Abstract:

This assignment will help me improve my Racket programming skills by performing various interactions, defining functions, and solving problems with the use of existing code. The goal is to enhance my problem-solving abilities and practice efficient code reuse.

Task 1: Interaction - Scrap of Tin

Arithmetic Expressions -

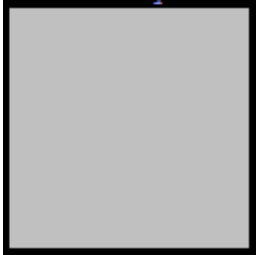
```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> 5
5
> 5.3
5.3
> ( * 3 10)
30
> ( + ( * 3 10) 4)
34
> (* 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)
121576654590569288 01
```

Solving a Simple Problem(Area of Scrap)

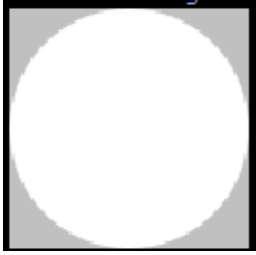
```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> pi
3.141592653589793
> side
side: undefined;
 cannot reference an identifier before its definition
> (define side 100)
> side
100
> (define square-area (* side side) )
> square-area
10000
> (define radius ( / side 2))
> radius
50
> (define circle-area ( * pi radius radius))
> circle-area
7853.981633974483
> (define scrap-area ( - square-area circle-area) )
> scrap-area
2146.018366025517
```

Rendering an Image of the Problem Situation

```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( require 2htdp/image )
> (define side 100)
> ( define the-square ( square side "solid" "silver" ) )
> the-square
```



```
> ( define radius ( / side 2) )
> ( define the-circle ( circle radius "solid" "white") )
> ( define the-image ( overlay the-circle the-square) )
> the-image
```

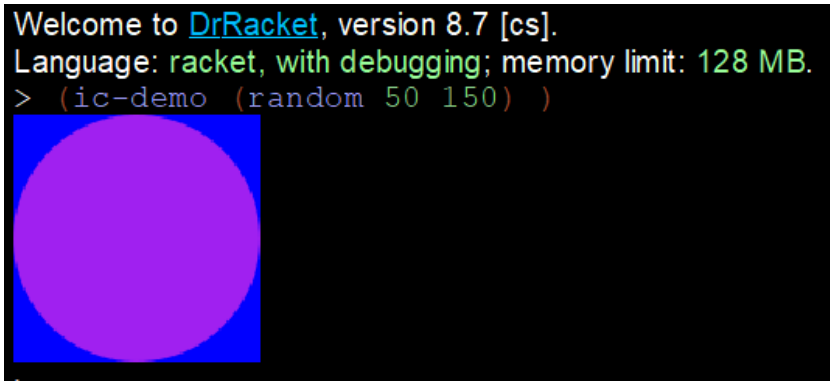


Task 2 - Definitions - Inscribing/Circumscribing Circle/Squares

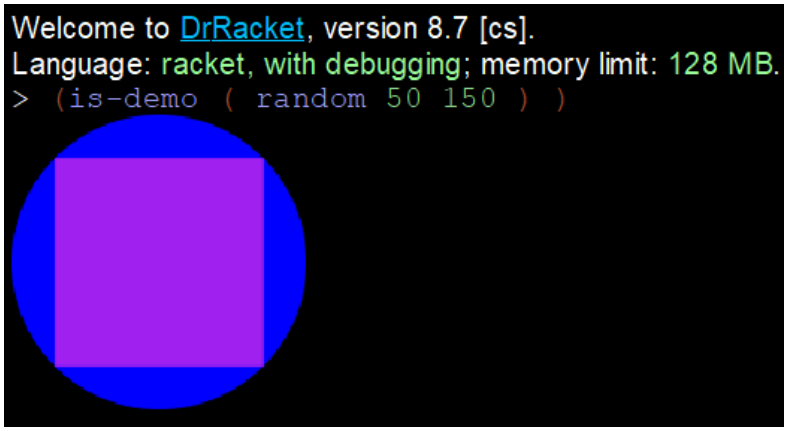
Cs-Demo

Cc-Demo

Ic-Demo



Is-Demo



Task 2 Code:

