Assignment: RLP HoFs

Abstract: This assignment tasks me to perform some recursive programming and to use higher order functions to complete some problems. Tasks 1 through 5 are done in a recursive manner while tasks 6 through 10 will utilize higher order functions.

Interaction: Task 1 Generate Uniform List

Demo:

(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))

>

Code:

```
( define ( generate-uniform-list num obj )
   ( cond
      ( ( = num 0 )
        '()
        )
      ( else
        ( cons obj ( generate-uniform-list ( - num 1 ) obj ) )
        )
      )
   )
```

Interaction: Task 2 Association list generator

Demo:

(a-list '(one two three four five) '(un duex trois quatre cinq))

'((one . un) (two . duex) (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))

Code:

```
( define ( a-list lo losl )
   ( cond
      ( ( not ( = ( length lo ) ( length losl ) ) )
        #f
        )
      ( ( and ( equal? lo null ) ( equal? losl null ) )
        '()
        )
      ( ( = ( length lo ) 1 )
        ( cons ( cons ( car lo ) ( car losl ) ) '() )
        )
      ( ( > ( length lo ) 1 )
        ( cons ( cons ( car lo ) ( car losl ) ) ( a-list ( cdr lo ) ( cdr losl ) ) )
        )
      )
   )
```

Interaction: Task 3 Assoc

Demo:

```
( 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) ) )
    )
> ( assoc 'two al1 )
'(two . deux)
> ( assoc 'five al1 )
'()
> ( assoc 'three al2 )
'(three 3 3 3)
> ( assoc 'four al2 )
'()
```

Code:

```
( define ( assoc lo al )
   ( define al-elem ( al-element al ) )
   ( cond
      ( ( equal? al null )
        '()
        )
      ( ( equal? lo al-elem )
        ( car al )
        )
      ( else
        ( assoc lo ( cdr al ) )
```

```
        )
      )
   )
( define ( al-element al )
   ( cond
      ( ( equal? al null )
        '()
        )
      ( else
        ( car ( car al ) )
        )
      )
   )
```

Interaction: Task 4 Rassoc

Demo:

```
( 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) ) )
    )
> ( rassoc 'three al1 )
'()
> ( rassoc 'trois al1 )
'(three . trois)
> ( rassoc '(1) al2 )
'(one 1)
> ( rassoc '(1) al2 )
'(one 1)
```

> (rassoc '(3 3 3) al2)

'(three 3 3 3)

> (rassoc 1 al2)

'()

>

Code:

```
( define ( rassoc lo al )
   ( define al-elem ( ral-element al ) )
   ( cond
      ( ( equal? al null )
        '()
        )
      ( ( equal? lo al-elem )
        ( car al )
        )
      ( else
        ( rassoc lo ( cdr al ) )
        )
      )
   )
( define ( ral-element al )
   ( cond
      ( ( equal? al null )
        '()
        )
      ( else
        ( cdr ( car al ) )
        )
      )
   )
```

Interaction: Task 5 Los->s

Demo:

```
> ( define li1
    '()
    )
> ( define li2
    '( "whatever" )
    )
> ( define li3
    '( "red" "yellow" "blue" "purple" )
    )
> ( define li4
    ( generate-uniform-list 20 "-" )
    )
> ( los->s li1 )
""
> ( los->s li2 )
"whatever"
> ( los->s li3 )
"red yellow blue purple"
> ( los->s li4 )
"--------------------"
```

Code:

```
( define ( los->s li )
   ( cond
      ( ( equal? li null )
        '""
        )
      ( ( singleton? li )
```

```
        ( car li )
        )
      ( else
        ( string-append ( car li ) " " ( los->s ( cdr li ) ) )
        )
      )
  )
```

Interaction: Task 6 Generate list

Demo:

> (generate-list 10 roll-die)

'(1 4 6 4 2 3 2 2 3 2)

> (generate-list 20 roll-die)

'(1 3 3 1 1 3 1 1 2 4 3 3 1 1 5 5 2 2 4 6)

> (generate-list 12 (lambda () (list-ref '(red yellow blue) (random 3))))

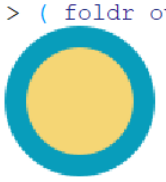
'(blue blue yellow yellow blue yellow red blue blue red yellow red)

> (define b (generate-list 10 big-dot))

