Racket Assignment #2: Interactions, Definitions, Applications

Learning Abstract:

In this assignment, I performed a certain set of tasks in Racket with the help of instructions provided by my professor. I got familiar with the characteristics of a function in Racket. This helped me understand the syntax used in this language better and also helped me form a strong base in Racket.

Task 1: Interactions - Scrap of Tin

Arithmetic Expressions

```
> 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 9)
12157665459056928801
>
```

Solve a Simple Problem (Area of scrap)

```
> pi
3.141592653589793
> side
100
> (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

```
> (require 2htdp/image)
> (define side 100)
> (define the-square (square side "solid" "silver"))
> the-square
```

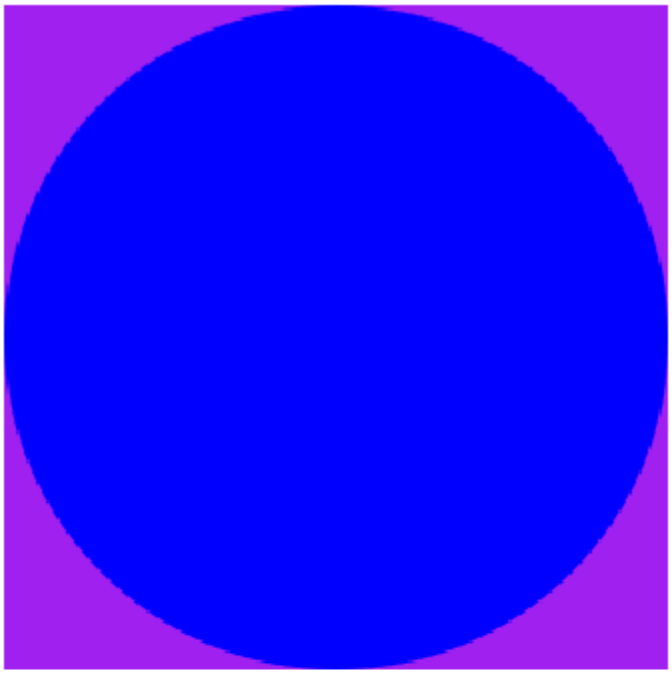
```
> (define radius( / side 2))
> (define the-circle( circle radius "solid" "white"))
> (define the-image( overlay the-circle the-square))
> the-image
```

```
>
```

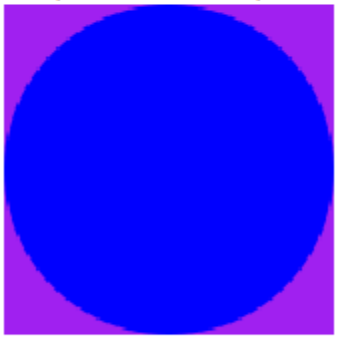
Task 2: Definitions - Inscribing/Circumscribing Circles/Squares

cs-demo

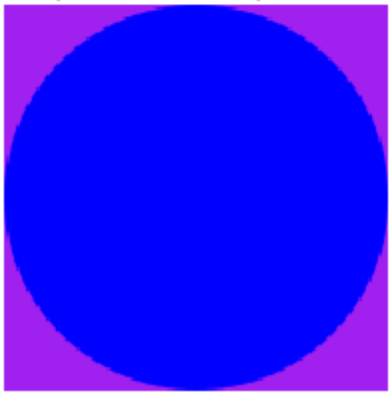
```
> ( cs-demo ( random 50 150 ))
```



