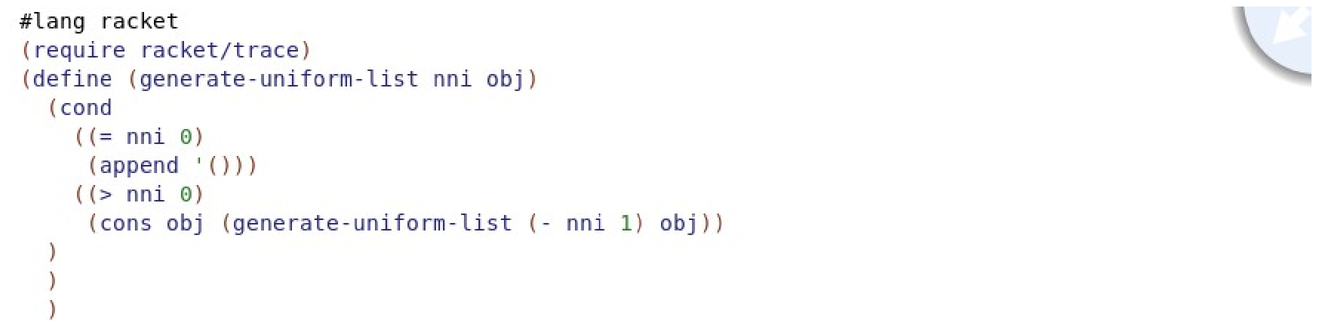
Fourth Racket Programming Assignment

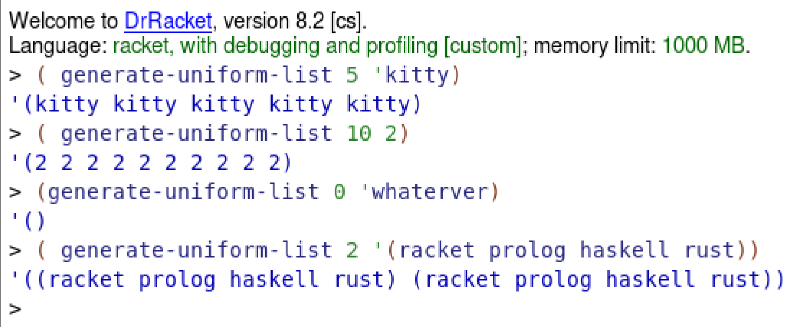
Abstract: In this assignment we developed some of the building block functions that racket has like assoc, a-lost , and a to string method. We also used recursion and map , filter , foldr. This was done in order to experience making programs with higher order functions. These are functions that use other functions as arguments.

Task 1 - Generate Uniform List



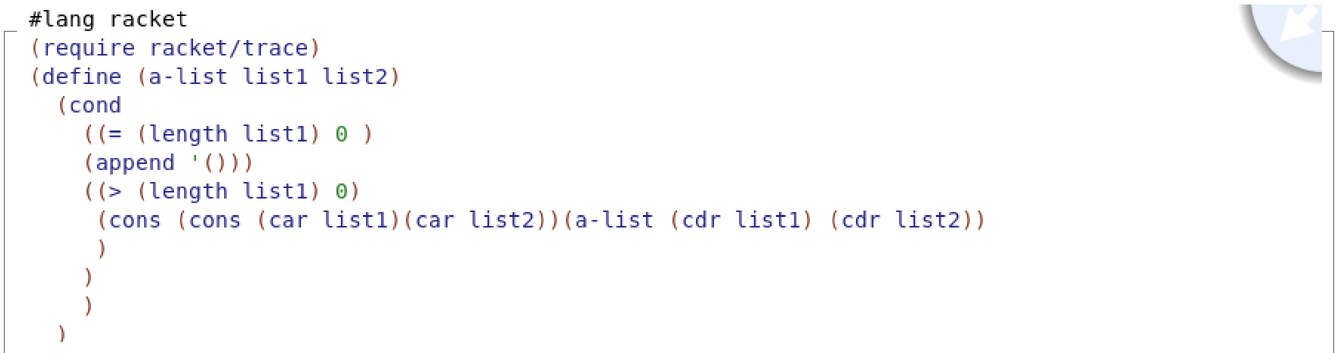
```
#lang racket
(require racket/trace)
(define (generate-uniform-list nni obj)
  (cond
    ((= nni 0)
     (append '()))
    ((> nni 0)
     (cons obj (generate-uniform-list (- nni 1) obj)))
  )
  )
  )
```

