Racket Assignment 4

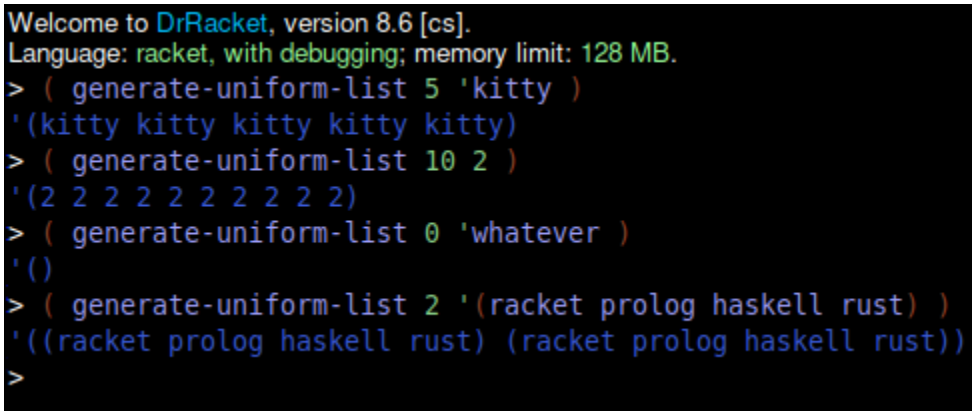
Eli Fereira

Abstract

Lab takes all the skills we've learned so far, and mashes them together. The first five tasks are five conceptually-complex list processing functions. After those five, we use the concepts, and the functions themselves from the first five tasks to perform complex tasks involving colors, shapes, recursion, and lists.

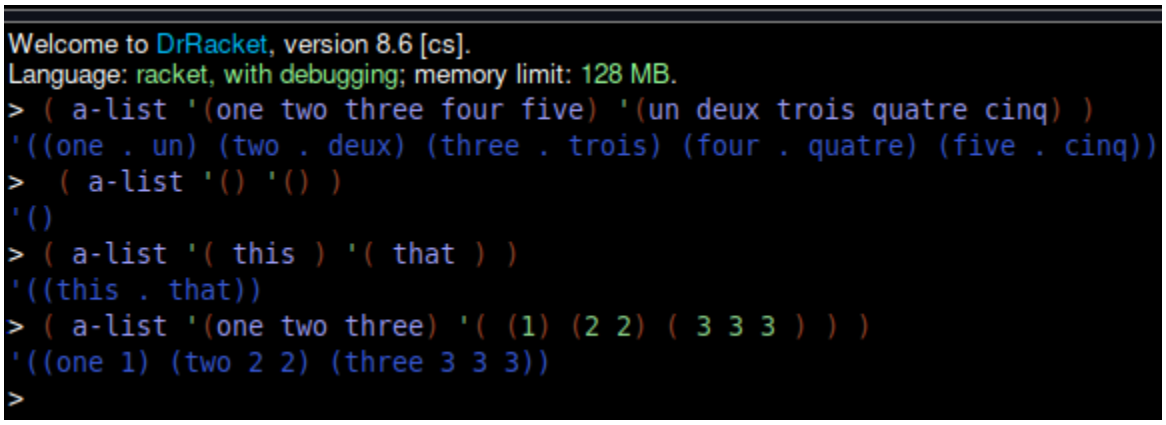
Task 1: Generate Uniform List

```
( define ( generate-uniform-list l form )
   ( cond
      ( ( = l 0) '() )
      ( else
        ( cons form ( generate-uniform-list ( - l 1 ) form ) )
        )
      )
   )
```



Task 2: Association List Generator

```
( define ( a-list l1 l2 )
   ( cond
      ( ( not ( = ( length l1 ) ( length l2) ) ) '() )
      ( ( = ( length l1) 0 ) '() )
      ( else
        ( cons
          ( cons ( car l1 ) ( car l2 ) )
          ( a-list ( cdr l1 ) (cdr l2 ) )
          )
        )
      )
   )
```



