Damian Randall

Racket Assignment 2: Interactions, Definitions, Applications

Abstract: In this assignment we were asked to complete tasks with specifications and limitations in order to further our understanding of racket in conjunction with problem solving skills we have already learned prior to the class.

Task 1: Interactins - Scrap of Tin:

1. 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
```

---

2. Solve a Simple Problem (Area of Scrap)

```
> 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
```

---

3. 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
```

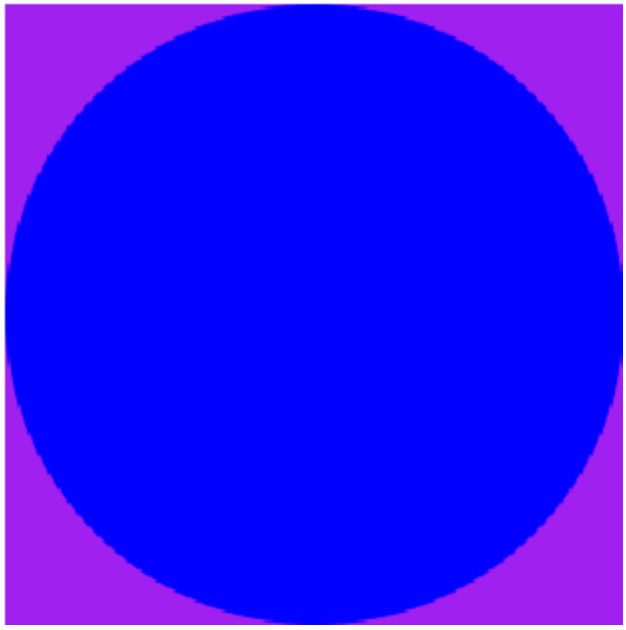
Damian Randall

Task 2: Definitions - Inscribing/Circumscribing Circles/Squares

---

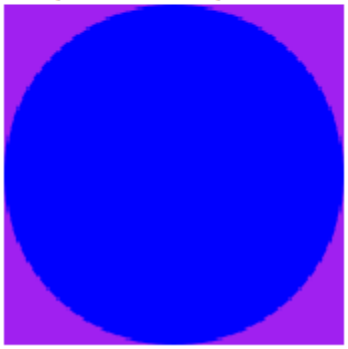
cs-demo:

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



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

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

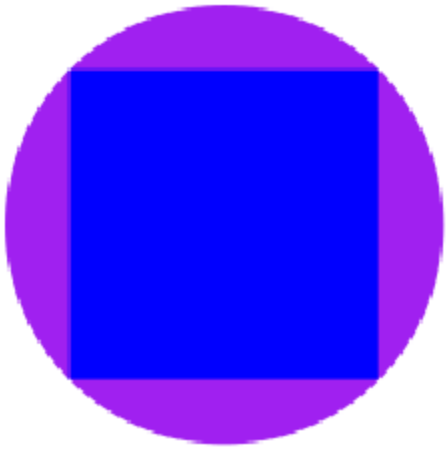


```
>
```

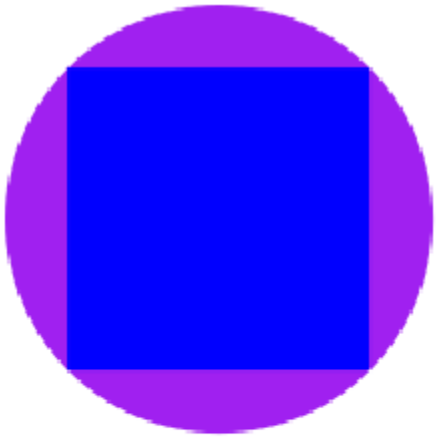
Damian Randall

cc-demo:

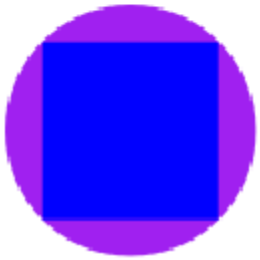
```
> (cc-demo(random 50 150))
```



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



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



Damian Randall

ic-demo:

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

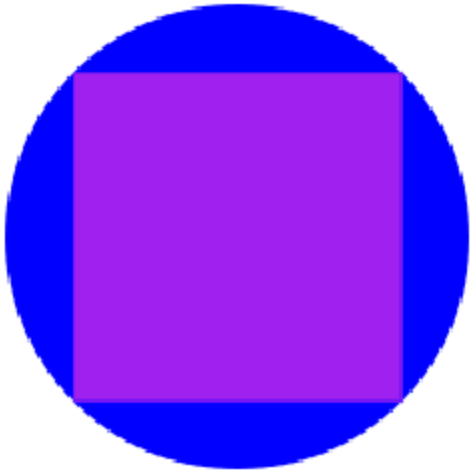
```
> (ic-demo(random 50 150))
```

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

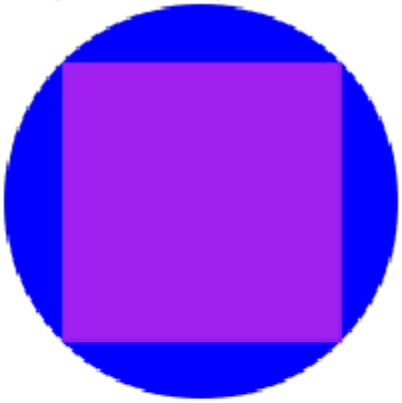
Damian Randall

is-demo:

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



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



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

Damian Randall

Code:

```
(require 2htdp/image)

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

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

(define(ic side)
 (/ side 2)
)

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

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

(define(cc-demo side)
 (define s(square side "solid" "blue"))
 (define c(circle(cc side) "solid" "purple"))
 (overlay s c)
)

(define(ic-demo side)
 (define s(square side "solid" "blue"))
 (define c(circle(ic side) "solid" "purple"))
 (overlay c s)
```

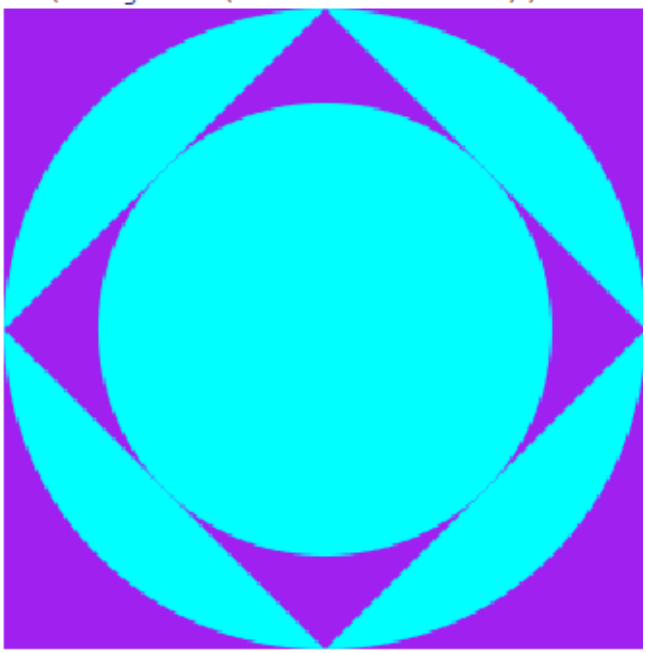
)

```
(define(is-demo radius)
 (define c( circle radius "solid" "blue"))
 (define s(square(is radius) "solid" "purple"))
 (overlay s c)
)
```

Task 3: Inscribing/Circumscribing Images

Image-1 demo:

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



Damian Randall

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

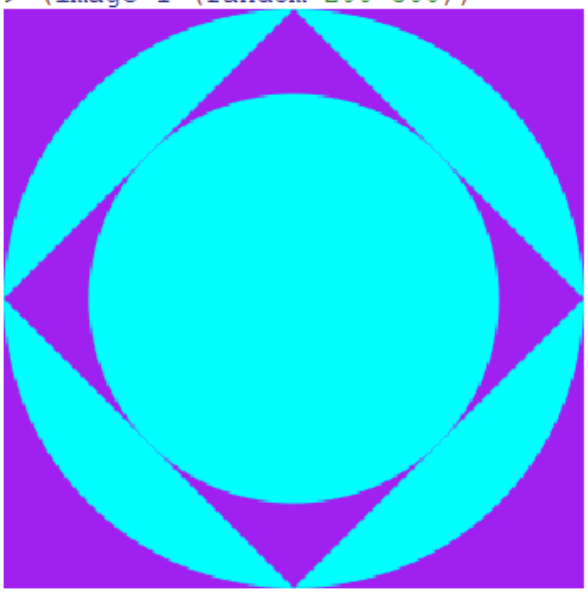
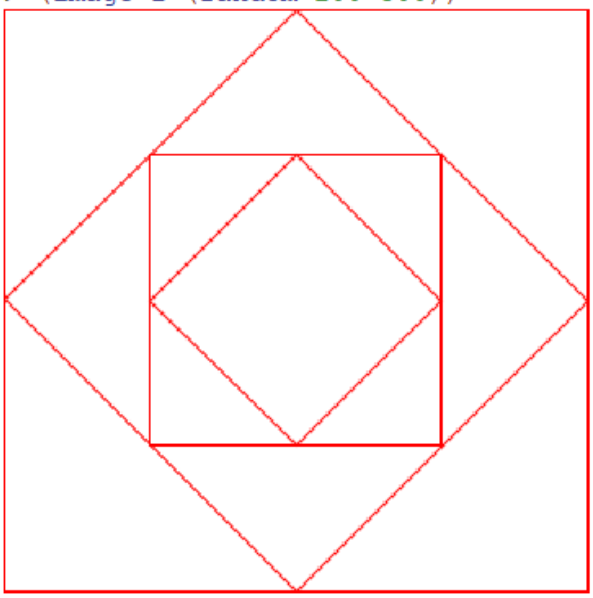