Demo



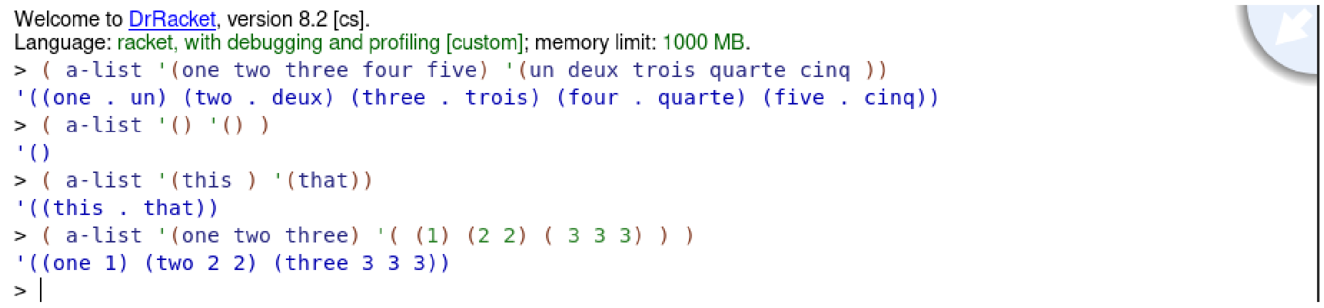
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> ( generate-uniform-list 5 'kitty)
'(kitty kitty kitty kitty kitty)
> ( generate-uniform-list 10 2)
'(2 2 2 2 2 2 2 2 2 2)
> (generate-uniform-list 0 'whaterver)
'()
> ( generate-uniform-list 2 '(racket prolog haskell rust))
'((racket prolog haskell rust) (racket prolog haskell rust))
>
```

Task 2 - Association List Generator



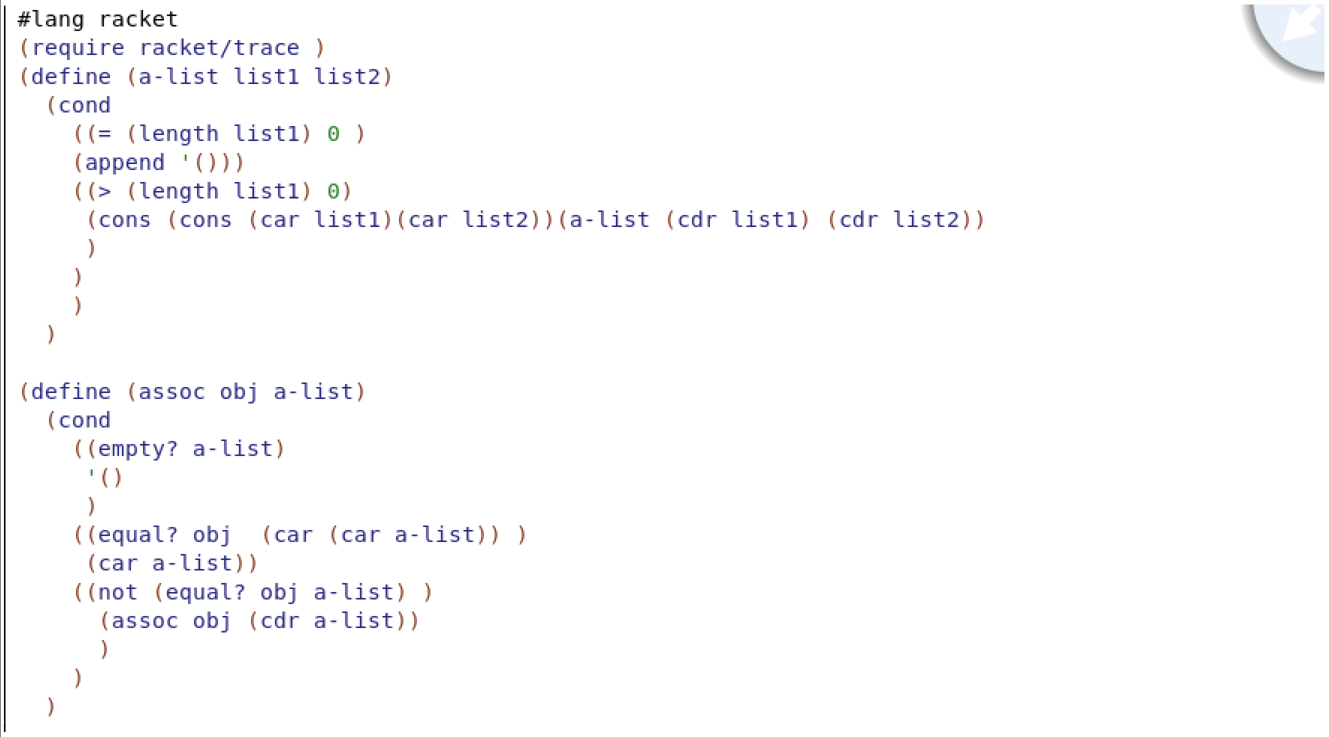
```
#lang racket
(require racket/trace)
(define (a-list list1 list2)
  (cond
    ((= (length list1) 0 )
    (append '()))
    ((> (length list1) 0)
     (cons (cons (car list1)(car list2))(a-list (cdr list1) (cdr list2))
     )
    )
    )
  )
```

Demo



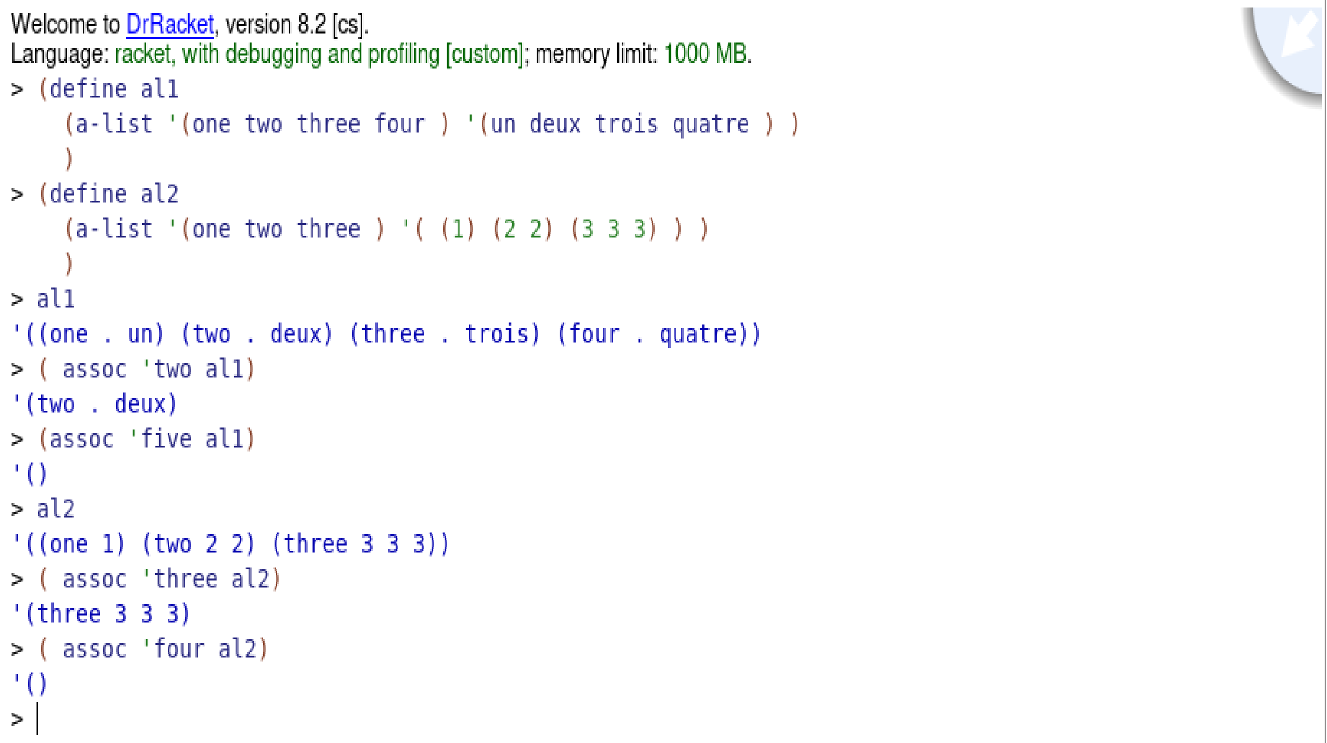
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> ( a-list '(one two three four five) '(un deux trois quarte cinq ))
'((one . un) (two . deux) (three . trois) (four . quarte) (five . cinq))
> ( a-list '() '() )
'()
> ( a-list '(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



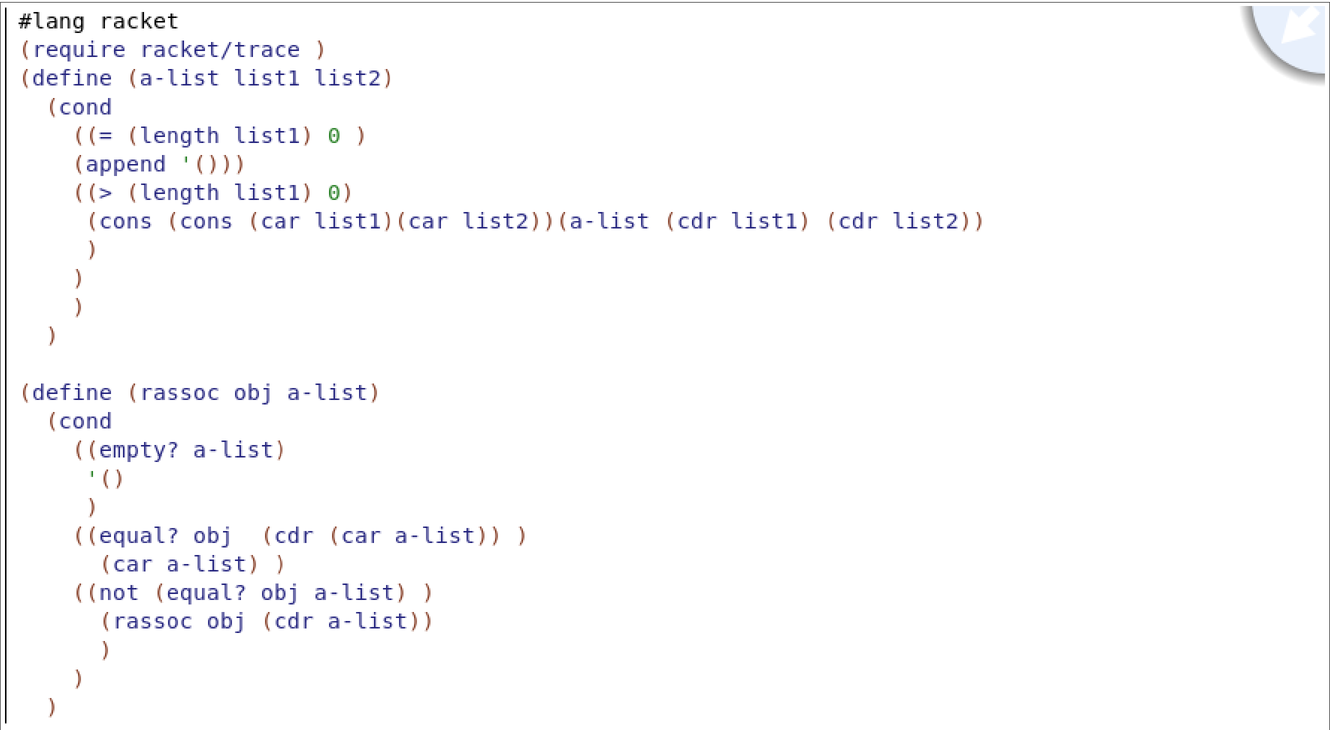
```
#lang racket
(require racket/trace )
(define (a-list list1 list2)
  (cond
    ((= (length list1) 0 )
    (append '()))
    ((> (length list1) 0)
     (cons (cons (car list1)(car list2))(a-list (cdr list1) (cdr list2))
     )
    )
    )
  )