```
> ( cs-demo ( random 50 150 ))
```

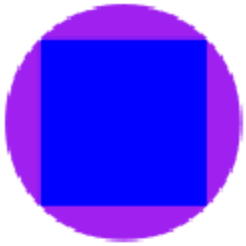


```
> ( cs-demo ( random 50 150 ))
```

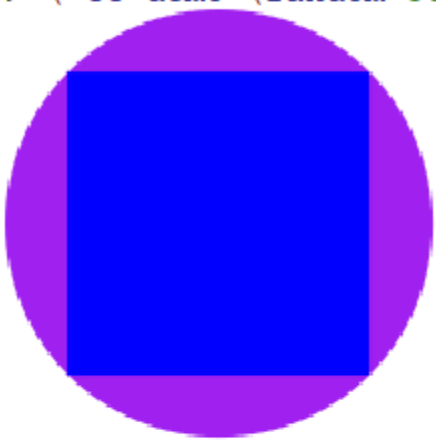


cc-demo

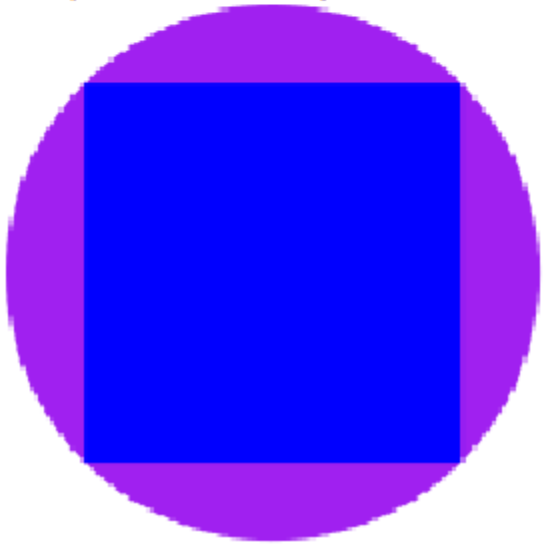
```
> ( cc-demo (random 50 150 ))
```



```
> ( cc-demo (random 50 150 ))
```



```
> ( cc-demo (random 50 150 ))
```

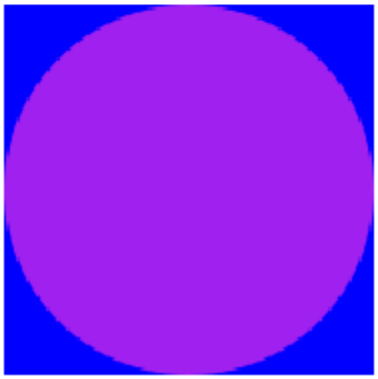


ic-demo

```
> ( ic-demo ( random 50 150 ))
```

```
> ( ic-demo ( random 50 150 ))
```

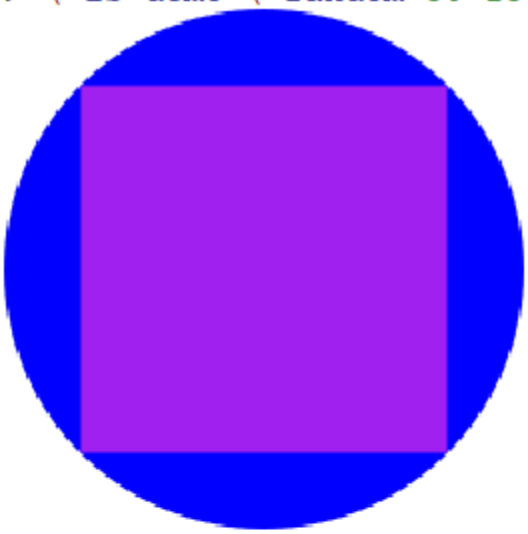
```
> ( ic-demo ( random 50 150 ))
```



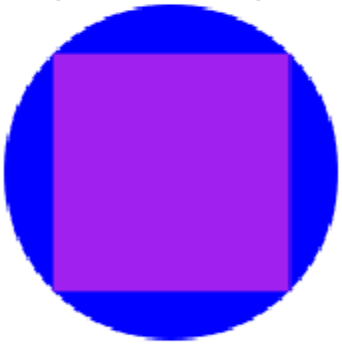
```
>
```

is-demo

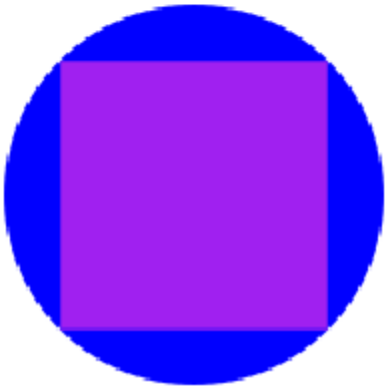
```
> ( is-demo ( random 50 150 ))
```



```
> ( is-demo ( random 50 150 ))
```



```
> ( is-demo ( random 50 150 ))
```



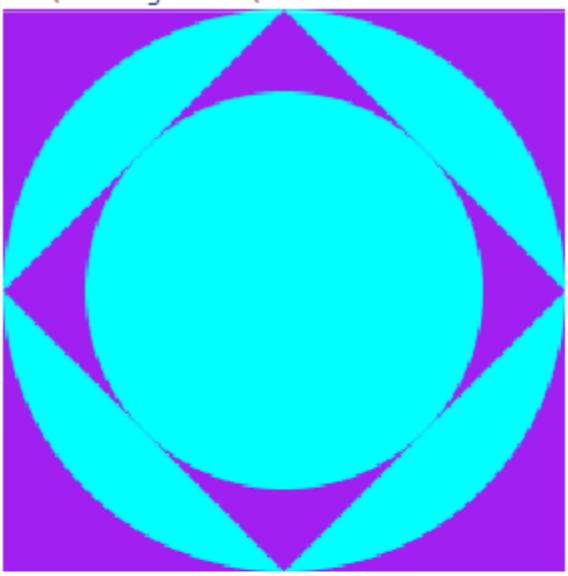
The Code