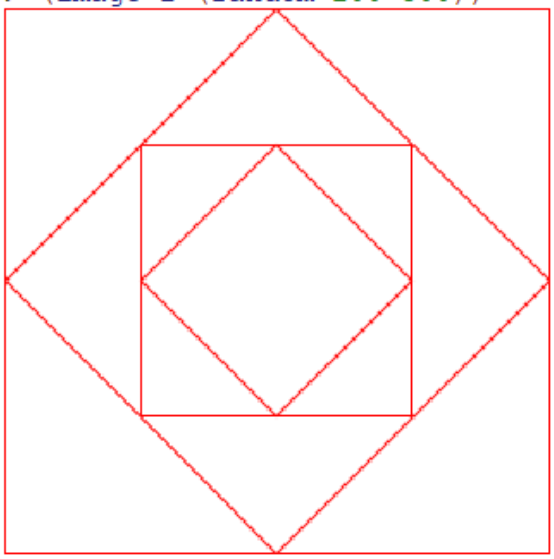
Image-2 demo:

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



Damian Randall

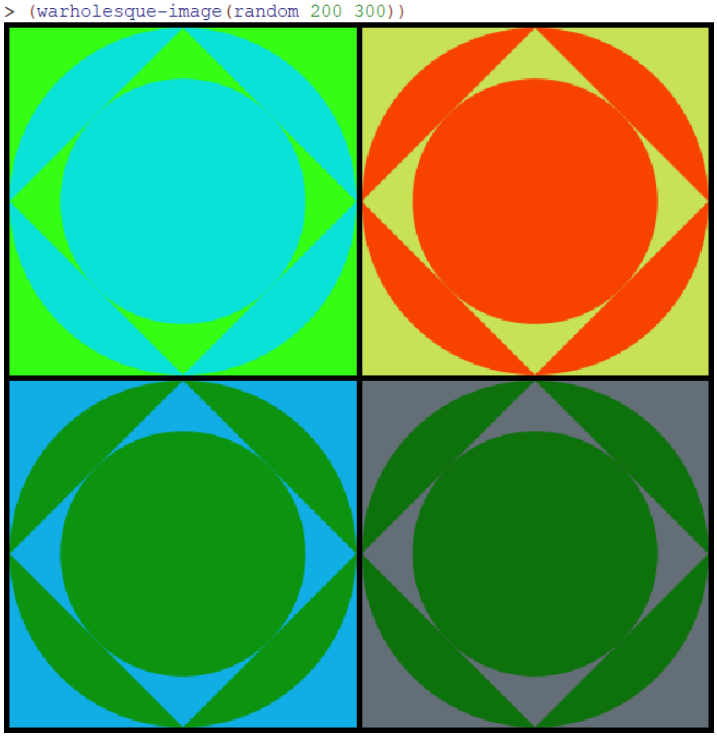
```
> (image-2 (random 200 300))
```



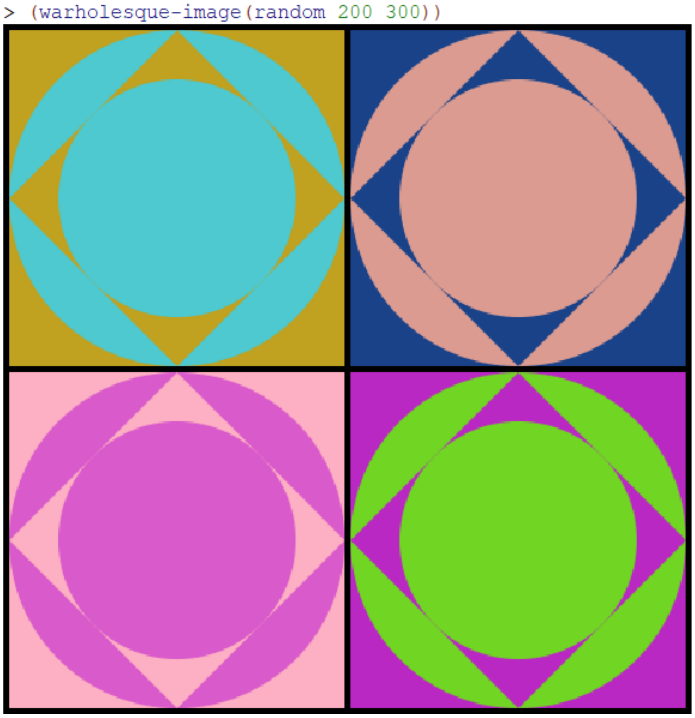
Damian Randall

Warholesque-image demo:

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



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



Damian Randall

Code:

```
(define(image-1 side)
  (define outer-square(square side "solid" "purple"))
  (define outer-radius(ic side))
  (define outer-circle(circle outer-radius "solid" "cyan"))
  (define inner-square-side(is outer-radius))
  (define inner-square(square inner-square-side "solid" "purple"))
  (define inner-square-rotated(rotate 45 inner-square))
  (define inner-radius(ic inner-square-side))
  (define inner-circle(circle inner-radius "solid" "cyan"))
  (define inside(overlay inner-circle inner-square-rotated))
  (define outside(overlay outer-circle outer-square))
  (overlay inside outside)
 )

(define(image-2 side)
  (define outside-square(square side "outline" "red"))
  (define outside-rotated-side(is(/ side 2)))
  (define outside-square-before(square outside-rotated-side "outline" "red"))
  (define outside-square-rotated(rotate 45 outside-square-before))
  (define inside-side(is(/ outside-rotated-side 2)))
  (define inside-square(square inside-side "outline" "red"))
  (define inside-rotated-side(is(/ inside-side 2)))
  (define inside-square-before(square inside-rotated-side "outline" "red"))
  (define inside-rotated(rotate 45 inside-square-before))
  (define inside(overlay inside-rotated inside-square))