(define (assoc obj a-list)
  (cond
    ((empty? a-list)
     '()
     )
    ((equal? obj  (car (car a-list)) )
     (car a-list))
    ((not (equal? obj a-list) )
      (assoc obj (cdr a-list))
      )
    )
  )
```

Demo



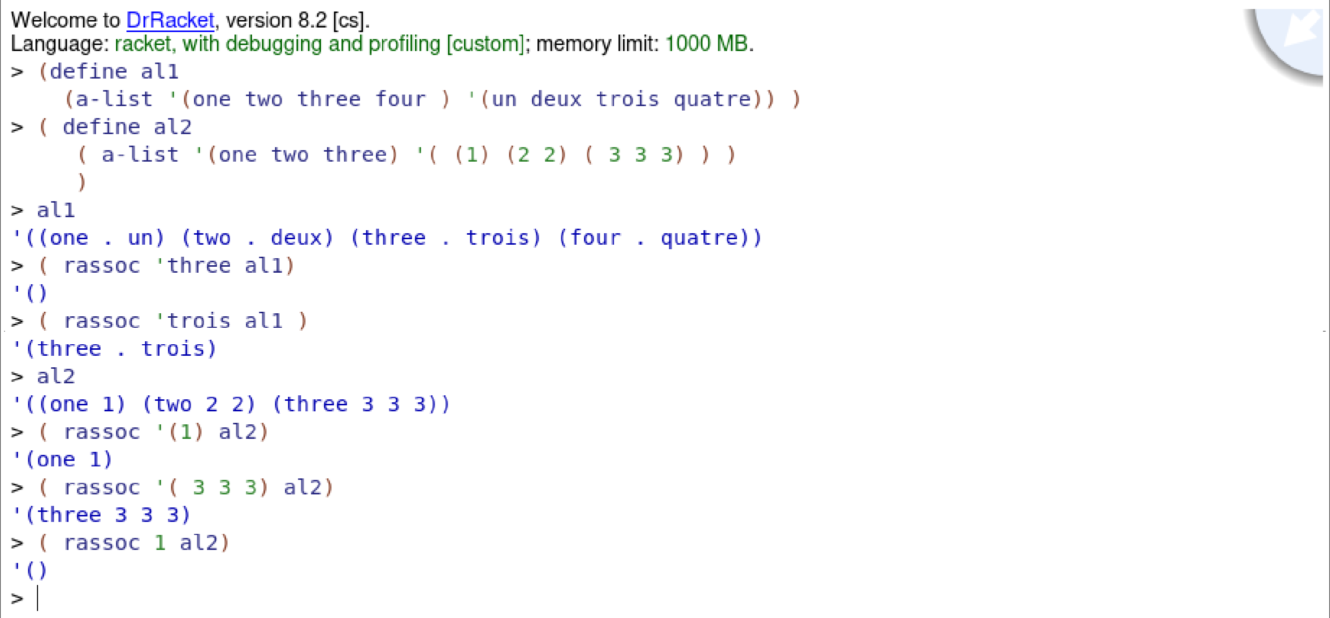
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> (define al1
    (a-list '(one two three four ) '(un deux trois quatre ) )
    )
> (define al2
    (a-list '(one two three ) '( (1) (2 2) (3 3 3) ) )
    )
> al1
'((one . un) (two . 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



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

(define (rassoc obj a-list)
  (cond
    ((empty? a-list)
     '()
     )
    ((equal? obj  (cdr (car a-list)) )
      (car a-list) )
    ((not (equal? obj a-list) )
      (rassoc obj (cdr a-list))
      )
    )
  )
```

Demo

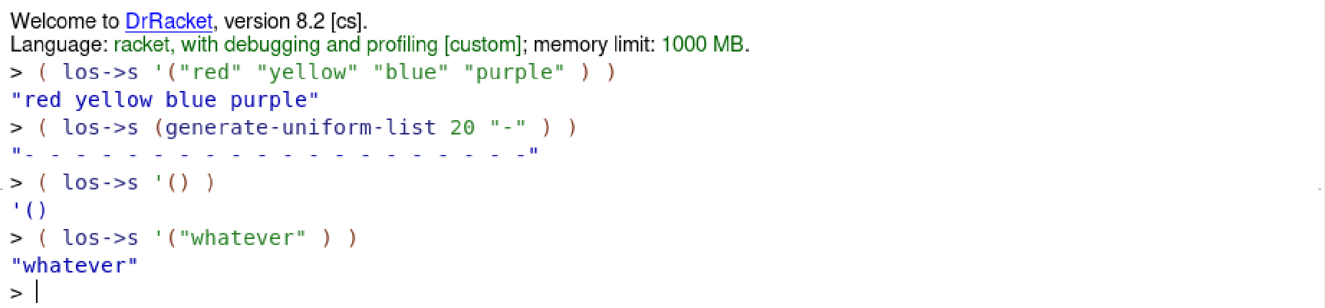


```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> (define al1
    (a-list '(one two three four ) '(un deux trois quatre)) )
> ( define al2
     ( a-list '(one two three) '( (1) (2 2) ( 3 3 3) ) )
     )
> al1
'((one . un) (two . 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 – Los→s

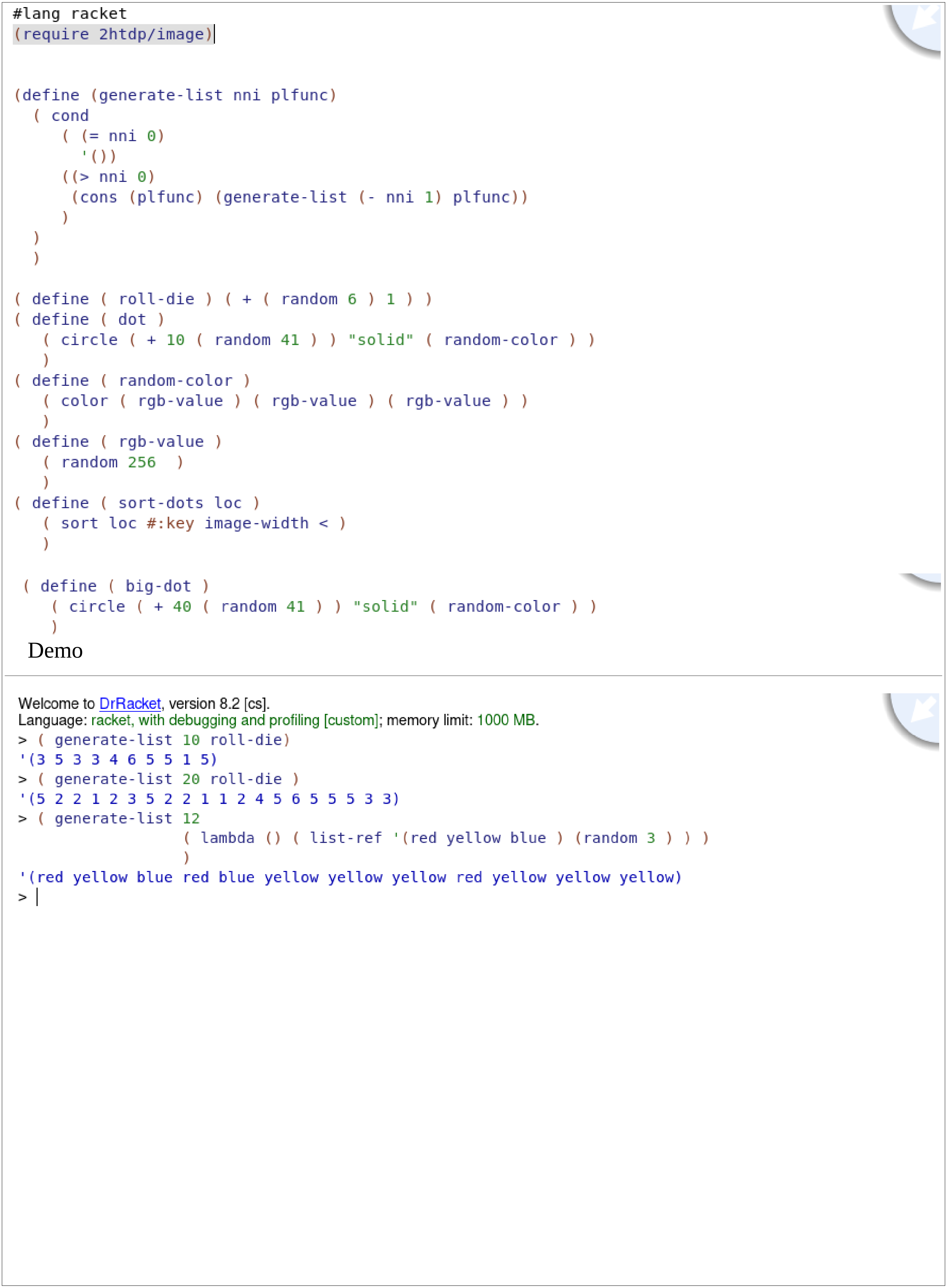
```racket
#lang racket
(require racket/trace)

(define (los->s listStrings)
  (cond
    ((= (length listStrings) 0) '())
    ((=  (length listStrings) 1)
     (string-append (car listStrings)))
    ((> (length listStrings)1)
        (string-append (car listStrings) " "  (los->s (cdr listStrings))))

  )
  )
(define (generate-uniform-list nni obj)
  (cond
    ((= nni 0)
     (append '()))
    ((> nni 0)
     (cons obj (generate-uniform-list (- nni 1) obj))
  )
  )
  )
(trace los->s)