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

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

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



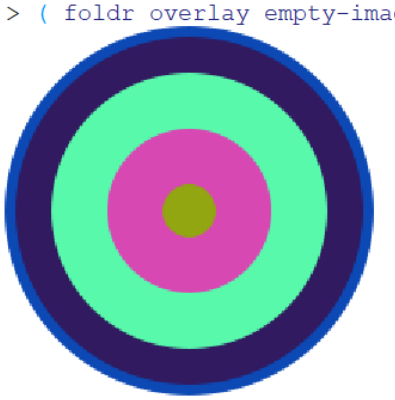
```
> ( sort-dots dots )
```

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

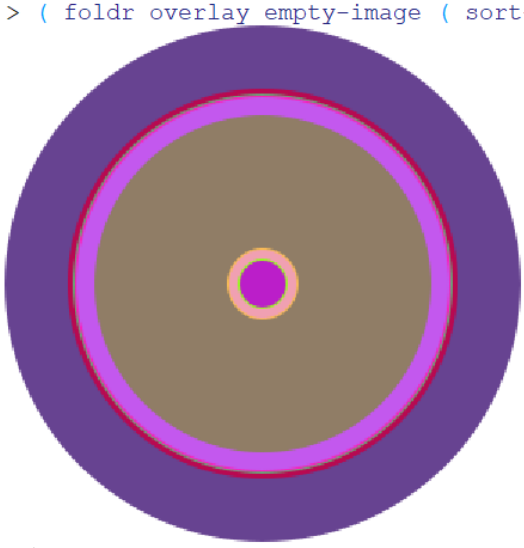
```
>
```

>

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



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



```
> |
```

Code:

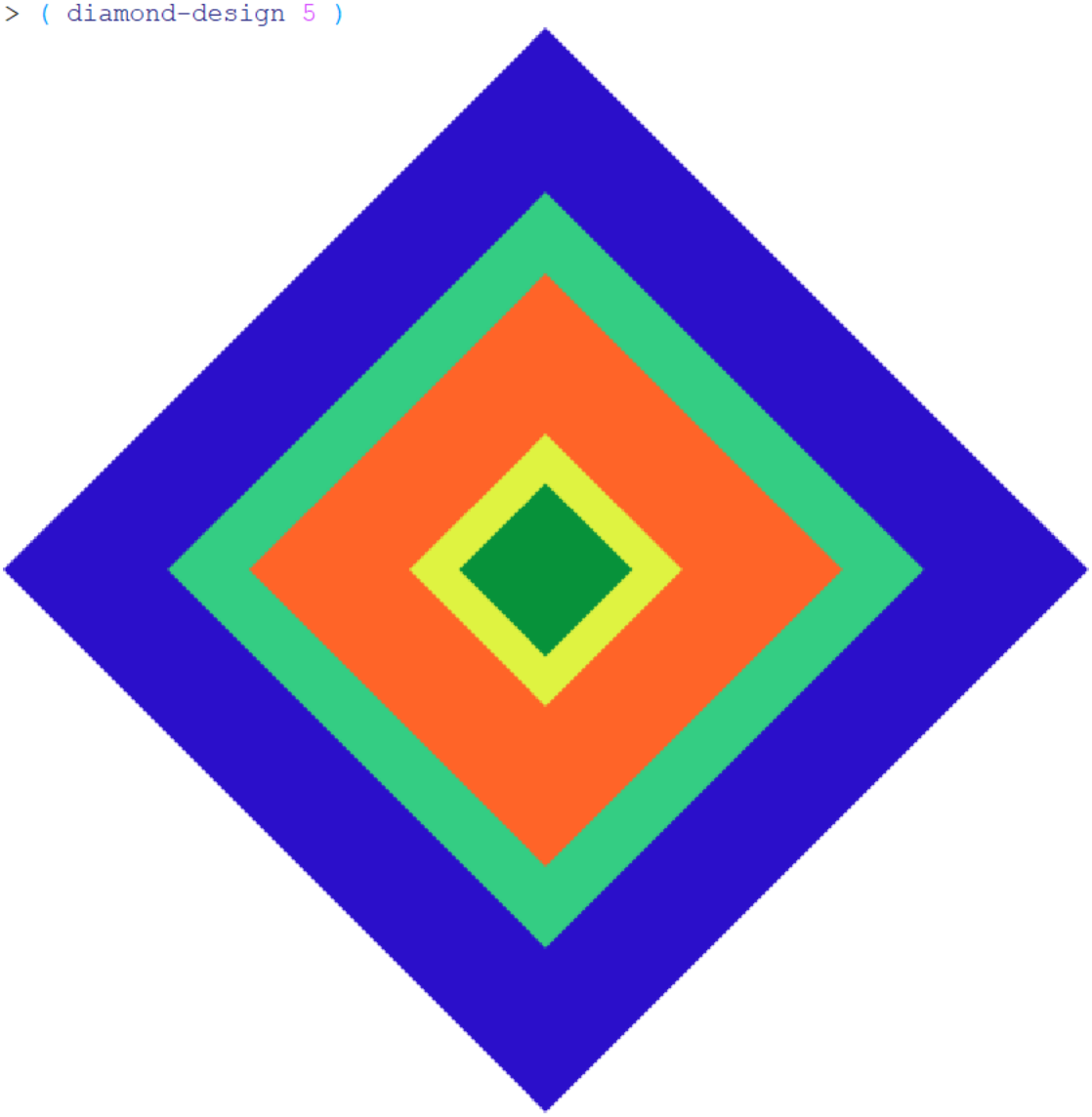
```
( define ( generate-list num fun )
   ( cond
      ( ( = num 0 )
        '()
        )
      ( else
        ( cons ( fun ) ( generate-list ( - num 1 ) fun ) )
        )
      )
   )
```

```
( define ( roll-die ) ( + ( random 6 ) 1 ) )

( define ( big-dot )
   ( circle ( + 10 ( random 150 ) ) "solid" ( random-color ) )
   )

( 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 < )
   )
```