Task 3: Assoc

```
( define ( assoc el li )
   ( cond
      ( ( = ( length li ) 0 ) '() )
      ( ( equal? el ( car ( car li ) ) ) ( car li ) )
      ( else
        ( assoc el ( cdr li ) )
        )
      )
   )
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 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 )
'()
>
```

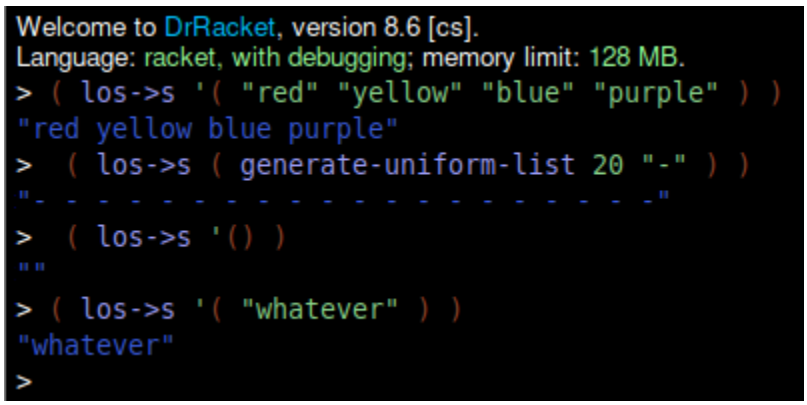
Task 4: Rassoc

```
( define ( rassoc el li )
   ( cond
      ( ( = ( length li ) 0 ) '() )
      ( ( equal? el ( cdr ( car li ) ) ) ( car li ) )
      ( else
        ( rassoc el ( cdr li ) )
        )
      )
   )
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 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 )
'(one 1)
> ( rassoc '(3 3 3) al2 )
'(three 3 3 3)
> ( rassoc 1 al2 )
'()
>
```

Task 5: List of Strings to Strings

```
( define ( los->s li )
   ( cond
      ( ( = ( length li ) 0 ) "" )
      ( else
        ( cond
           ( ( = ( length li ) 1 )
             ( string-append ( car li ) "" ( los->s ( cdr li ) ) )
             )
           ( else
             ( string-append ( car li ) " " ( los->s ( cdr li ) ) )
           )
           )
        )
      )
   )
```



Task 6: Generate List

```
( define ( generate-list n func )
   ( cond
      ( ( = n 0 ) '() )
      ( else
        ( cons ( func ) ( generate-list ( - n 1 ) func ) )
        )
      )
   )
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( generate-list 10 roll-die )
'(3 2 2 1 1 1 3 4 3 4)
> ( generate-list 20 roll-die )
'(4 1 1 5 1 3 1 4 3 2 2 2 3 2 1 4 1 1 3 2)
> ( generate-list 12
                  ( lambda () ( list-ref '( red yellow blue ) ( random 3 ) ) )
                  )
'(blue blue blue red blue yellow blue yellow blue blue red blue)
>
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( define dots ( generate-list 3 dot ) )
> dots
```