;(los->s '( "red" "yellow" "blue" "purple" ) )
(define listS '( "red" "yellow" "blue" "purple" ))
```

Demo



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> ( los->s '("red" "yellow" "blue" "purple" ) )
"red yellow blue purple"
> ( los->s (generate-uniform-list 20 "-" ) )
"- - - - - - - - - - - - - - - - - - - -"
> ( los->s '() )
'()
> ( los->s '("whatever" ) )
"whatever"
> |
```

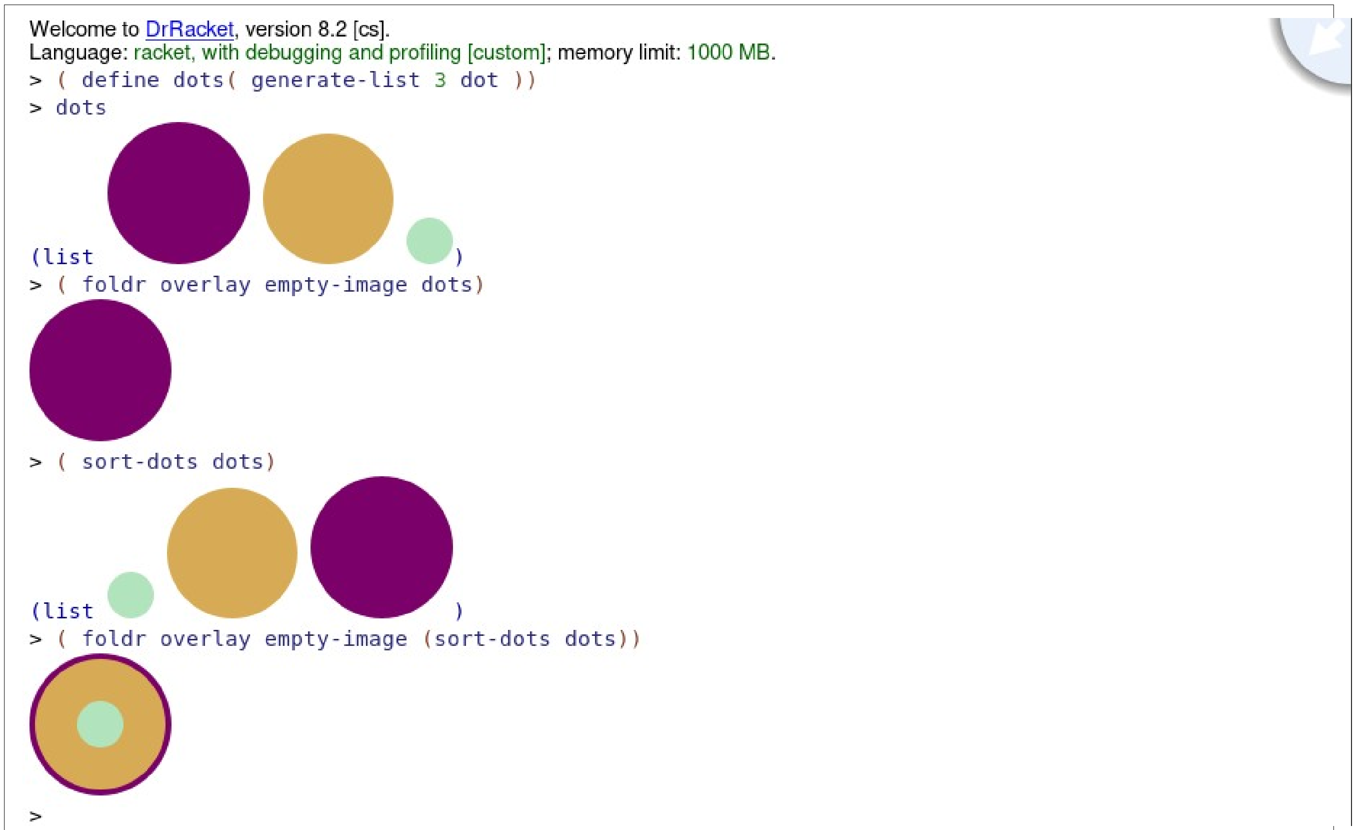
Task 6 - Generate list



```racket
#lang racket
(require 2htdp/image)


(define (generate-list nni plfunc)
  ( cond
     ( (= nni 0)
       '())
     ((> nni 0)
      (cons (plfunc) (generate-list (- nni 1) plfunc))
     )
  )
  )

( 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 < )
   )

 ( define ( big-dot )
    ( circle ( + 40 ( random 41 ) ) "solid" ( random-color ) )
    )
```

