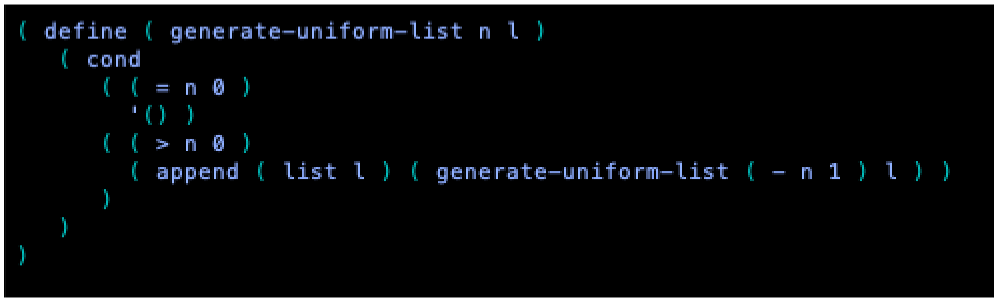
Fourth Racket Programming Assignment Solution

Learning Abstract:

In the first task of this Racket assignment, I learned how to do some basic list processing in Racket to get a better understanding of how Lisp functions such as car, cdr, append, etc., work. In the remaining tasks, I learned how to do list processing with higher order functions. The last task I was able to create a function of my own design.

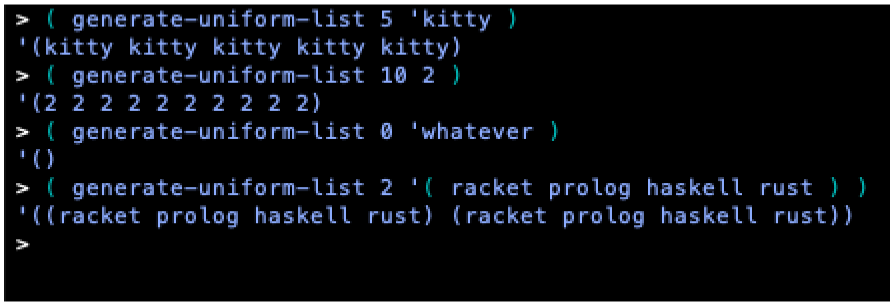
Task 1 - Generate Uniform List

Code:



```
( define ( generate-uniform-list n l )
   ( cond
      ( ( = n 0 )
        '() )
      ( ( > n 0 )
        ( append ( list l ) ( generate-uniform-list ( - n 1 ) l ) )
      )
   )
)
```

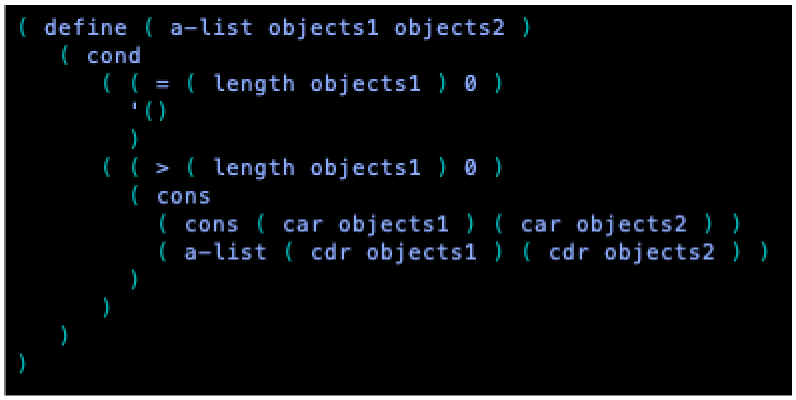
Demo:



```
> ( 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))
>
```

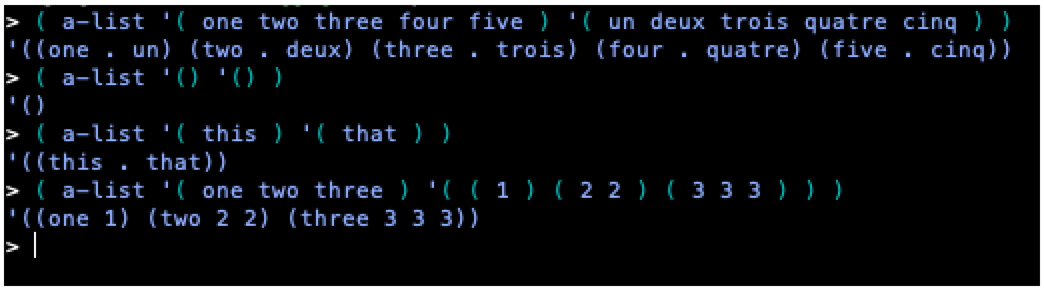
Task 2 - Association List Generator

Code:



```
( define ( a-list objects1 objects2 )
   ( cond
      ( ( = ( length objects1 ) 0 )
        '()
        )
      ( ( > ( length objects1 ) 0 )
        ( cons
          ( cons ( car objects1 ) ( car objects2 ) )
          ( a-list ( cdr objects1 ) ( cdr objects2 ) )
        )
      )
   )
)
```