Interaction: Task 7 The Diamond

Demo:

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (diamond-design 5)



>

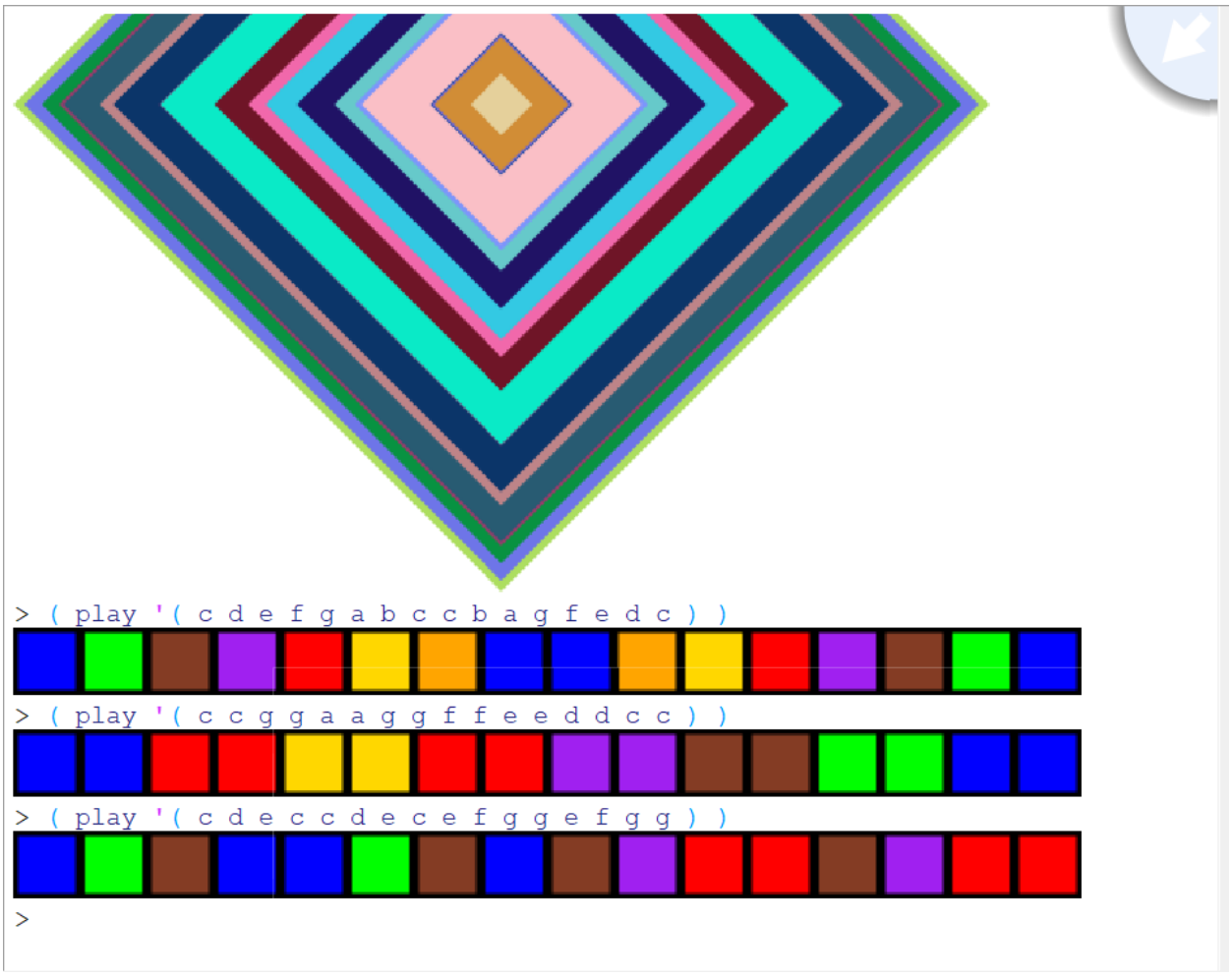
Code:

```
( define ( diamond-design num )
   ( define dia ( generate-list num diamond ) )
   ( foldr overlay empty-image ( sort-dots dia ) )
   )
( define ( generate-list num fun )
   ( cond
      ( ( = num 0 )
        '()
        )
```

```
      ( else
        ( cons ( fun ) ( generate-list ( - num 1 ) fun ) )
        )
      )
   )
( define ( diamond )
   ( rhombus ( + 20 ( random 381 ) ) 90 "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 < )
   )
```