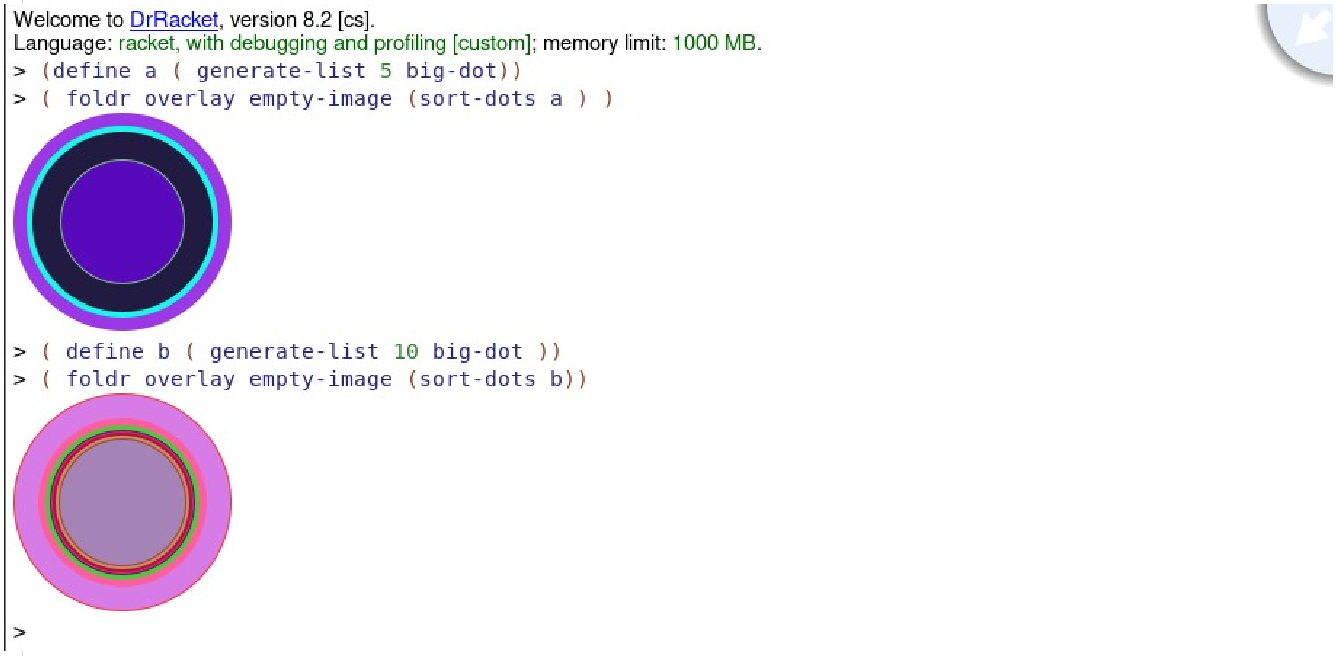
Demo

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> ( generate-list 10 roll-die)
'(3 5 3 3 4 6 5 5 1 5)
> ( generate-list 20 roll-die )
'(5 2 2 1 2 3 5 2 2 1 1 2 4 5 6 5 5 5 3 3)
> ( generate-list 12
                  ( lambda () ( list-ref '(red yellow blue ) (random 3 ) ) )
                  )
'(red yellow blue red blue yellow yellow yellow red yellow yellow yellow)
> 
```



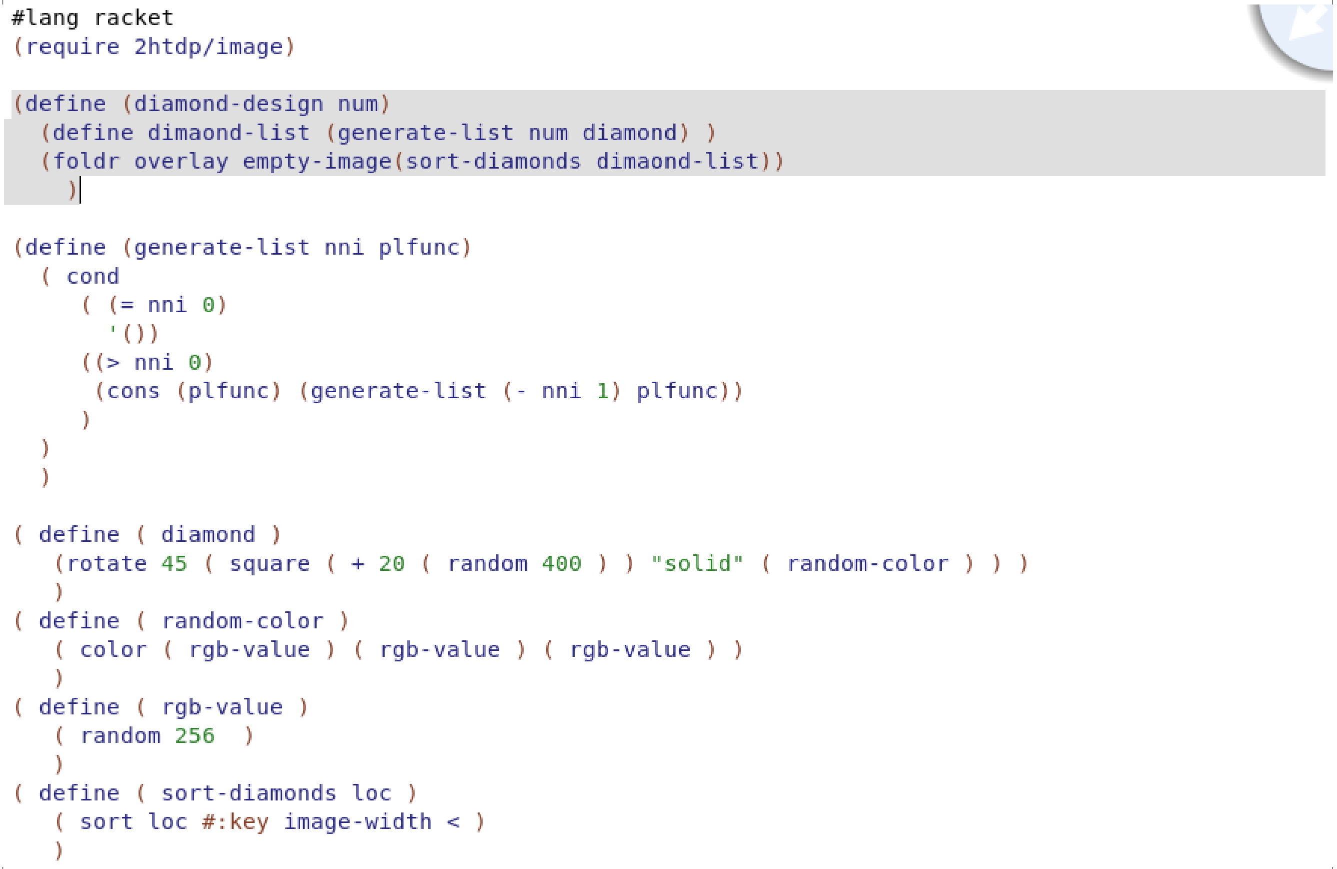
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> ( define dots( generate-list 3 dot ))
> dots
(list                                    )
> ( foldr overlay empty-image dots)

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

>
```

A4T6 Demo 3



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 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))

>
```

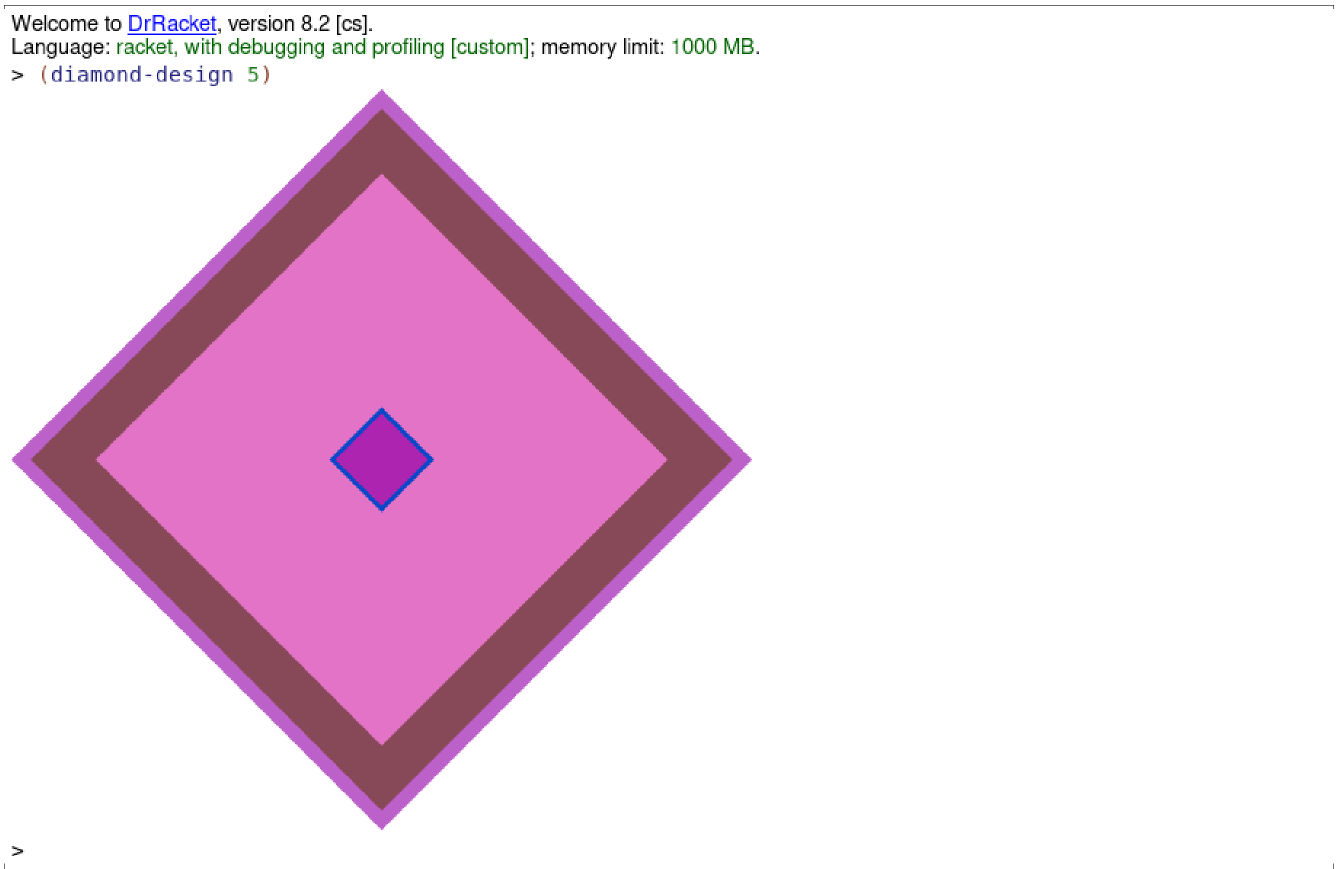
A4T7 Src



```
#lang racket
(require 2htdp/image)

(define (diamond-design num)
  (define dimaond-list (generate-list num diamond) )
  (foldr overlay empty-image(sort-diamonds dimaond-list))
    )

(define (generate-list nni plfunc)
  ( cond
     ( (= nni 0)
       '())
     ((> nni 0)
      (cons (plfunc) (generate-list (- nni 1) plfunc))
     )
  )
  )