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

( define ( cs radius )
   (* radius 2 )
   )

(define (cc side-length)
  ( / ( * side-length (sqrt 2) ) 2 )
  )

  ( define (ic side-length )
     ( / side-length 2)
     )
(define (is radius)
  ( * radius ( sqrt 2 ))
  )


(define (cs-demo radius)
( define the-square ( square (cs radius) "solid" "purple" ) )
( define the-circle ( circle radius "solid" "blue") )
( overlay the-circle the-square)
  )

(define (cc-demo side-length)
  (define the-circle ( circle (cc side-length) "solid" "purple") )
  (define the-square (square side-length "solid" "blue" ) )
  (overlay the-square the-circle)
  )

(define (ic-demo side-length )
  (define the-square (square side-length  "solid" "blue") )
  (define the-circle ( circle (ic side-length)  "solid" "purple"))
  (overlay the-circle the-square)
  )

(define (is-demo radius)
  (define the-square ( square (is radius) "solid" "purple" ) )
  (define the-circle ( circle radius "solid" "blue" ) )
  (overlay the-square the-circle)
  )

```

Task 3: Inscribing/ Circumscribing Images

Image-1 Demo

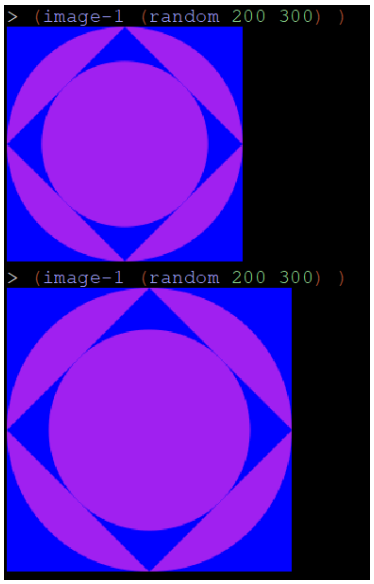
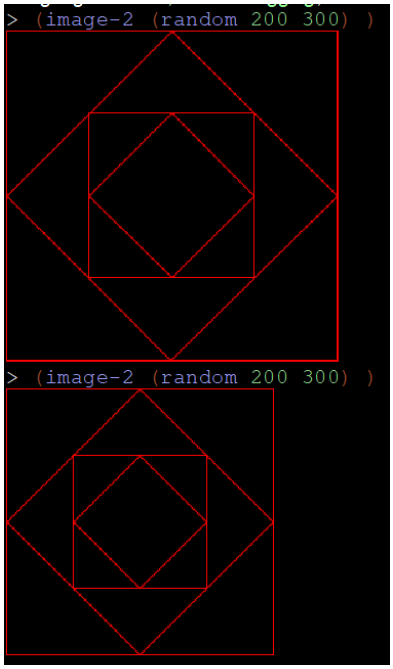
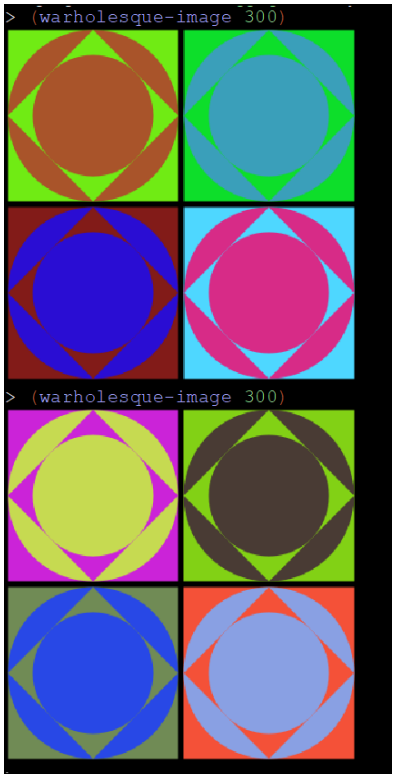


Image-2 Demo:



Warholesque Image:

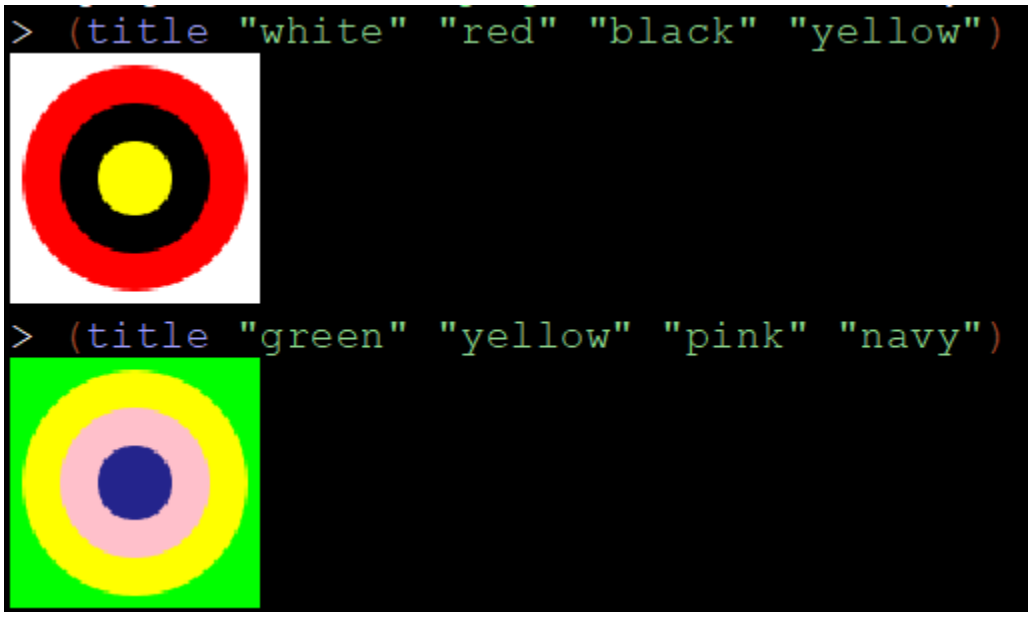


Task 3 Code:

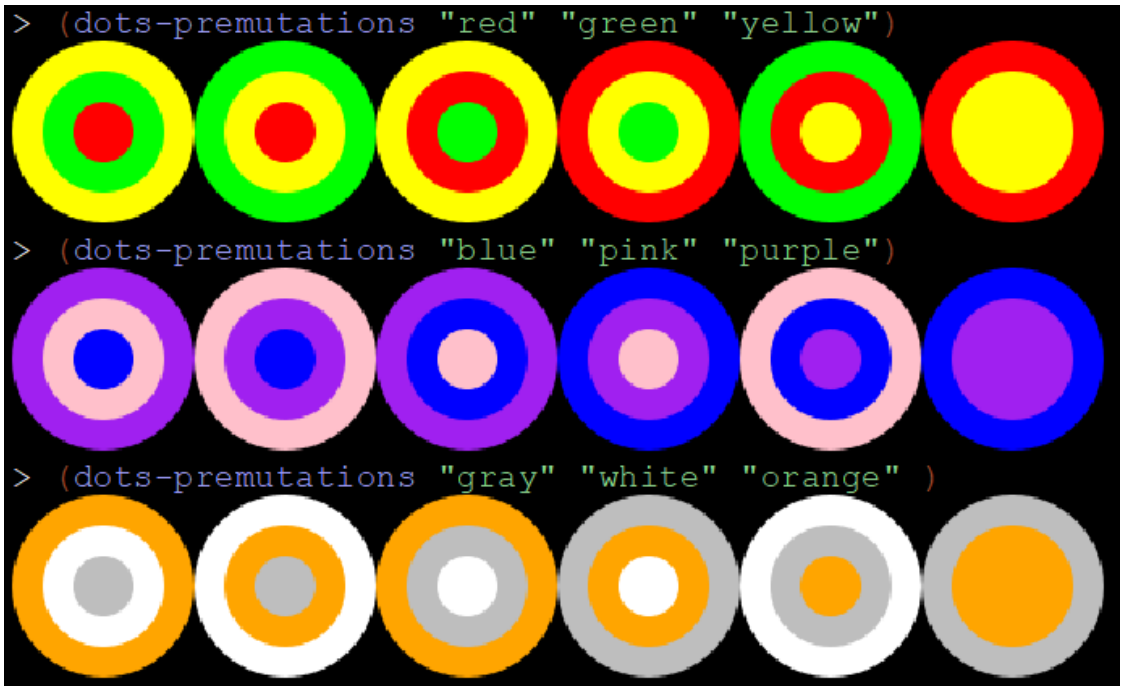
```
(define (image-1 side-length)
  (overlay ( rotate 45 (ic-demo ( is (ic side-length ) ) ) ) (ic-demo side-length) )
  )