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

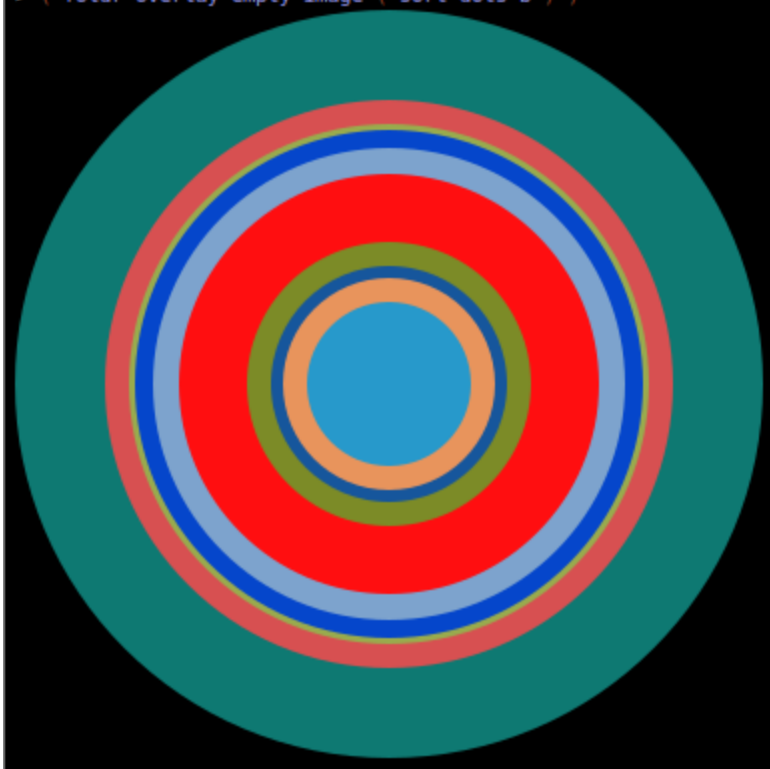
```
> ( sort-dots dots )
```

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

```
>
```

```
Language: racket, with debugging; memory limit: 128 MB.
> ( define a ( generate-list 5 big-dot ) )
> ( foldr overlay empty-image ( sort-dots a ) )
```

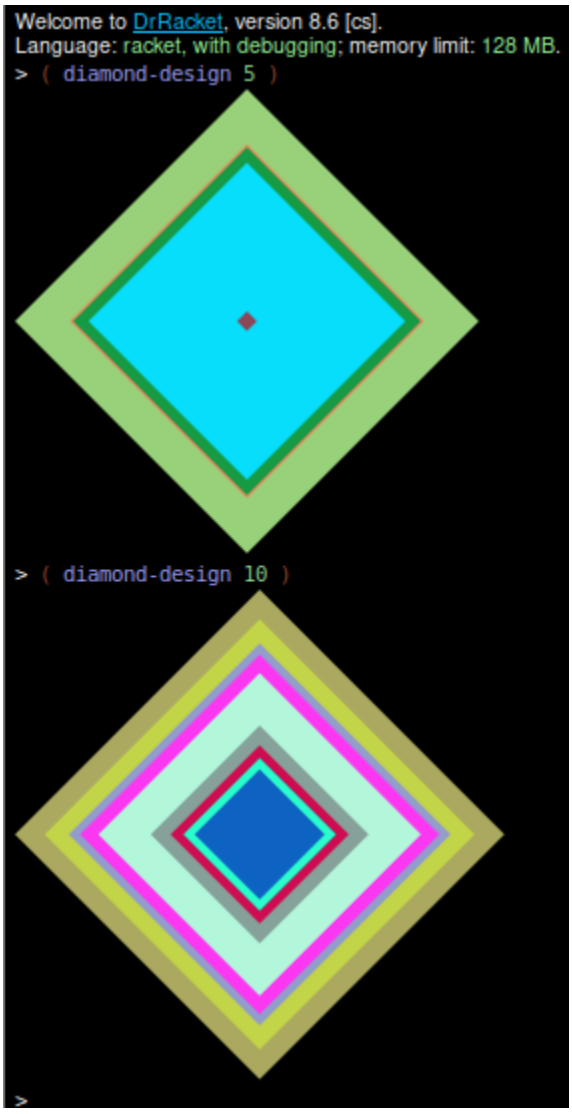
```
> ( define b ( generate-list 10 big-dot ) )
> ( foldr overlay empty-image ( sort-dots b ) )
```



```
>
```