( define ( diamond )
   (rotate 45 ( square ( + 20 ( random 400 ) ) "solid" ( random-color ) ) )
   )
( define ( random-color )
   ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) )
   )
( define ( rgb-value )
   ( random 256  )
   )
( define ( sort-diamonds loc )
   ( sort loc #:key image-width < )
   )
```

sas

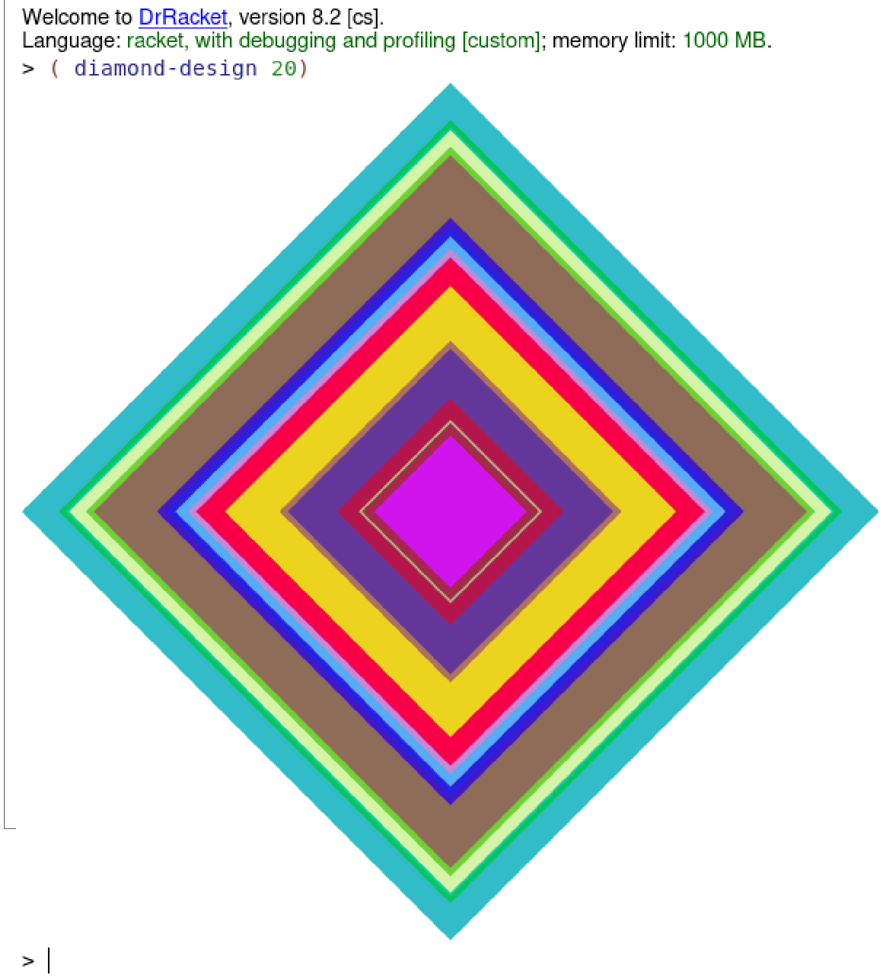
A4T7 Demo



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> (diamond-design 5)
```

```
>
```

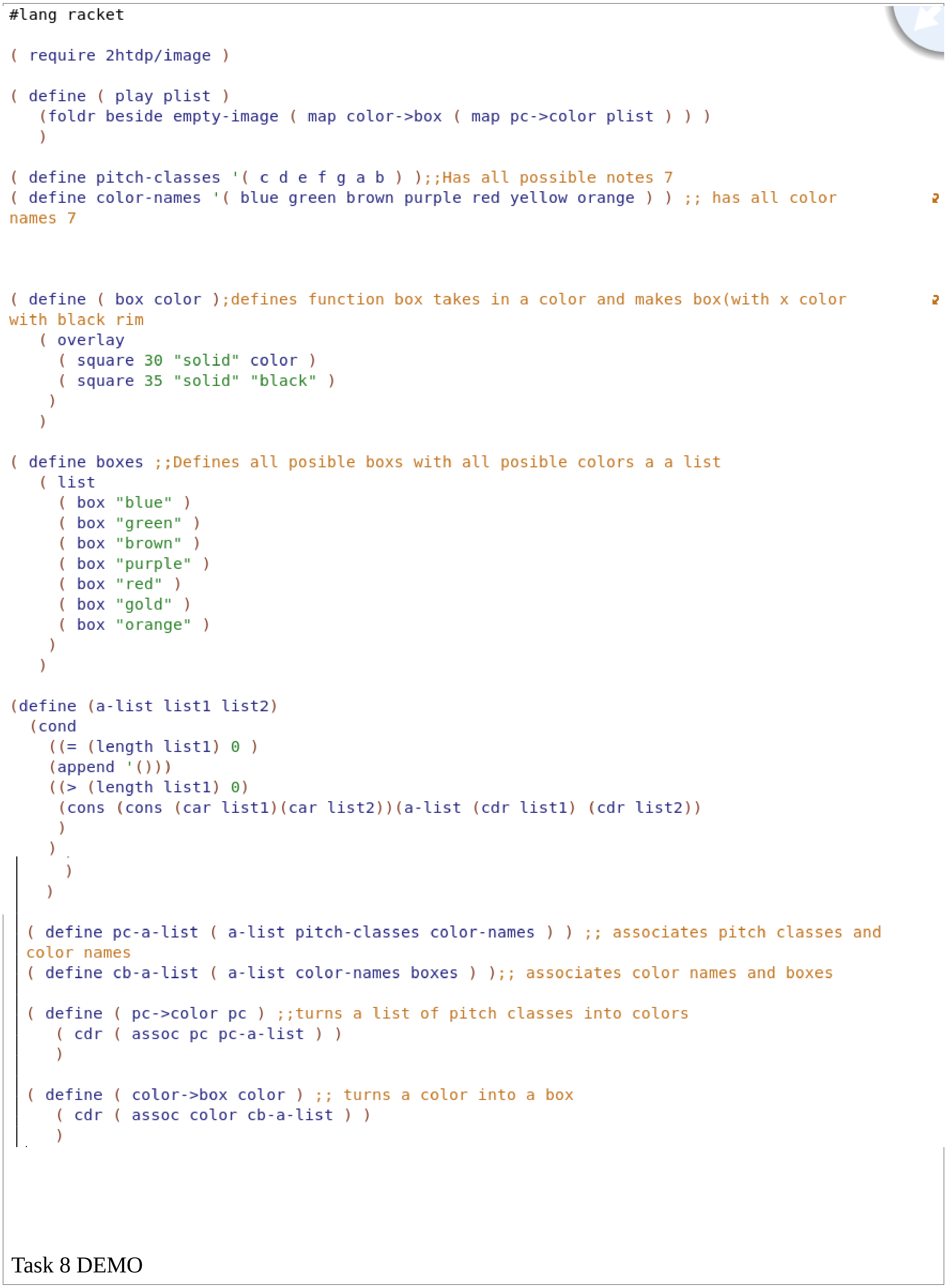
A4T7 Demo 2



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> ( diamond-design 20)
```

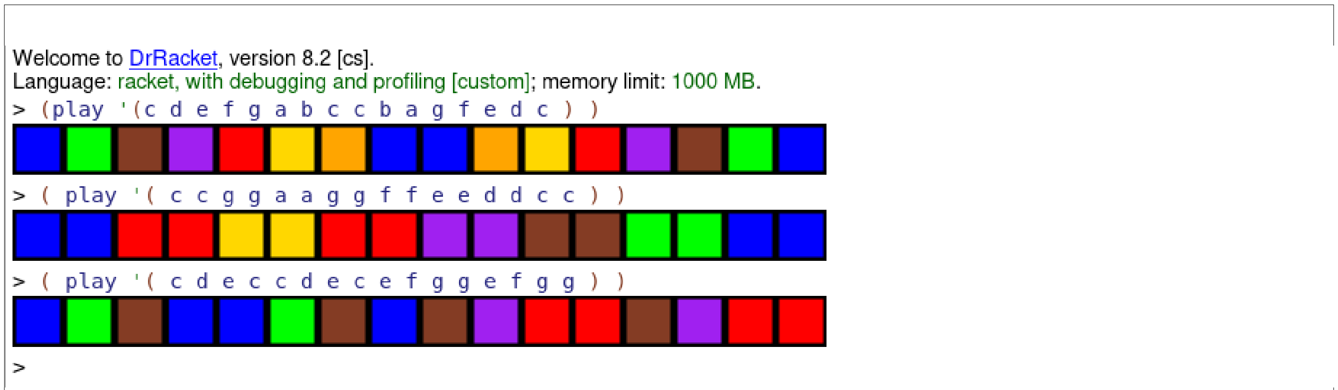
```
>
```

A4T8



```
#lang racket

( require 2htdp/image )

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

( define pitch-classes '( c d e f g a b ) );;Has all possible notes 7
( define color-names '( blue green brown purple red yellow orange ) ) ;; has all color
names 7



( define ( box color );defines function box takes in a color and makes box(with x color
with black rim
   ( overlay
     ( square 30 "solid" color )
     ( square 35 "solid" "black" )
    )
   )

( define boxes ;;Defines all posible boxs with all posible colors a a list
   ( list
     ( box "blue" )
     ( box "green" )
     ( box "brown" )
     ( box "purple" )
     ( box "red" )
     ( box "gold" )
     ( box "orange" )
    )
   )

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

( define pc-a-list ( a-list pitch-classes color-names ) ) ;; associates pitch classes and
color names
( define cb-a-list ( a-list color-names boxes ) );; associates color names and boxes

( define ( pc->color pc ) ;;turns a list of pitch classes into colors
   ( cdr ( assoc pc pc-a-list ) )
   )