(define (image-2 side-length)
  (define square1 ( square side-length "outline" "red" ) )
  ( define next-square1 (is (ic side-length) ) )
  ( define square2 (rotate 45 (square next-square1 "outline" "red" ) ) )
  (define next-square2 (is ( ic next-square1 ) ) )
  ( define square3 (square next-square2 "outline" "red" ) )
  (define next-square3 ( is ( ic next-square2) ) )
  (define square4 (rotate 45 ( square next-square3 "outline" "red" ) ) )
  (overlay square1 square2 square3 square4)
  )

(define (warholesque-image canvas)
  (define ( image-1 canvas)
  (define imageB ( / canvas 2 ) )
    ( define (random-color) ( color (random 0 256) (random 0 256) (random 0 256) ) )
    (define circleColor (random-color) )
    (define squareColor (random-color) )
    (define border (square ( + 5 imageB) "outline" "black") )
    (define circleA (circle (ic imageB ) "solid" circleColor) )
    (define squareA (square imageB "solid" squareColor) )
    (define squareB (rotate 45 (square ( is (ic imageB) ) "solid" squareColor) ) )
    (define circleB (circle (ic (is (ic imageB) ) ) "solid" circleColor) )
    (overlay circleB squareB circleA squareA border)
    )
  (define bigBorder (square ( + 2 canvas) "solid" "black") )
  (overlay ( above
    (beside (image-1 canvas ) (image-1 canvas) )
    (beside (image-1 canvas) (image-1 canvas) ) ) bigBorder )
  )
```

Task 4: Permutations of Randomly Colored Stacked Dots

Demo 1 :



Demo 2:



Code for Task 4:

```
#lang racket

(require 2htdp/image)

(define (title color1 color2 color3 color4)
  (overlay (circle 15 "solid" color4)
          (overlay ( circle 30 "solid" color3)
                   (overlay (circle 45 "solid" color2)
                            (square 100 "solid" color1) ) ) )
  )

(define (dots color1 color2 color3)
  (overlay (circle 15 "solid" color1)
           (overlay(circle 30 "solid" color2)
                   (circle 45 "solid" color3) ) )
  )

(define (dots-premutations color1 color2 color3)
  (beside(dots color1 color2 color3)
         (beside (dots color1 color3 color2)
                 (beside (dots color2 color1 color3)
                         (beside (dots color2 color3 color1)
                                 (beside (dots color3 color1 color2)
                                         (dots color3 color3 color1) ) ) ) ) )
  )

```