```
#lang racket
(require 2htdp/image)
( define (cs radius)
   (* radius 2)
   )

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

( define ( ic square-length )
   ( / square-length 2 )
   )

( define ( is circle-radius )
   ( / ( * circle-radius 2 ) (sqrt 2))
   )

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

( define ( cc-demo cc-side )
   ( define sq ( square cc-side "solid" "blue" ))
   ( define cir ( circle ( cc cc-side ) "solid" "purple" ))
   ( overlay sq cir )
   )

( define ( ic-demo ic-side )
   ( define sq ( square ic-side "solid" "blue" ))
   (define cir ( circle (ic ic-side) "solid" "purple" ))
   ( overlay cir sq )
   )

( define (is-demo is-radius)
   ( define cir ( circle is-radius "solid" "blue" ))
    ( define sq ( square ( is is-radius ) "solid" "purple" ))
   (overlay sq cir )
   )
```

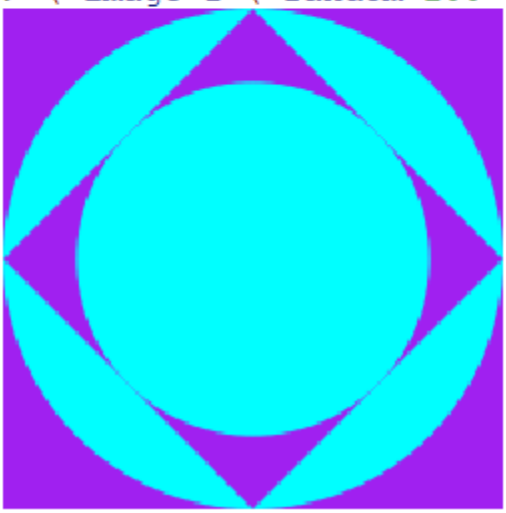
Task 3: Inscribing/Circumscribing Images

Image 1 Demo

```
> ( image-1 ( random 200 300 ) )
```



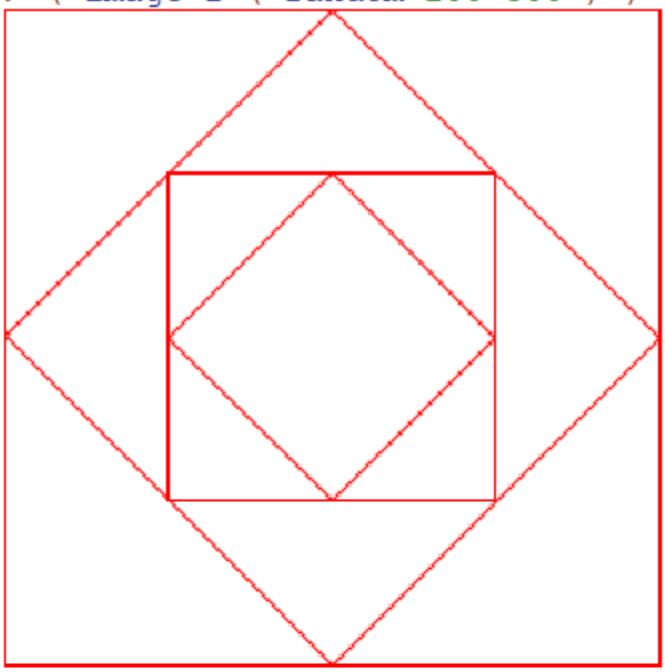
```
> ( image-1 ( random 200 300 ) )
```



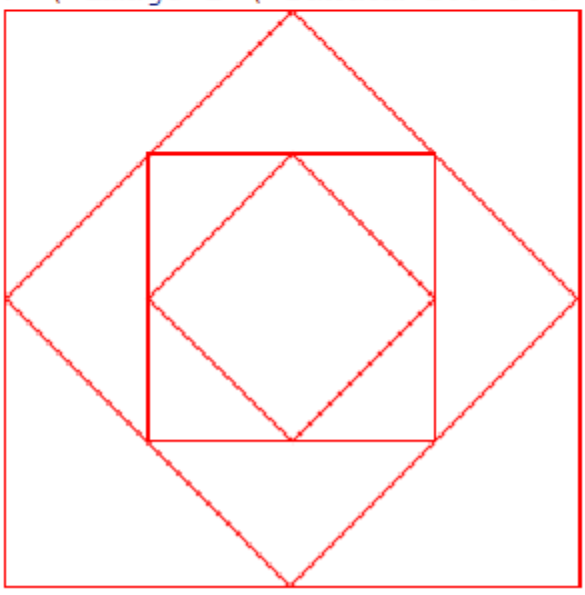
```
>
```

Image 2 Demo