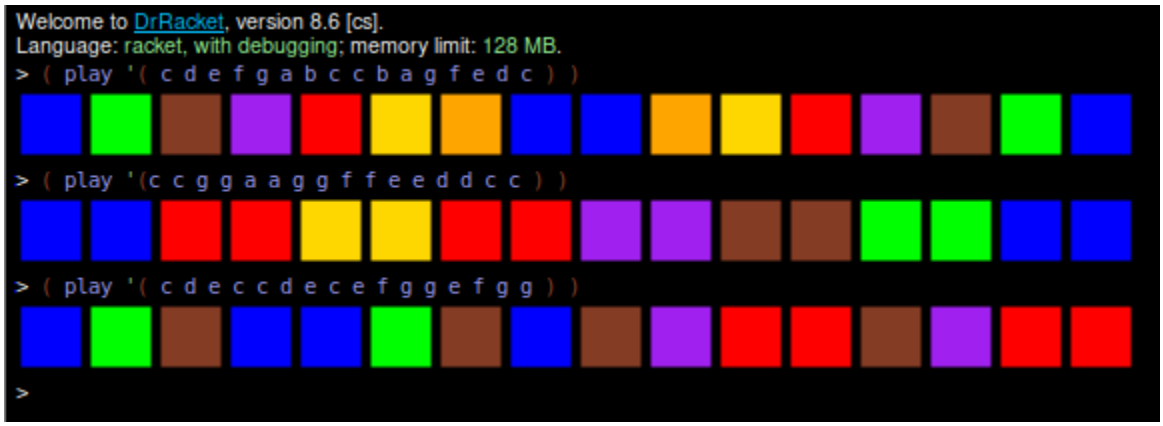
Task 7: The Diamond

```
( define ( diamond )
   ( rotate 45 ( square  ( random 201 ) 'solid ( random-color ) ))
   )

( define ( diamond-design n )
   ( define diamond-list ( generate-list n diamond ) )
   ( foldr overlay empty-image ( sort-dots diamond-list ) )
   )
```



Task 8: Chromesthetic Renderings

```
( define ( play  l )
   ( cond
      ( ( = ( length l ) 0 ) empty-image )
      ( else
              ( beside ( color->box ( pc->color ( car l ) ) ) ( play ( cdr l ) ) )
               )
      )
   )
```



Task 9: Diner

```
( define menu '( ( hotdog .  5.0 ) ( milkshake . 2.0 ) ( tea . 1.0 ) ( pancakes .
4.5 ) ( waffles . 3.5 ) ( french toast . 6.5 ) ) )

( define sales
'(tea
  milkshake
  hotdog
  tea
  tea
  pancakes
  tea
  pancakes
  hotdog
  waffles
  frenchtoast
  pancakes
  pancakes
  milkshake
  hotdog
  hotdog
  frenchtoast
  milkshake
  tea
  milkshake
  hotdog
  tea
  milkshake
  hotdog
  hotdog
  frenchtoast
  hotdog
  milkshake
  frenchtoast
  milkshake
  )
   )

( define ( price item )
   ( cond
      ( ( = ( length menu ) 0 ) '() )
      ( ( equal? item ( car ( car menu ) ) ) ( cdr ( car menu ) ) )
      ( else
        ( price item ( cdr menu ) )
        )
```

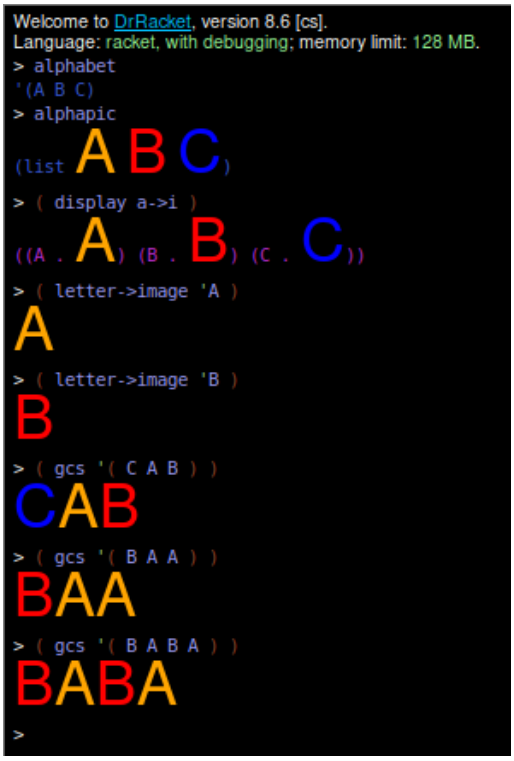
```
      )
   )

( define ( total sales item )
   ( define filtered-list ( filter ( lambda (el) (eq? el item) ) sales ) )
   ( define prices ( map price filtered-list ) )
   ( foldr + 0 prices )
   )
```

```
> menu
'((hotdog . 5.0) (milkshake . 2.0) (tea . 1.0) (pancakes . 4.5) (waffles . 3.5)
(french toast . 6.5))
> sales
'(tea
  milkshake
  hotdog
  tea
  tea
  pancakes
  tea
  pancakes
  hotdog
  waffles
  frenchtoast
  pancakes
  pancakes
  milkshake
  hotdog
  hotdog
  frenchtoast
  milkshake
  tea
  milkshake
  hotdog
  tea
  milkshake
  hotdog
  hotdog
  frenchtoast
  hotdog
  milkshake
  frenchtoast
  milkshake)
> ( total sales 'hotdog )
40.0
```

```
> ( total sales 'milkshake )
14.0
> ( total sales 'tea )
6.0
> ( total sales 'pancakes )
18.0
> ( total sales 'waffles )
3.5
> ( total sales 'frenchtoast )
26.0
```

