb-a-list ) )
   )

( define ( play note )
   ( foldr beside empty-image ( map box ( map pc->color note ) )
          )
   )
```

Demo:

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
> ( play '( c d e f g a b c c b a g f e d c ) )
```

```
> ( play '( c c g g a a g g f f e e d d c c ) )
```

```
> ( play '( c d e c c d e c e f g g e f g g ) )
```

```
>
```

Problem 9 – Diner

Code:

```
#lang racket
( require 2htdp/image )
( define ( a-list list1 list2 )
   ( cond
      ( ( = ( length list1 ) 0 )
        '()
        )
      ( ( > ( length list1 ) 0 )
        ( cons ( cons ( car list1 ) ( car list2 ) ) ( a-list ( cdr list1 ) ( cdr list2 ) ) )
        )
      )
   )

( define choices '( bagel donut munchkin muffin croissant latte ) )

( define prices '( 4 2 3 3 5 6 ) )

( define menu
   ( a-list choices prices )
   )

( define ( foodsort s )
   ( cond
      ( ( = s 0 )
        '()
        )
      ( ( > s 0 )
        ( cons ( list-ref choices ( random 6 ) ) ( foodsort ( - s 1 ) )
               )
        )
      )
   )

( define sales ( foodsort 32 ) )

( define ( price s )
   ( cdr ( assoc s menu ) )
   )
```

```
( define ( total s t )
   ( foldr + 0 ( map price ( filter ( lambda ( x ) ( equal? x t ) ) sales ) )
          )
   )
```

Demo:

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> menu
'((bagel . 4) (donut . 2) (munchkin . 3) (muffin . 3) (croissant . 5) (latte . 6))
> sales
'(latte
  munchkin
  donut
  latte
  croissant
  latte
  munchkin
  latte
  bagel
  bagel
  latte
  latte
  latte
  latte
  latte
  bagel
  donut
  bagel
  bagel
  croissant
  bagel
  muffin
  donut
  munchkin
  donut
  muffin
  bagel
  latte
  munchkin
  munchkin
  munchkin
  munchkin)
> ( total sales 'bagel )
28
> ( total sales 'donut )
8
> ( total sales 'munchkin )
21
> ( total sales 'muffin )
6
> ( total sales 'croissant )
10
> ( total sales 'latte )
60
>
```

Problem 10 - Grapheme Color Synesthesia

Code 1:

```
#lang racket
( require 2htdp/image)
( require 2htdp/image )
( define AI ( text "A" 36 "orange" ) )
( define BI ( text "B" 36 "red" ) )
( define CI ( text "C" 36 "blue" ) )

( define alphabet '( A B C ) )
( define alphapic ( list AI BI CI ) )

( define ( a-list list1 list2 )
   ( cond
      ( ( = ( length list1 ) 0 )
        '()
        )
      ( ( > ( length list1 ) 0 )
        ( cons ( cons ( car list1 ) ( car list2 ) ) ( a-list ( cdr list1 ) ( cdr list2 ) ) )
        )
      )
   )

( define ( assoc object a-list )
   ( cond
      ( ( = ( length a-list ) 0 )
        '()
        )
      ( ( eq? ( car ( car a-list ) ) object )
        ( car a-list )
        )
      ( else
        ( assoc object ( cdr a-list ) )
        )
      )
   )

( define a->i ( a-list alphabet alphapic ) )
( define ( letter->image alphabet) ( cdr ( assoc alphabet a->i ) ) )
( define ( gcs letters )
   ( cond
      ( ( empty? letters ) ( empty-image ) )
      ( else
        ( foldr beside empty-image ( map letter->image letters ) ) )
      )
   )
```

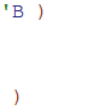
Demo 1:

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
> alphabet
'(A B C)
> alphapic
```

(list)

```
> ( display a->i )
```

((A . ) (B . ) (C .))

```
> ( letter->image 'A )
```

```
> ( letter->image 'B )
```

```
> ( gcs '( C A B ) )
```



```
> ( gcs '( B A A ) )
```



```
> ( gcs '( B A B A ) )
```



```
>
```

Code 2:

```
#lang racket
( require 2htdp/image)
( require 2htdp/image )
( define AI ( text "A" 36 "orange" ) )
( define BI ( text "B" 36 "red" ) )
( define CI ( text "C" 36 "blue" ) )
( define DI ( text "D" 36 "green" ) )
( define EI ( text "E" 36 "yellow" ) )
( define FI ( text "F" 36 "indigo" ) )
( define GI ( text "G" 36 "violet" ) )
( define HI ( text "H" 36 "brown" ) )
( define II ( text "I" 36 "black" ) )
( define JI ( text "J" 36 "cyan" ) )
( define KI ( text "K" 36 "magenta" ) )
( define LI ( text "L" 36 "maroon" ) )
( define MI ( text "M" 36 "teal" ) )
( define NI ( text "N" 36 "plum" ) )
( define OI ( text "O" 36 "orchid" ) )
( define PI ( text "P" 36 "tomato" ) )
( define QI ( text "Q" 36 "salmon" ) )
( define RI ( text "R" 36 "mint cream" ) )
( define SI ( text "S" 36 "slate blue" ) )
( define TI ( text "T" 36 "cornflower blue" ) )
( define UI ( text "U" 36 "pale turquoise" ) )
( define VI ( text "V" 36 "honeydew" ) )
( define WI ( text "W" 36 "sea green" ) )
( define XI ( text "X" 36 "dark green" ) )
( define YI ( text "Y" 36 "misty rose" ) )
( define ZI ( text "Z" 36 "hot pink" ) )


( define alphabet '( A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ) )
( define alphapic ( list AI BI CI DI EI FI GI HI II JI KI LI MI NI OI PI QI RI SI TI UI VI WI XI YI ZI ) )


( define ( a-list list1 list2 )
   ( cond
      ( ( = ( length list1 ) 0 )
        '()
        )
      ( ( > ( length list1 ) 0 )
        ( cons ( cons ( car list1 ) ( car list2 ) ) ( a-list ( cdr list1 ) ( cdr list2 ) ) )
        )
      )
   )

( define ( assoc object a-list )
   ( cond
      ( ( = ( length a-list ) 0 )
        '()
        )
      ( ( eq? ( car ( car a-list ) ) object )
        ( car a-list )
        )
      ( else
        ( assoc object ( cdr a-list ) )
        )
      )
   )

( define a->i ( a-list alphabet alphapic ) )
( define ( letter->image alphabet) ( cdr ( assoc alphabet a->i ) ) )
( define ( gcs letters )
   ( cond
      ( ( empty? letters ) ( empty-image ) )
      ( else
        ( foldr beside empty-image ( map letter->image letters ) ) )
      )
   )
```

Demo 2:

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
> ( gcs '( A L P H A B E T ) )
```

ALPHABET

```
> ( gcs '( D A N D E L I O N ) )
```

DANDELION

```
> ( gcs '( F A I R Y ) )
```

FAIRY

```
> ( gcs '( T E M I T O P E ) )
```

TEMITOPE

```
> ( gcs '( B U N N Y ) )
```

BUNNY

```
> ( gcs '( A N I M E ) )
```

ANIME

```
> ( gcs '( F L Y I N G ) )
```

FLYING

```
> ( gcs '( P L U S H I E ) )
```

PLUSHIE

```
> ( gcs '( C O M P U T E R ) )
```

COMPUTER

```
> ( gcs '( M U S I C ) )
```

MUSIC

```
>
```