( define ( color->box color ) ;; turns a color into a box
   ( cdr ( assoc color cb-a-list ) )
   )
```

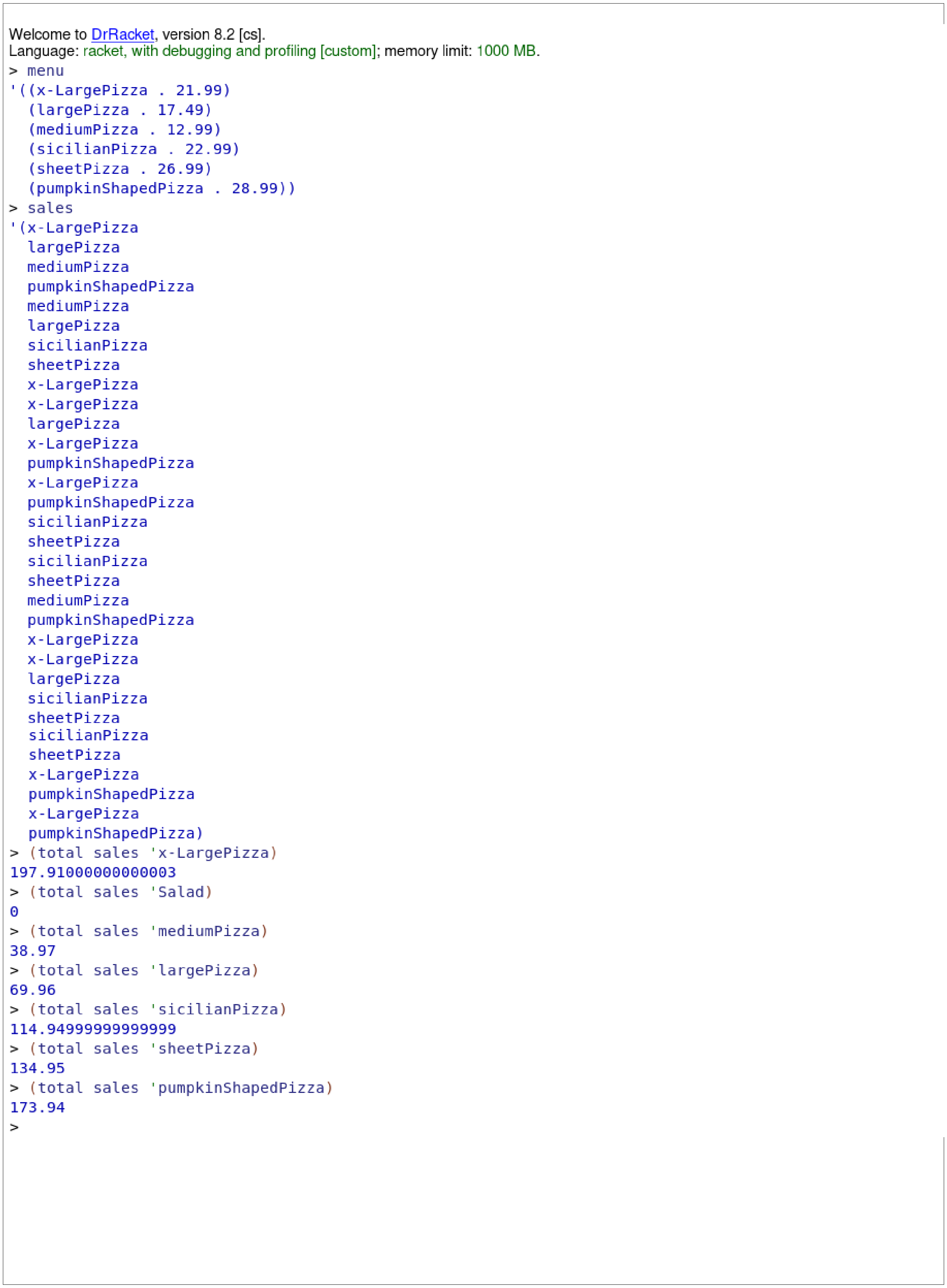
Task 8 DEMO



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 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 ) )
🟦🟩🟫🟦🟦🟩🟫🟦🟫🟪🟥🟥🟫🟪🟥🟥
>
```

Task 9

```
#lang racket
(require racket/trace)


(define menu '((x-LargePizza . 21.99) (largePizza . 17.49) (mediumPizza . 12.99)
(sicilianPizza . 22.99) (sheetPizza . 26.99) (pumpkinShapedPizza . 28.99))
  )


( define price ( lambda(x) ( cdr ( assoc x menu ) ) ) ) ;;returns the price of item X (has
to be in menu)


( define ( total listsold item)
  ( define allOfItem (filter (lambda (x) (equal? x item)) listsold))
   (foldr + 0 (map price allOfItem) )
   )


(trace total)

( define sales '(x-LargePizza largePizza mediumPizza pumpkinShapedPizza mediumPizza
largePizza sicilianPizza sheetPizza x-LargePizza x-LargePizza largePizza x-LargePizza
pumpkinShapedPizza x-LargePizza pumpkinShapedPizza sicilianPizza sheetPizza sicilianPizza
sheetPizza mediumPizza pumpkinShapedPizza x-LargePizza x-LargePizza largePizza
sicilianPizza sheetPizza sicilianPizza sheetPizza x-LargePizza pumpkinShapedPizza
x-LargePizza pumpkinShapedPizza) )
```



Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.

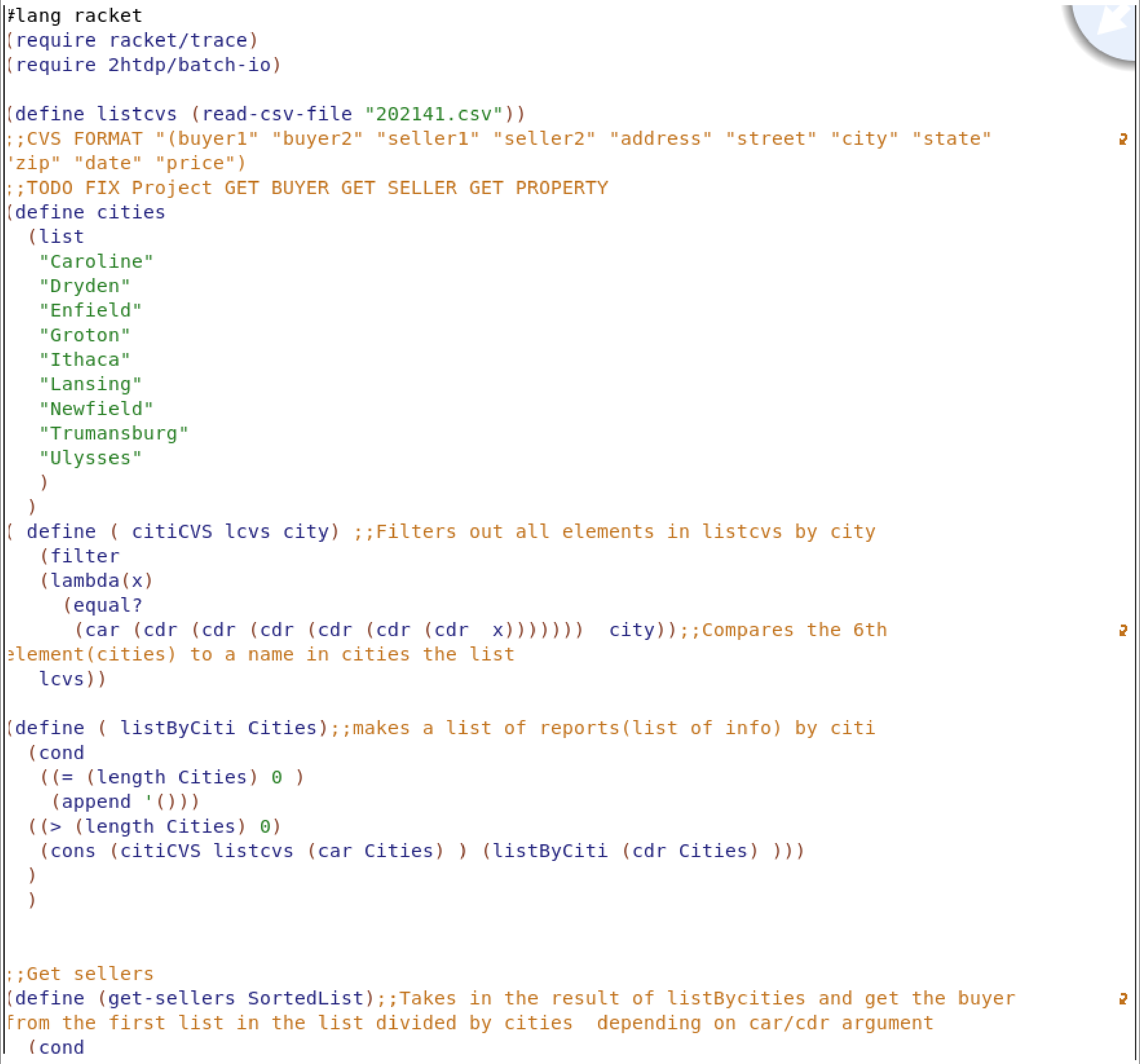
```
> menu
'((x-LargePizza . 21.99)
  (largePizza . 17.49)
  (mediumPizza . 12.99)
  (sicilianPizza . 22.99)
  (sheetPizza . 26.99)
  (pumpkinShapedPizza . 28.99))
> sales
'(x-LargePizza
  largePizza
  mediumPizza
  pumpkinShapedPizza
  mediumPizza
  largePizza
  sicilianPizza
  sheetPizza
  x-LargePizza
  x-LargePizza
  largePizza
  x-LargePizza
  pumpkinShapedPizza
  x-LargePizza
  pumpkinShapedPizza
  sicilianPizza
  sheetPizza
  sicilianPizza
  sheetPizza
  mediumPizza
  pumpkinShapedPizza
  x-LargePizza
  x-LargePizza
  largePizza
  sicilianPizza
  sheetPizza
  sicilianPizza
  sheetPizza
  x-LargePizza
  pumpkinShapedPizza
  x-LargePizza
  pumpkinShapedPizza)
> (total sales 'x-LargePizza)
197.91000000000003
> (total sales 'Salad)
0
> (total sales 'mediumPizza)
38.97
> (total sales 'largePizza)
69.96
> (total sales 'sicilianPizza)
114.94999999999999
> (total sales 'sheetPizza)
134.95
> (total sales 'pumpkinShapedPizza)
173.94
>
```