  (define outside(overlay outside-square-rotated outside-square))
  (overlay inside outside)

 )



(define (wholeImage side)
  (define (rgb)(random 0 256))
  (define c1(color(rgb)(rgb)(rgb)))
  (define c2(color(rgb)(rgb)(rgb)))

  (define outer-square(square side "solid" c1))
```

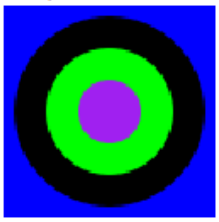
```
  (define outer-radius(ic side))
  (define outer-circle(circle outer-radius "solid" c2))
  (define inner-square-side(is outer-radius))
  (define inner-square(square inner-square-side "solid" c1))
  (define inner-square-rotated(rotate 45 inner-square))
  (define inner-radius(ic inner-square-side))
  (define inner-circle(circle inner-radius "solid" c2))
  (define inside(overlay inner-circle inner-square-rotated))
  (define outside(overlay outer-circle outer-square))
  (overlay inside outside)
)

(define (warholesque-image side)

  (define vertical-boarder(rectangle 4 side "solid" "black"))
  (define horizontal-boarder(rectangle (+ 4 side) 4 "solid" "black"))
  (define long-bottom(rectangle (+ (* side 2) 8) 4 "solid" "black"))
  (define long-right(rectangle 4 (+ (* side 2) 12) "solid" "black"))


  (define image-1(wholeImage side))
  (define image-2(wholeImage side))
  (define image-3(wholeImage side))
  (define image-4(wholeImage side))

  (define top(beside (above horizontal-boarder(beside vertical-boarder image-1)) (above
horizontal-boarder(beside vertical-boarder image-2))))
  (define bottom(beside (above horizontal-boarder(beside vertical-boarder image-3))
(above horizontal-boarder(beside vertical-boarder image-4))))
  (beside(above (above top bottom) long-bottom) long-right)

)
```