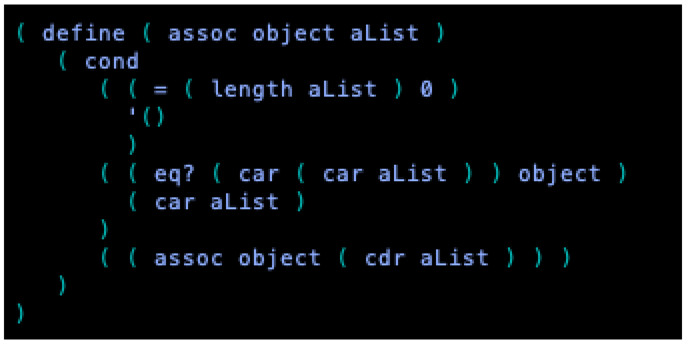
Demo:



```
> ( 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))
> |
```

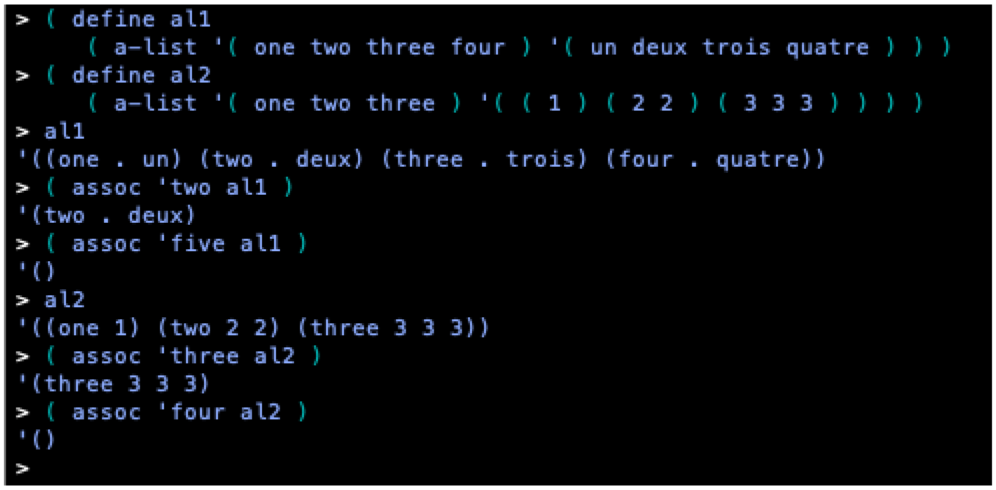
Task 3 - Assoc

Code:



```
( define ( assoc object aList )
   ( cond
      ( ( = ( length aList ) 0 )
        '()
        )
      ( ( eq? ( car ( car aList ) ) object )
        ( car aList )
      )
      ( ( assoc object ( cdr aList ) ) )
   )
)
```

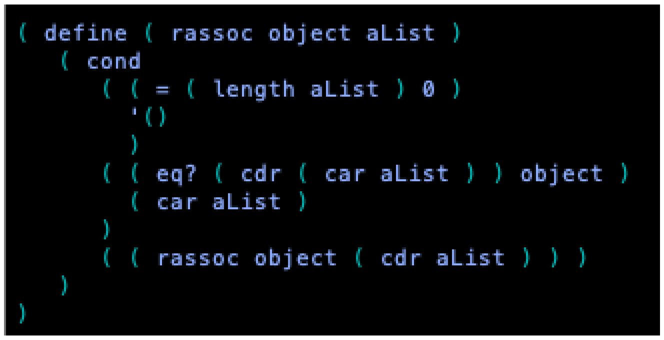
Demo:



```
> ( 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 ) ) ) )
> 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 )
'()
>
```

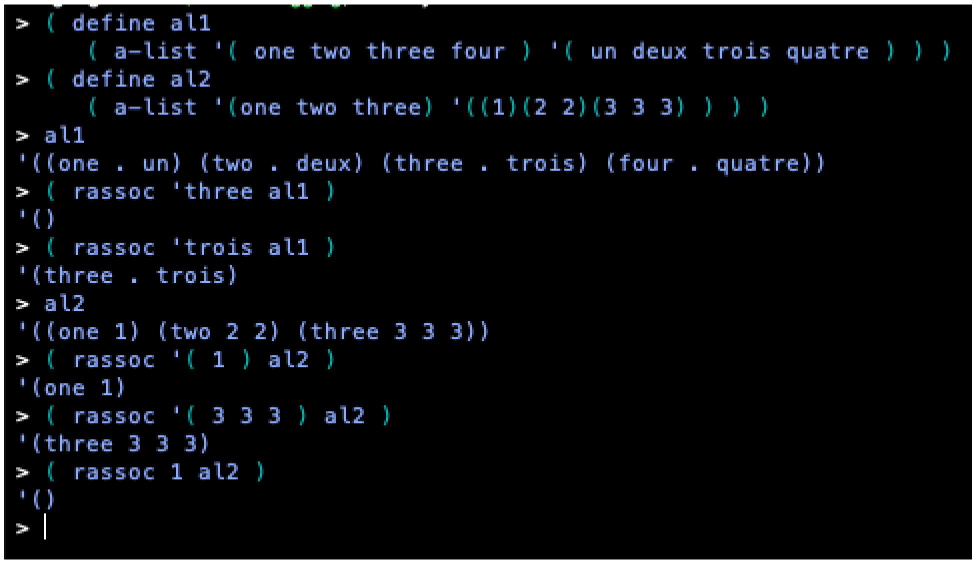
Task 4 - Rassoc

Code:



```
( define ( rassoc object aList )
   ( cond
      ( ( = ( length aList ) 0 )
        '()
        )
      ( ( eq? ( cdr ( car aList ) ) object )
        ( car aList )
      )
      ( ( rassoc object ( cdr aList ) ) )
   )
)
```