Interaction: Task 8 Chromesthetic renderings

Demo:



Code:

```
( define ( play pl )
   ( define nl ( map car pc-a-list ) )
   ( define bl ( map cdr cb-a-list ) )
   ( define image-list ( gen-pbl nl bl ) )
   ( gen-image pl image-list )
   ( foldr beside empty-image ( gen-image pl image-list ) )
   )


( 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 ( gen-pbl nl bl )
   ( cond
      ( ( equal? nl null )
        '()
        )
      ( ( singleton? nl )
        ( list ( cons ( car nl ) ( car bl ) ) )
        )
      ( else
        ( cons ( cons ( car nl ) ( car bl ) ) ( gen-pbl ( cdr nl ) ( cdr bl ) ) )
        )
      )
   )

( define ( gen-image notelist imagelist )
   ( cond
      ( ( equal? notelist null )
        empty-image
        )
      ( ( singleton? notelist )
        ( list ( cdr ( assoc ( car notelist ) imagelist ) ) )
        )
      ( else
        ( cons ( cdr ( assoc ( car notelist ) imagelist ) ) ( gen-image ( cdr notelist ) imagelist ))
        )
      )
   )
```

Interaction: Task 9 Diner

Demo:

```
> ( menu-list )
'((water . 1.0)
  (cookie . 0.5)
  (salad . 3.95)
  (small-meatball-sub . 9.95)
  (medium-meatball-sub . 11.95)
  (large-meatball-sub . 13.95))
> ( define sales ( sales-list ) )
> sales
'(water
  water
  cookie
  water
  water
  cookie
  salad
  salad
  small-meatball-sub
  small-meatball-sub
  cookie
  small-meatball-sub
  cookie
  small-meatball-sub
  medium-meatball-sub
  large-meatball-sub
  large-meatball-sub
```

```
 large-meatball-sub
 medium-meatball-sub
 cookie
 cookie
 cookie
 water
 small-meatball-sub
 water
 small-meatball-sub
 water
 large-meatball-sub
 medium-meatball-sub
 large-meatball-sub
 medium-meatball-sub)
> ( total sales 'water )
7.0
> ( total sales 'cookie )
3.5
> ( total sales 'salad )
7.9
> ( total sales 'small-meatball-sub )
59.7
> ( total sales 'medium-meatball-sub )
47.8
> ( total sales 'large-meatball-sub )
69.75
>
```

Code:

```
( define ( menu-list )
```

'((water . 1.00) (cookie . 0.50) (salad . 3.95) (small-meatball-sub . 9.95) (medium-meatball-sub . 11.95) (large-meatball-sub . 13.95))