Damian Randall

Task 4: Permutations of Randomly Colored Stacked Dots

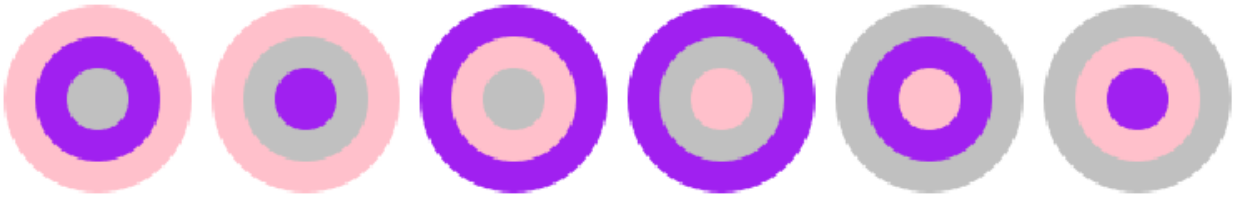
Demo:

```
> (tile "blue" "black" "green" "purple")
```

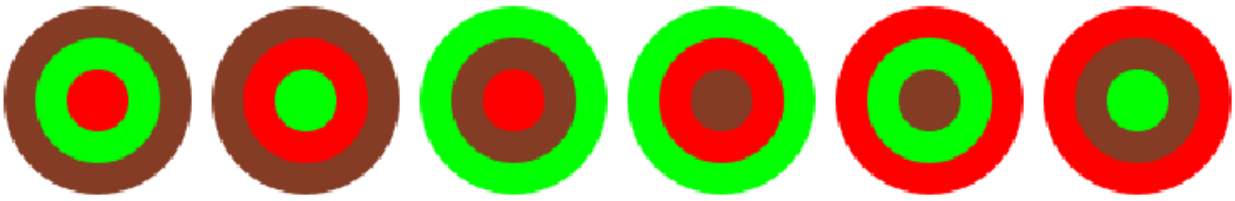


```
> (tile "black" "pink" "purple" "silver")
```

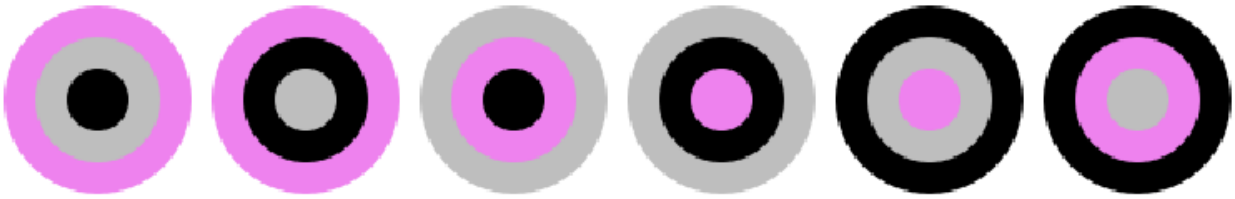
```
> (dots-permutations "pink" "purple" "silver")
```



```
> (dots-permutations "brown" "green" "red")
```

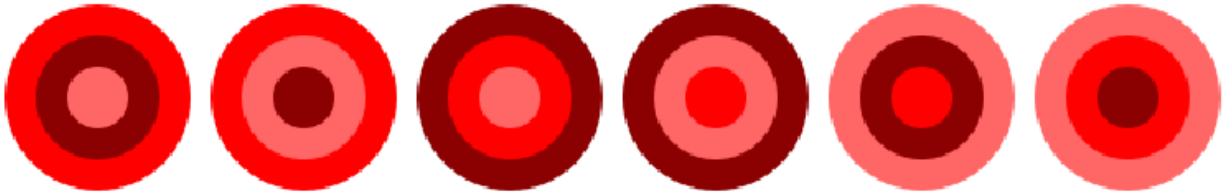


```
> (dots-permutations "violet" "grey" "black")
```



Damian Randall

```
> (dots-permutations "red" "dark red" "light red")
```



Code:

```
(define (tile tile-color c1 c2 c3)

  (define square-tile(square 100 "solid" tile-color))
  (define circle-1(circle (/ 90 2) "solid" c1))
  (define circle-2(circle (/ 60 2) "solid" c2))
  (define circle-3(circle (/ 30 2) "solid" c3))
  (define middle(overlay circle-3 circle-2))
  (define outside(overlay circle-1 square-tile))
  (overlay middle outside)


  )

(define (dots-permutations c1 c2 c3)


  (define cir-1(tile "white" c1 c2 c3))
  (define cir-2(tile "white" c1 c3 c2))
  (define cir-3(tile "white" c2 c1 c3))
  (define cir-4(tile "white" c2 c3 c1))
  (define cir-5(tile "white" c3 c2 c1))
  (define cir-6(tile "white" c3 c1 c2))
  (beside(beside(beside(beside(beside cir-1 cir-2) cir-3) cir-4) cir-5) cir-6)


  )
```