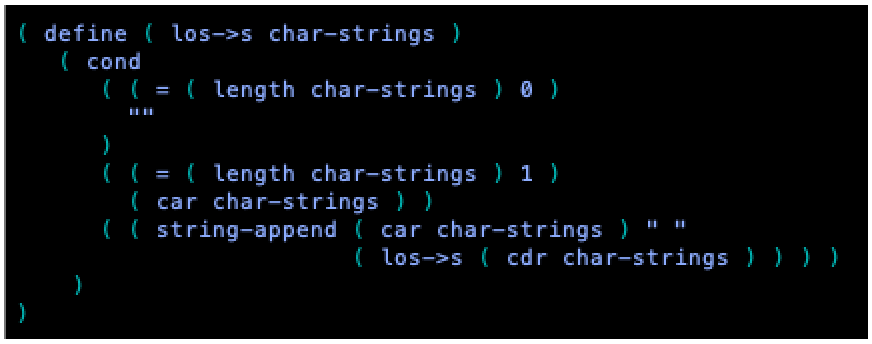
Demo:



```
> ( 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) ) ) )
> 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 )
'()
> 
```

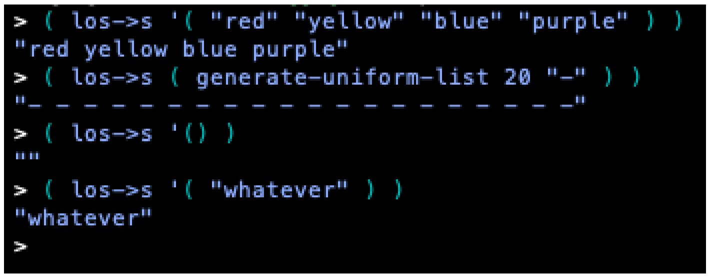
Task 5 - Los->

Code:



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

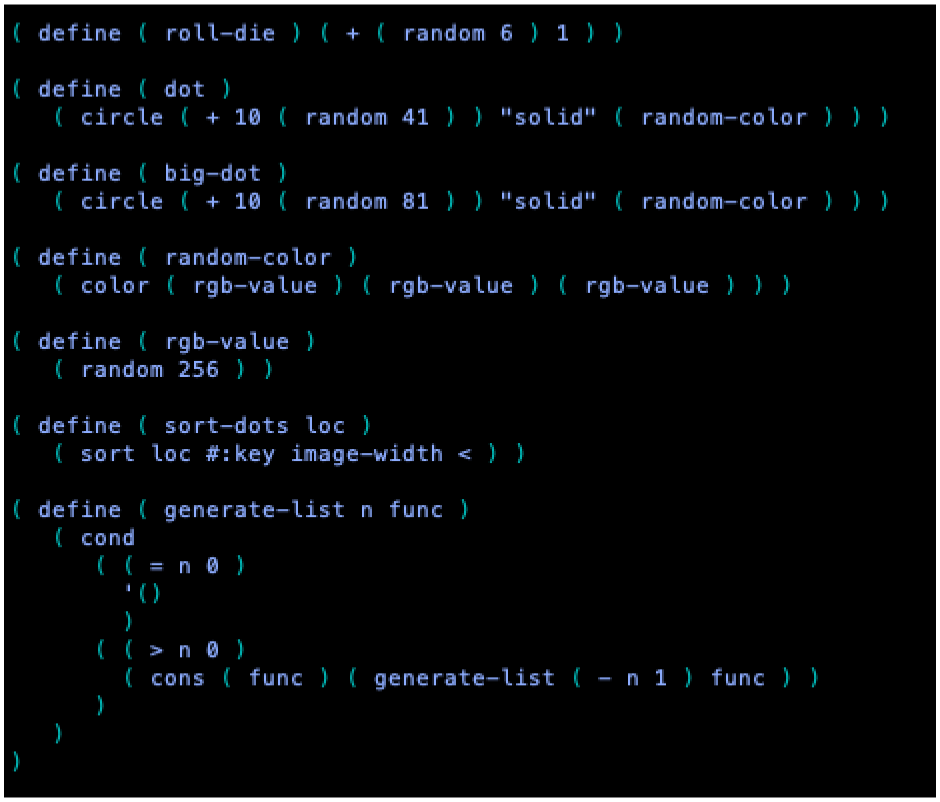
Demo:



```
> ( los->s '( "red" "yellow" "blue" "purple" ) )
"red yellow blue purple"
> ( los->s ( generate-uniform-list 20 "-" ) )
"- - - - - - - - - - - - - - - - - - - -"
> ( los->s '() )
""
> ( los->s '( "whatever" ) )
"whatever"
>
```

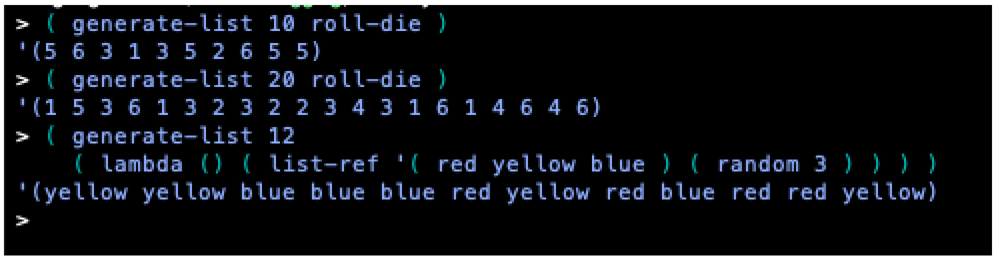
Task 6 - Generate list

Code:



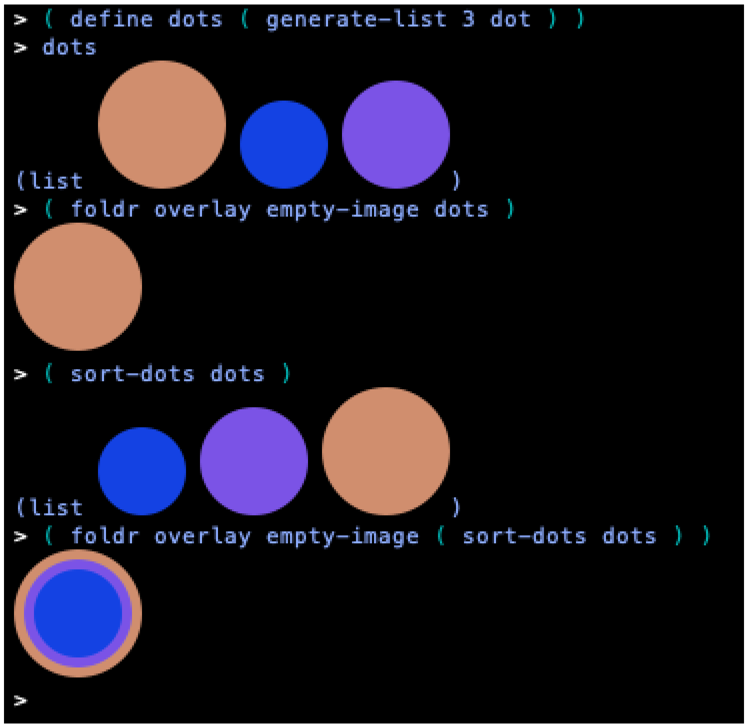
```
( define ( roll-die ) ( + ( random 6 ) 1 ) )

( define ( dot )
   ( circle ( + 10 ( random 41 ) ) "solid" ( random-color ) ) )

( define ( big-dot )
   ( circle ( + 10 ( random 81 ) ) "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 < ) )

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

Demo1:



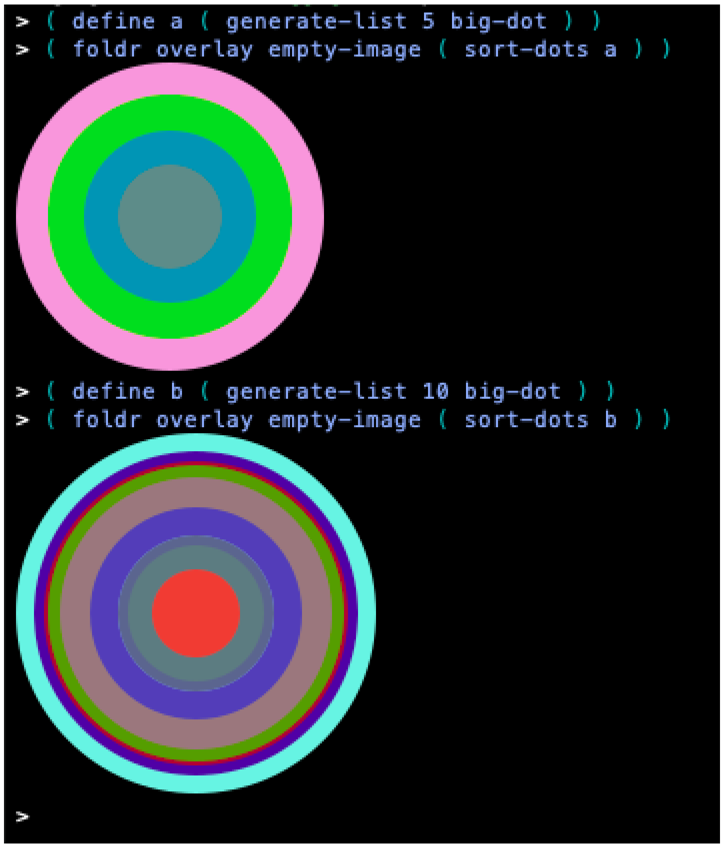
```
> ( generate-list 10 roll-die )
'(5 6 3 1 3 5 2 6 5 5)
> ( generate-list 20 roll-die )
'(1 5 3 6 1 3 2 3 2 2 3 4 3 1 6 1 4 6 4 6)
> ( generate-list 12
    ( lambda () ( list-ref '( red yellow blue ) ( random 3 ) ) ) )
'(yellow yellow blue blue blue red yellow red blue red red yellow)
>
```

Demo2:



```
> ( define dots ( generate-list 3 dot ) )
> dots
(list                    )
> ( foldr overlay empty-image dots )
> ( sort-dots dots )
(list                    )
> ( foldr overlay empty-image ( sort-dots dots ) )
>
```

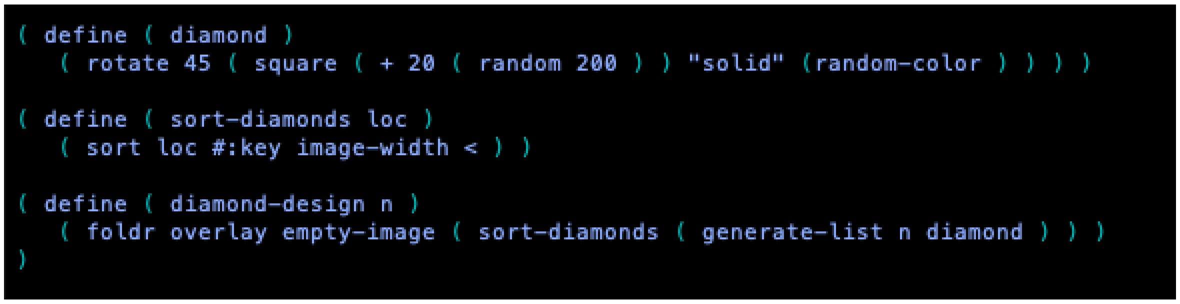
Demo3:



```
> ( 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 ) )
>
```

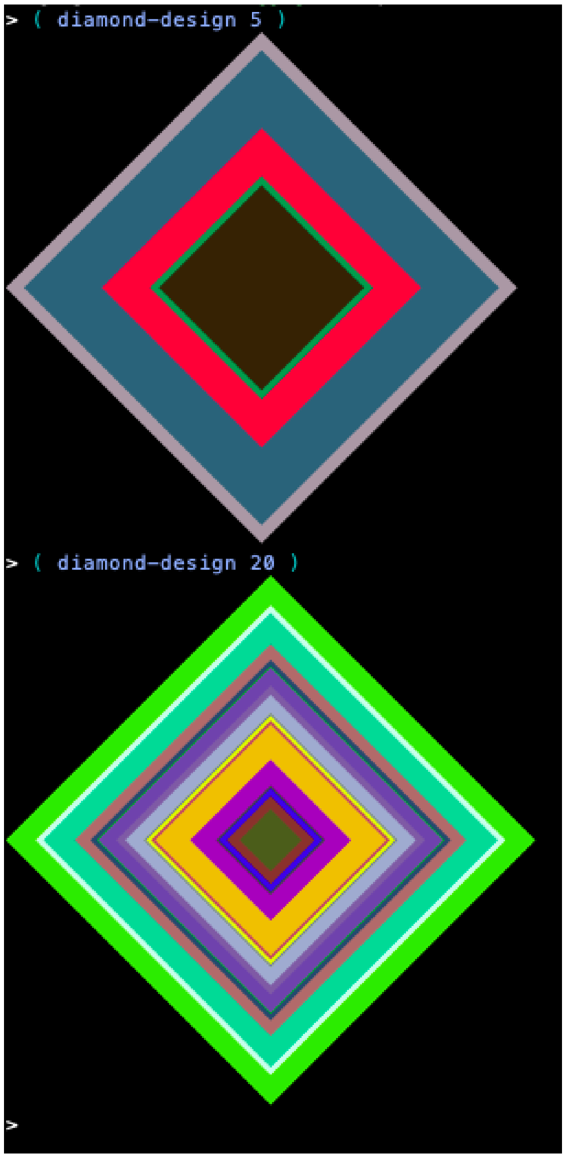
Task 7 - The Diamond

Code:



```
( define ( diamond )
   ( rotate 45 ( square ( + 20 ( random 200 ) ) "solid" (random-color ) ) ) )

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

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

Demo:



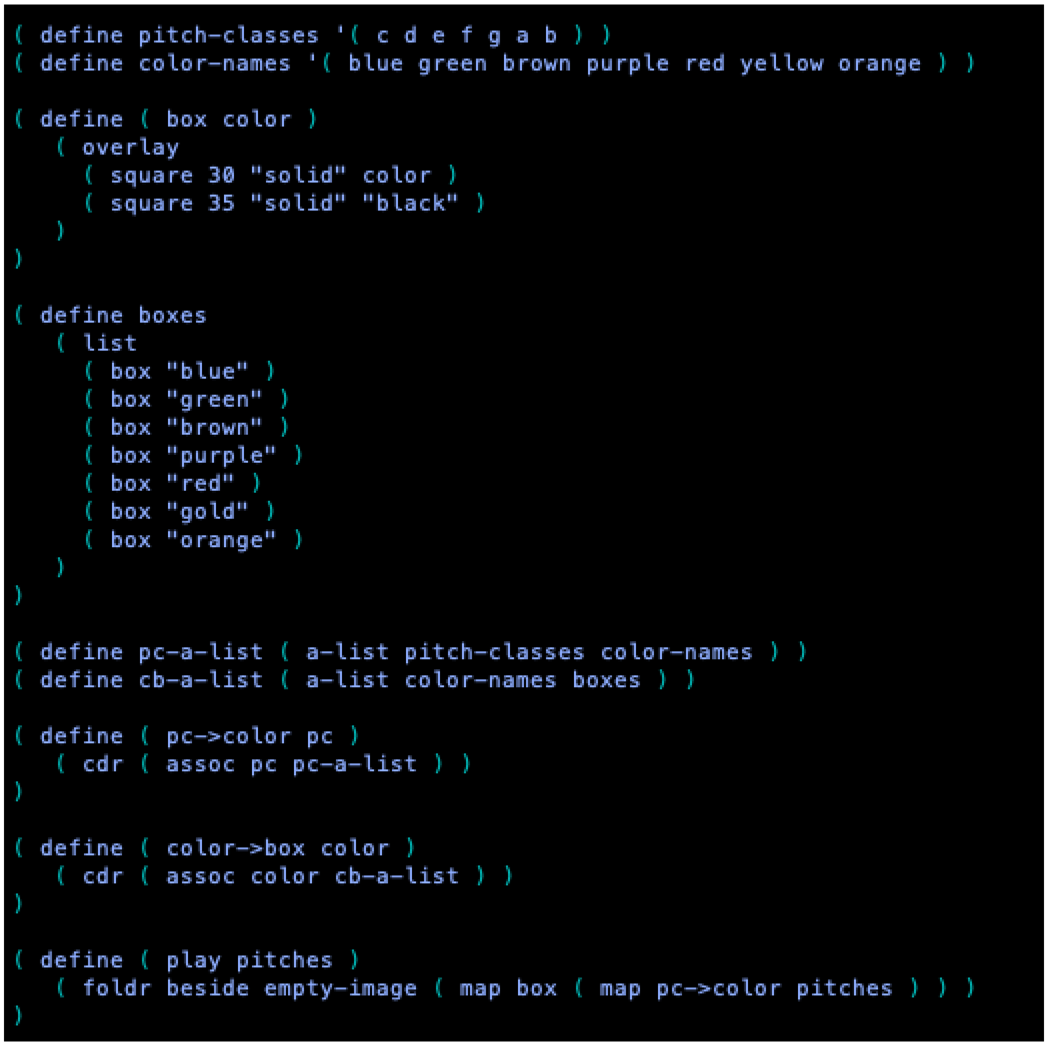
```
> ( diamond-design 5 )
```

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

```
>
```

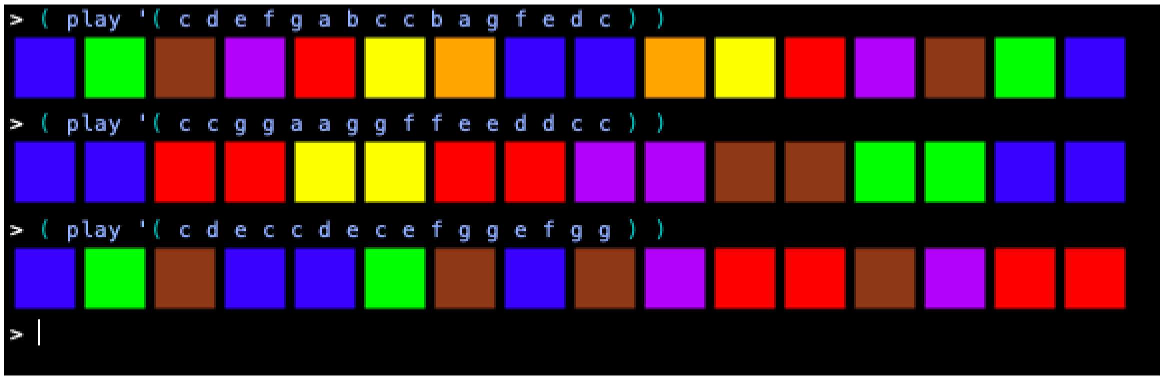
Task 8 - Chromesthetic renderings

Code:



```
( 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 pitches )
   ( foldr beside empty-image ( map box ( map pc->color pitches ) ) )
)
```

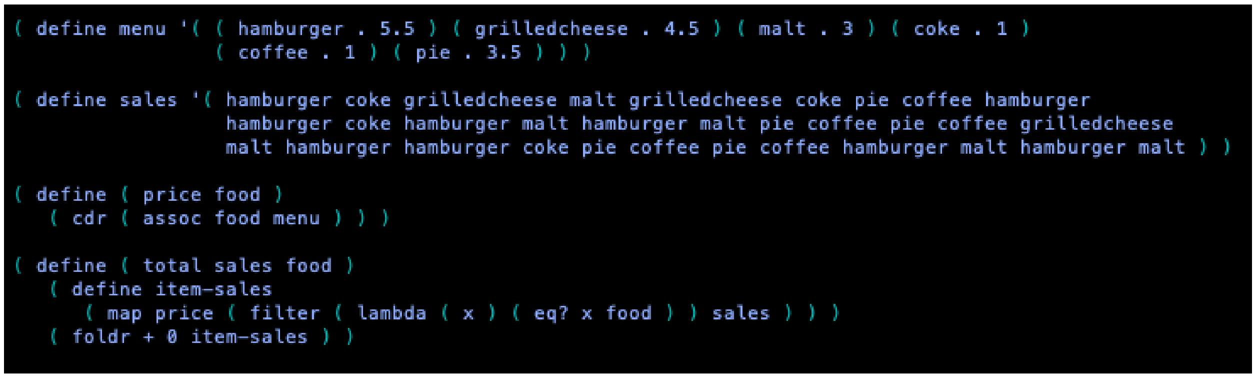
Demo:



```
> ( 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 ) )
> |
```