```
> ( image-2 ( random 200 300 ) )
```



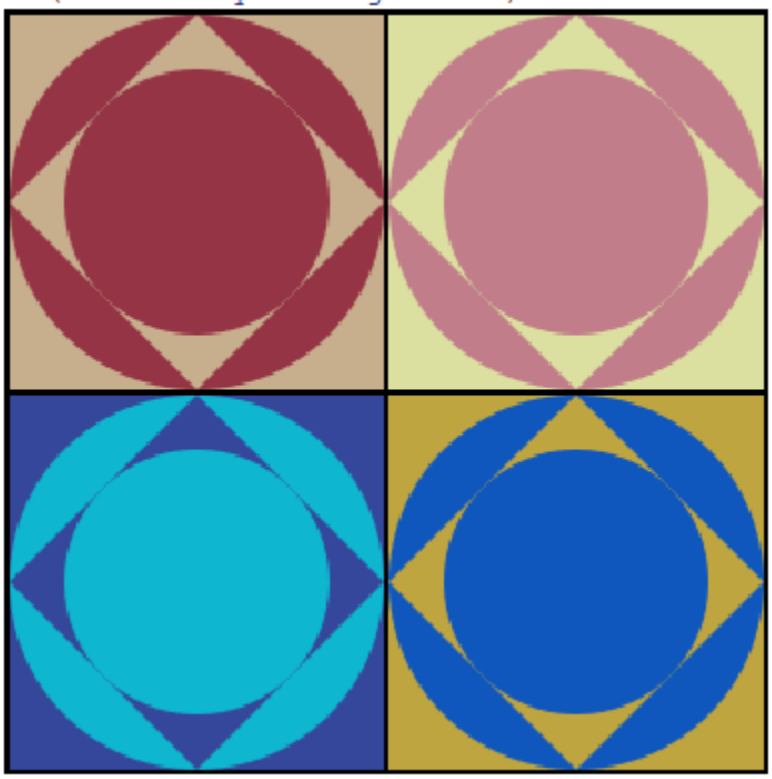
```
> ( image-2 ( random 200 300 ) )
```



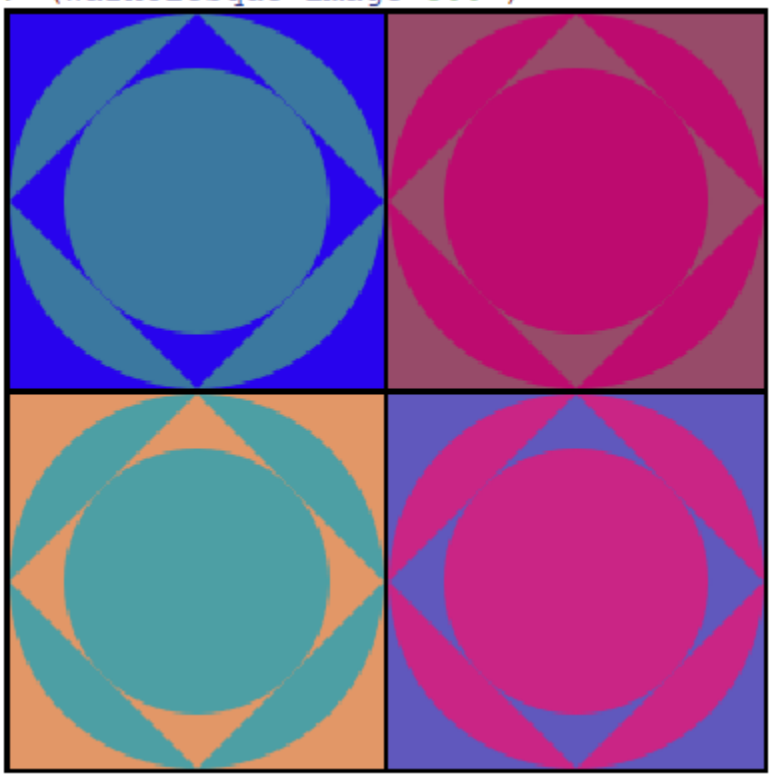
```
>
```

Warholesque Image