Task 10

Specification: A csv file separated into (buyer1" "buyer2" "seller1" "seller2" "address" "street" "city" "state" "zip" "date" "price") is used to print out a sales report(by selected city) showing who sold what property to who and how much they sold it for. Then a total for how much money was spent in these transaction is displayed .

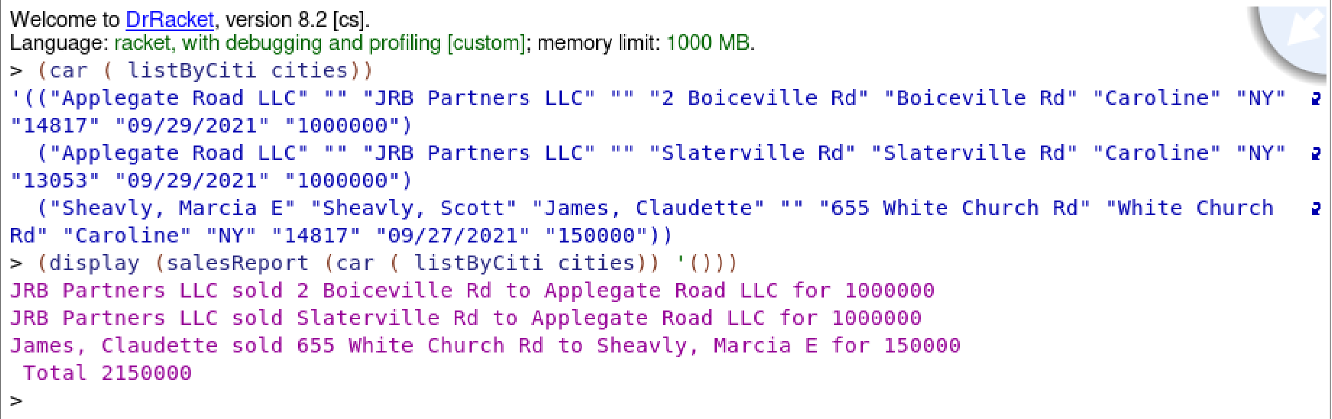
- The first parameter is a list of list separated by cities and the transactions in that city
- The second should be an empty list this will be used to sum up the total money

SRC



```
#lang racket
(require racket/trace)
(require 2htdp/batch-io)

(define listcvs (read-csv-file "202141.csv"))
;;CVS FORMAT "(buyer1" "buyer2" "seller1" "seller2" "address" "street" "city" "state"
"zip" "date" "price")
;;TODO FIX Project GET BUYER GET SELLER GET PROPERTY
(define cities
  (list
   "Caroline"
   "Dryden"
   "Enfield"
   "Groton"
   "Ithaca"
   "Lansing"
   "Newfield"
   "Trumansburg"
   "Ulysses"
   )
  )
( define ( citiCVS lcvs city) ;;Filters out all elements in listcvs by city
   (filter
   (lambda(x)
     (equal?
      (car (cdr (cdr (cdr (cdr (cdr (cdr  x)))))))  city));;Compares the 6th
element(cities) to a name in cities the list
   lcvs))

(define ( listByCiti Cities);;makes a list of reports(list of info) by citi
  (cond
   ((= (length Cities) 0 )
    (append '()))
  ((> (length Cities) 0)
   (cons (citiCVS listcvs (car Cities) ) (listByCiti (cdr Cities) )))
  )
  )


;;Get sellers
(define (get-sellers SortedList);;Takes in the result of listBycities and get the buyer
from the first list in the list divided by cities  depending on car/cdr argument
  (cond
    ((eq?  (car (cdr (cdr (cdr (car SortedList))))) "" ) ;; checks to see if there is a
second buyer if not retrun first buyer
     (cons (car(cdr(cdr(car  SortedList)))) '()))
    ((not(eq? (car (cdr (cdr (car SortedList) ))) "")) ;;if there is second buyer return
list of both buyers
     (cons (car (cdr(cdr (car SortedList)))) (cons  (cdr (cdr (cdr (car SortedList))))
'())))
    )
  )
  ;;CVS FORMAT "(buyer1" "buyer2" "seller1" "seller2" "address" "street" "city" "state"
"zip" "date" "price")
;;get buyers
(define (get-buyer SortedListS);;Takes in the result of listBycities and get the buyer
from the first list in the list divided by cities  depending on car/cdr argument
  (cond
    ((eq? (car (cdr (car SortedListS))) "" ) ;; checks to see if there is a second buyer
if not retrun first buyer
     (cons (car(car  SortedListS)) '()))
    ((not(eq? (car (cdr (car SortedListS) )) "")) ;;if there is second buyer return list
of both buyers
     (cons (car (car SortedListS)) (cons (car (cdr (car SortedListS))) '())))
    )
  )

;;get property/address

(define (get-property SortedListP);;Takes in the result of listBycities and get the
property from the list depending on car/cdr argument
 (cons  (car (cdr (cdr (cdr (cdr (car SortedListP)))))) '())
  )


;;get price
(define (get-price SortedListPri);;Takes in the result of listBycities and get the price
from the list depending on car/cdr argument
 (cons  (car (cdr (cdr(cdr(cdr(cdr(cdr(cdr (cdr (cdr (cdr (car SortedListPri)))))))))))
'())
  )

;;get date
(define (get-date SortedListD);;Takes in the result of listBycities and get the price from
the list depending on car/cdr argument
 (cons  (car (cdr(cdr(cdr(cdr(cdr(cdr (cdr (cdr (cdr (car SortedListD))))))))))) '())
  )



( define (salesReport X Revenue) ;; Takes in the car of the result of listBycities cities
and a empty list and outputs  ;;SELLER "sold" PROPERTY/ADDRESS "to" BUYER "for" PRICE and
TOTAL
       (cond
         ( (=(length X) 0)
           (string-append " Total "   (number->string (foldr + 0 (map string->number
Revenue))) ))
         ( (>( length  X) 0)
           (string-append
            (car (get-sellers  X))  " sold "  (car (get-property X)) " to " (car
(get-buyer X)) " for " (car (get-price  X))
            "\n" (salesReport (cdr X) (append Revenue (get-price  X))) )
           )




           )


   )
```

DEMO



Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.

```
> (car ( listByCiti cities))
'(("Applegate Road LLC" "" "JRB Partners LLC" "" "2 Boiceville Rd" "Boiceville Rd" "Caroline" "NY"
"14817" "09/29/2021" "1000000")
  ("Applegate Road LLC" "" "JRB Partners LLC" "" "Slaterville Rd" "Slaterville Rd" "Caroline" "NY"
"13053" "09/29/2021" "1000000")
  ("Sheavly, Marcia E" "Sheavly, Scott" "James, Claudette" "" "655 White Church Rd" "White Church
Rd" "Caroline" "NY" "14817" "09/27/2021" "150000"))
> (display (salesReport (car ( listByCiti cities)) '()))
JRB Partners LLC sold 2 Boiceville Rd to Applegate Road LLC for 1000000
JRB Partners LLC sold Slaterville Rd to Applegate Road LLC for 1000000
James, Claudette sold 655 White Church Rd to Sheavly, Marcia E for 150000
 Total 2150000
>
```