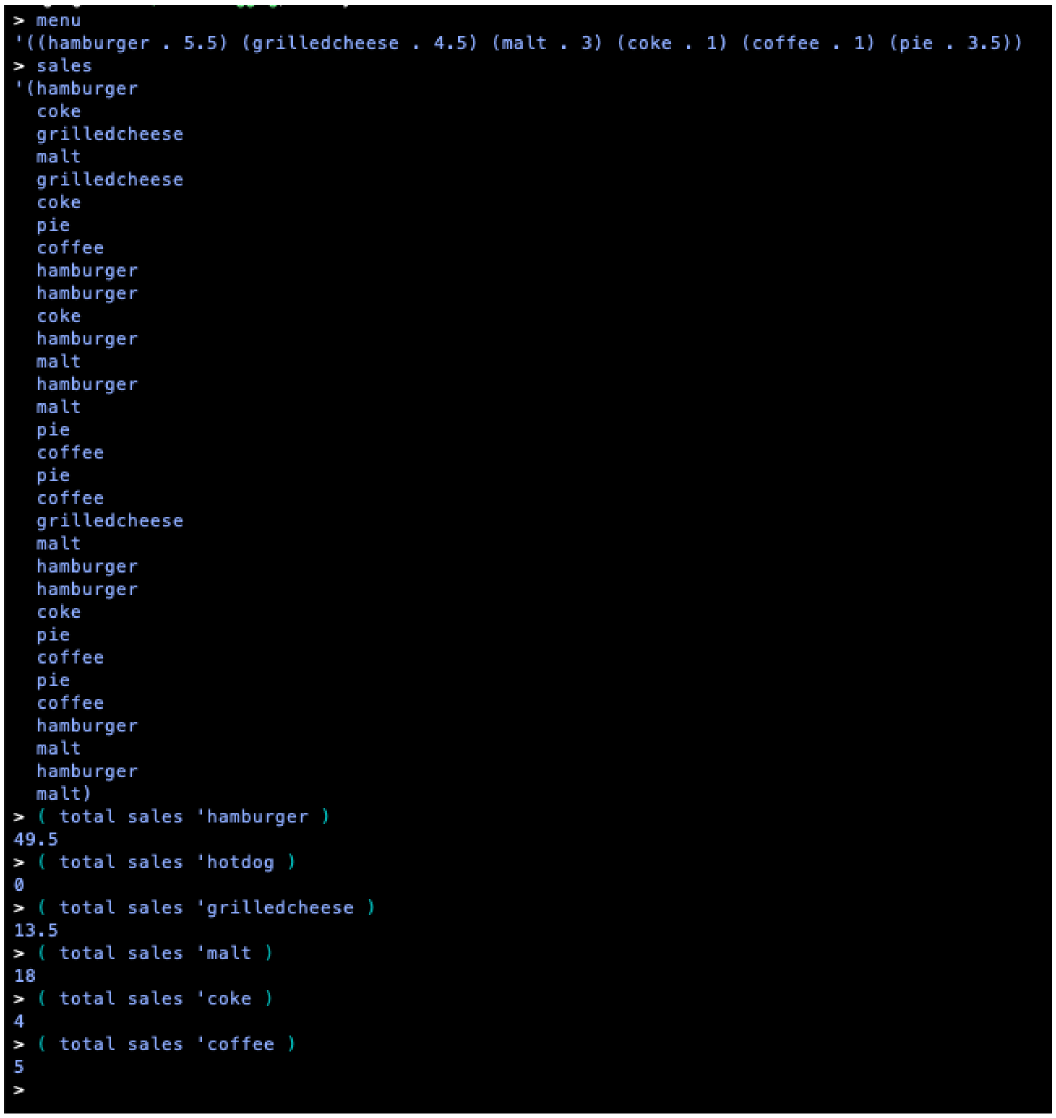
Task 9 - Diner

Code:



```
( define menu '( ( hamburger . 5.5 ) ( grilledcheese . 4.5 ) ( malt . 3 ) ( coke . 1 )
                 ( coffee . 1 ) ( pie . 3.5 ) ) )

( define sales '( hamburger coke grilledcheese malt grilledcheese coke pie coffee hamburger
                  hamburger coke hamburger malt hamburger malt pie coffee pie coffee grilledcheese
                  malt hamburger hamburger coke pie coffee pie coffee hamburger malt hamburger malt ) )

( define ( price food )
   ( cdr ( assoc food menu ) ) )

( define ( total sales food )
   ( define item-sales
      ( map price ( filter ( lambda ( x ) ( eq? x food ) ) sales ) ) )
   ( foldr + 0 item-sales ) )
```

Demo:



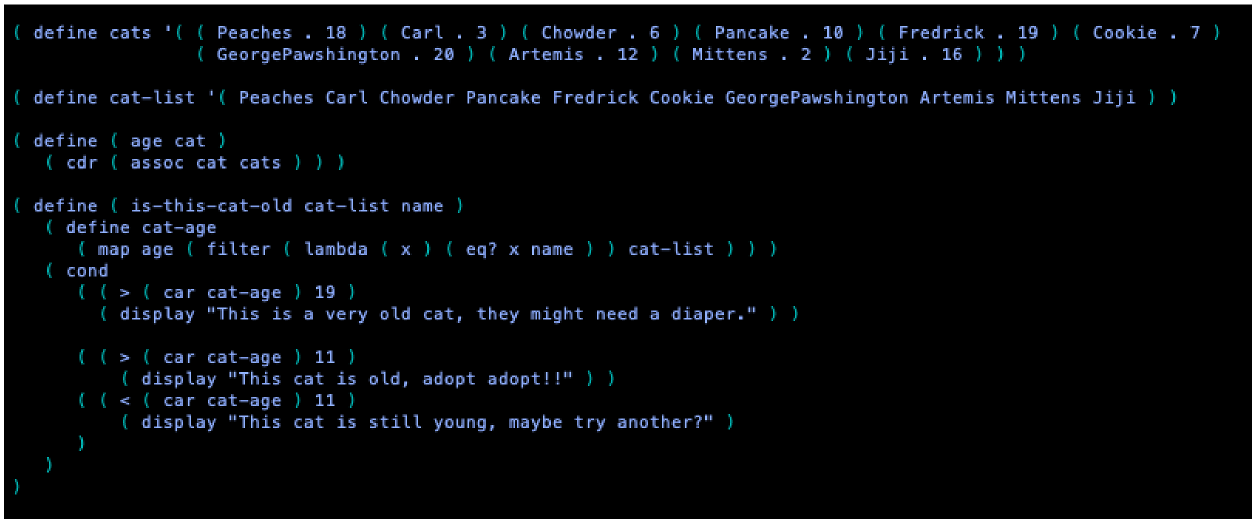
```
> menu
'((hamburger . 5.5) (grilledcheese . 4.5) (malt . 3) (coke . 1) (coffee . 1) (pie . 3.5))
> sales
'(hamburger
  coke
  grilledcheese
  malt
  grilledcheese
  coke
  pie
  coffee
  hamburger
  hamburger
  coke
  hamburger
  malt
  hamburger
  malt
  pie
  coffee
  pie
  coffee
  grilledcheese
  malt
  hamburger
  hamburger
  coke
  pie
  coffee
  pie
  coffee
  hamburger
  malt
  hamburger
  malt)
> ( total sales 'hamburger )
49.5
> ( total sales 'hotdog )
0
> ( total sales 'grilledcheese )
13.5
> ( total sales 'malt )
18
> ( total sales 'coke )
4
> ( total sales 'coffee )
5
>
```

Task 10 - Wild Card

You own a cat shelter that houses 10 cats currently. Many people who come into your shelter are looking to adopt senior cats so that they can give them a nice place to live as they grow old. There is a directory that contains the names of all the cats and how old they are (in pet years). In this scenario, a cat is considered a senior when it is between the ages of 11 and 20. The function called **is-this-cat-old** will have:

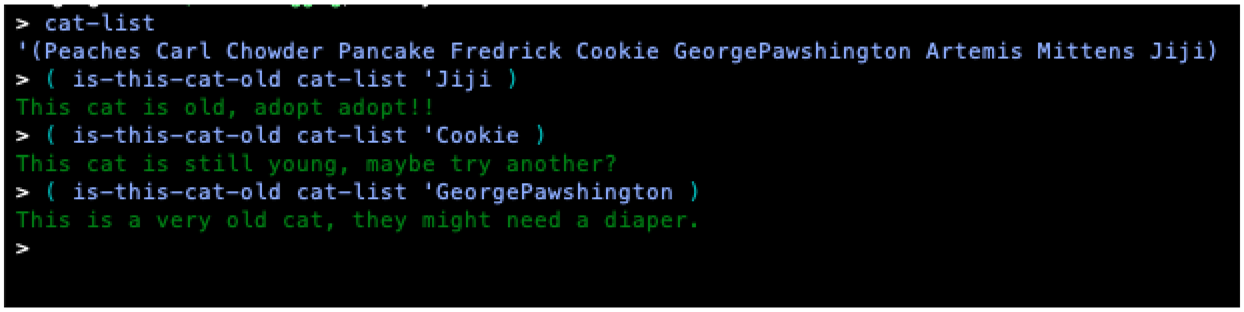
1. The first parameter is the list of cats at the shelter.
2. The second parameter is the name of the cat you want to adopt.
3. The result will be a message stating whether the cat is old or young.

Code:



```
( define cats '( ( Peaches . 18 ) ( Carl . 3 ) ( Chowder . 6 ) ( Pancake . 10 ) ( Fredrick . 19 ) ( Cookie . 7 )
                 ( GeorgePawshington . 20 ) ( Artemis . 12 ) ( Mittens . 2 ) ( Jiji . 16 ) ) )

( define cat-list '( Peaches Carl Chowder Pancake Fredrick Cookie GeorgePawshington Artemis Mittens Jiji ) )

( define ( age cat )
   ( cdr ( assoc cat cats ) ) )

( define ( is-this-cat-old cat-list name )
   ( define cat-age
      ( map age ( filter ( lambda ( x ) ( eq? x name ) ) cat-list ) ) )
   ( cond
      ( ( > ( car cat-age ) 19 )
        ( display "This is a very old cat, they might need a diaper." ) )

      ( ( > ( car cat-age ) 11 )
          ( display "This cat is old, adopt adopt!!" ) )
      ( ( < ( car cat-age ) 11 )
          ( display "This cat is still young, maybe try another?" )
      )
   )
)
```

Demo:



```
> cat-list
'(Peaches Carl Chowder Pancake Fredrick Cookie GeorgePawshington Artemis Mittens Jiji)
> ( is-this-cat-old cat-list 'Jiji )
This cat is old, adopt adopt!!
> ( is-this-cat-old cat-list 'Cookie )
This cat is still young, maybe try another?
> ( is-this-cat-old cat-list 'GeorgePawshington )
This is a very old cat, they might need a diaper.
>
```