```
> (Warholesque-image 300 )
```



```
> (Warholesque-image 300 )
```



```
>
```

Code

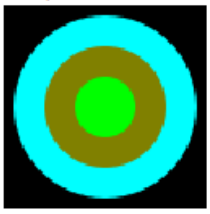
```
( define ( image-1 square-side )
   ( define sq1 ( square square-side "solid" "purple" ))
   ( define cir1 ( circle ( ic square-side ) "solid" "cyan" ))
   ( define sq2 ( rotate 45 ( square ( is (ic square-side) ) "solid"
"purple" )))
   ( define cir2 ( circle (ic ( is (ic square-side) )) "solid" "cyan"
))
   ( overlay cir2 sq2 cir1 sq1 )
   )


( define ( image-2 image-2-side)
   ( define sq1 ( square image-2-side "outline" "red" ))
   ( define sq2 ( rotate 45 ( square ( is ( ic image-2-side ) )
"outline" "red" )))
   ( define sq3 ( square ( is (ic ( is ( ic image-2-side ) ) ) )
"outline" "red" ))
   ( define sq4 ( rotate 45 ( square ( is ( ic ( is ( ic ( is ( ic
image-2-side ) ) ) ) ) ) "outline" "red" )))
   ( overlay sq1 sq2 sq3 sq4 )
   )

( define ( Warholesque-image canvas-side )
 ( define ( image1 canvas-side )
 ( define sq-side ( / canvas-side 2 ) )
 ( define (random-color) ( color ( random 0 256 ) ( random 0 256 ) (
random 0 256 ) ) )
 ( define circle-color ( random-color ) )
 ( define square-color ( random-color ) )
 ( define border ( square ( + 2 sq-side ) "solid" "black" ) )
 ( define cir1 ( circle ( ic sq-side ) "solid" circle-color ) )
 ( define sq1 ( square sq-side "solid" square-color ) )
 ( define sq2 ( rotate 45 ( square ( is ( ic sq-side ) ) "solid"
square-color ) ) )
 ( define cir2 ( circle ( ic ( is ( ic sq-side ) ) ) "solid"
circle-color ) )
 ( overlay cir2 sq2 cir1 sq1 border )
 )
 ( define border1 ( square ( + 6 canvas-side ) "solid" "black" ) )
 ( overlay
```

```
 ( above
 ( beside ( image1 canvas-side ) ( image1 canvas-side ) )
 ( beside ( image1 canvas-side ) ( image1 canvas-side ) )
 )
 border1
 )
)
```

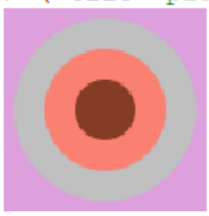
Task 4: Permutations of Randomly Colored Stacked Dots

Demo

```
> ( tile "black" "cyan" "olive" "green" )
```



```
> ( tile "plum" "silver" "salmon" "brown" )
```



```
> |
```

```
> ( dots-permutations "blue" "salmon" "olive" )
```

```
> ( dots-permutations "firebrick" "cyan" "black" )
```

```
> ( dots-permutations "yellow" "olive" "brown" )
```

```
>
```

Code

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

( define ( tile sq-color color-1 color-2 color-3 )
   ( define sq ( square 100 "solid" sq-color ) )
   ( define cir-1 ( circle (/ 90 2 ) "solid" color-1 ) )
   ( define cir-2 ( circle (/ 60 2 ) "solid" color-2 ) )
   ( define cir-3 ( circle (/ 30 2 ) "solid" color-3 ) )
   ( overlay cir-3 cir-2 cir-1 sq )
   )
```

```
( define ( target color1 color2 color3 )
   ( define cir1 ( circle (/ 90 2 ) "solid" color1 ))
   ( define cir2 ( circle (/ 60 2 ) "solid" color2 ))
   ( define cir3 ( circle (/ 30 2 ) "solid" color3 ))
   ( overlay cir3 cir2 cir1 )
   )

(define gap ( square 10 "solid" "white" ))


( define ( dots-permutations clr-1 clr-2 clr-3 )
   (beside gap ( target clr-1 clr-2 clr-3 )
           gap ( target clr-2 clr-3 clr-1 )
           gap ( target clr-3 clr-1 clr-2 )
           gap ( target clr-1 clr-3 clr-2 )
           gap ( target clr-2 clr-1 clr-3 )
           gap ( target clr-3 clr-2 clr-1 ))
   )
```