Task 10: Grapheme Color Synesthesia

```
; https://docs.google.com/spreadsheets/d/1t6yZEi1T0rPkgFPpuhTKIbwDE9o4aDEb7KSxULVig0Q
; I used an excel script to automatically generate the colors

( define AI (text "A" 36 "orange") )
( define BI (text "B" 36 "red") )
( define CI (text "C" 36 "blue") )
( define DI (text "D" 36 ( color 214 194 199 )) )
( define EI (text "E" 36 ( color 61 79 133 )) )
( define FI (text "F" 36 ( color 81 167 248 )) )
( define GI (text "G" 36 ( color 80 183 67 )) )
( define HI (text "H" 36 ( color 121 32 29 )) )
( define II (text "I" 36 ( color 107 130 2 )) )
( define JI (text "J" 36 ( color 248 84 171 )) )
( define KI (text "K" 36 ( color 16 42 44 )) )
( define LI (text "L" 36 ( color 60 132 202 )) )
( define MI (text "M" 36 ( color 228 163 230 )) )
( define NI (text "N" 36 ( color 122 176 66 )) )
( define OI (text "O" 36 ( color 182 122 92 )) )
( define PI (text "P" 36 ( color 70 150 245 )) )
( define QI (text "Q" 36 ( color 28 182 88 )) )
( define RI (text "R" 36 ( color 185 61 106 )) )
( define SI (text "S" 36 ( color 94 85 62 )) )
( define TI (text "T" 36 ( color 254 30 131 )) )
( define UI (text "U" 36 ( color 243 121 109 )) )
( define VI (text "V" 36 ( color 38 128 93 )) )
( define WI (text "W" 36 ( color 109 151 21 )) )
( define XI (text "X" 36 ( color 106 27 152 )) )
( define YI (text "Y" 36 ( color 99 58 12 )) )
( define ZI (text "Z" 36 ( color 207 78 77 )) )

( 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->i ( a-list alphabet alphapic ) )

( define ( letter->image letter )
   ( cdr ( assoc letter a->i ) )
   )

( define ( gcs list )
   ( define i-list ( map letter->image list ) )
   ( foldr beside empty-image i-list )
   )
```


Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> alphabet
'(A B C)
> alphapic
(list A B C)
> (display a->i)
((A . A) (B . B) (C . C))
> (letter->image 'A)
A
> (letter->image 'B)
B
> (gcs '(C A B))
CAB
> (gcs '(B A A))
BAA
> (gcs '(B A B A))
BABA
>


Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> 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)
> alphapic
(list 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)
> (gcs '(A L P H A B E T))
ALPHABET
> (gcs '(D A N D E L I O N))
DANDELION
> (gcs '(Q U I C K))
QUICK
> (gcs '(H O M E O S T A S I S))
HOMEOSTASIS
> (gcs '(X Y L O P H O N E))
XYLOPHONE
> (gcs '(R A C K E T))
RACKET
> (gcs '(S Y N E S T H E S I A))
SYNESTHESIA
> (gcs '(O S W E G O))
OSWEGO
> (gcs '(O N T A R I O))
ONTARIO
> (gcs '(Y G G D R A S I L))
YGGDRASIL
>