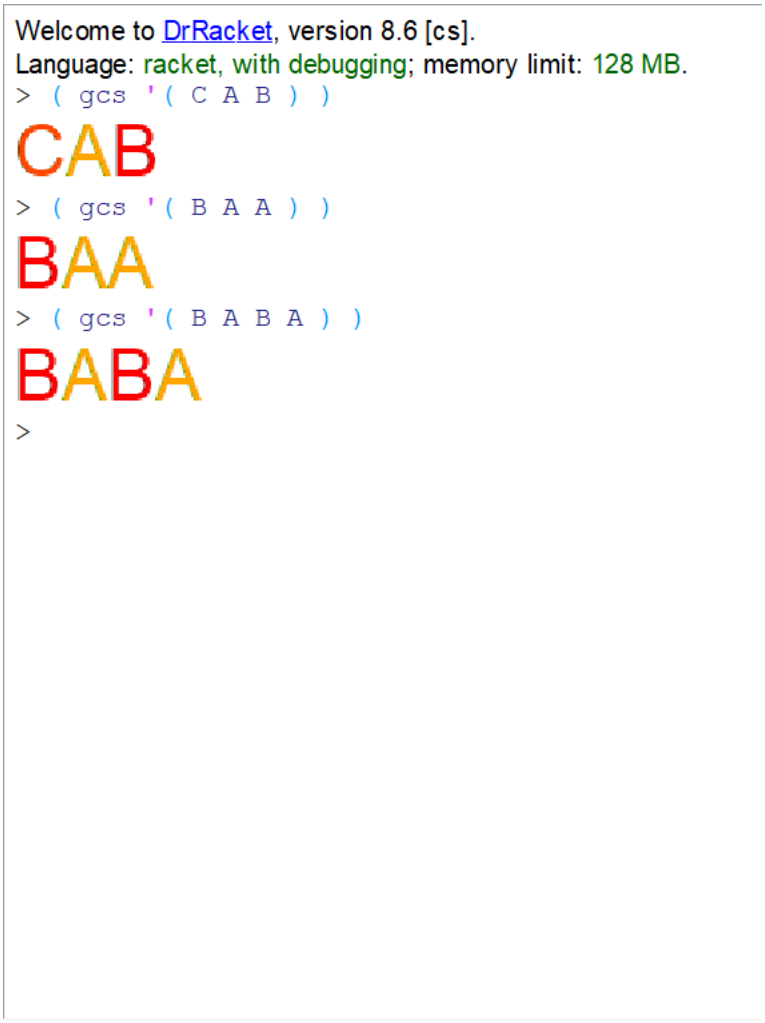
)

(define (sales-list)

```
   '( water water cookie water water cookie salad salad small-
meatball-sub small-meatball-sub cookie small-meatball-sub cookie
small-meatball-sub medium-meatball-sub large-meatball-sub large-
meatball-sub large-meatball-sub medium-meatball-sub cookie cookie
cookie water small-meatball-sub water small-meatball-sub water large-
meatball-sub medium-meatball-sub large-meatball-sub medium-meatball-
sub )
   )


( define ( total sales-list item )
   ( define menu ( menu-list ) )
   ( define  test ( filter ( lambda ( x ) ( equal? x item ) ) sales-
list ) )
   ( define price-list ( map ( lambda ( x ) ( price x menu ) ) test )
)
   ( define total-price ( foldr + 0 price-list ) )
   total-price
   )


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

Interaction: Task 10 Grapheme Color Synesthesia

Demo1:


Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (gcs '(C A B))
CAB
> (gcs '(B A A))
BAA
> (gcs '(B A B A))
BABA
>

Demo2:

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.

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

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

```
> ( gcs '( H E L L O ) )
```

```
> ( gcs '( P R O G R A M M I N G ) )
```

```
> ( gcs '( S U N Y O S W E G O ) )
```

```
> ( gcs '( R A D I O ) )
```

```
> ( gcs '( R O B O T ) )
```

```
> ( gcs '( R A C K E T ) )
```

```
> ( gcs '( A R T I F I C I A L ) )
```

```
> ( gcs '( I N T E L L I G E N C E ) )
```

```
>
```

Code:

```
( define AI ( text "A" 36 "orange" ) )
( define BI ( text "B" 36 "red" ) )
( define CI ( text "C" 36 "orange red" ) )
( define DI ( text "D" 36 "dark red" ) )
( define EI ( text "E" 36 "firebrick" ) )
( define FI ( text "F" 36 "deep pink" ) )
```

```
( define GI ( text "G" 36 "indian red" ) )
( define HI ( text "H" 36 "blue" ) )
( define II ( text "I" 36 "violet red" ) )
( define JI ( text "J" 36 "hot pink" ) )
( define KI ( text "K" 36 "light pink" ) )
( define LI ( text "L" 36 "pink" ) )
( define MI ( text "M" 36 "lavender blush" ) )
( define NI ( text "N" 36 "chocolate" ) )
( define OI ( text "O" 36 "brown" ) )
( define PI ( text "P" 36 "coral" ) )
( define QI ( text "Q" 36 "peru" ) )
( define RI ( text "R" 36 "goldenrod" ) )
( define SI ( text "S" 36 "light salmon" ) )
( define TI ( text "T" 36 "gold" ) )
( define UI ( text "U" 36 "olive" ) )
( define VI ( text "V" 36 "tan" ) )
( define WI ( text "W" 36 "peach puff" ) )
( define XI ( text "X" 36 "dark khaki" ) )
( define YI ( text "Y" 36 "wheat" ) )
( define ZI ( text "Z" 36 "pale goldenrod" ) )


( 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 li ( map letter->image list ) )
   ( foldr beside empty-image li )
   )
```