Racket Assignment #2: Interactions, Definitions, Applications

Learning Abstract:

This assignment is for the purpose to get familiar with numeric processing, various interactions, and function definitions. Several programming demos will aid in learning of the above processes.

Task 1: Interactions – 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 9)
12157665459056928801
>
```

Solve a Simple Problem (Area of Scrap)

```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> pi
3.141592653589793
> (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
```

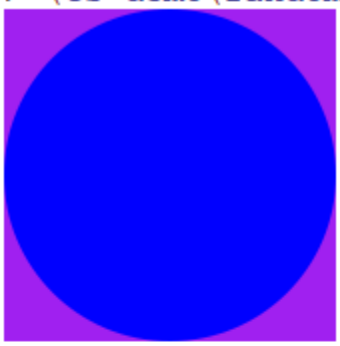
```
>
```

Task 2: Definitions – Inscribing/Circumscribing Circles/Squares

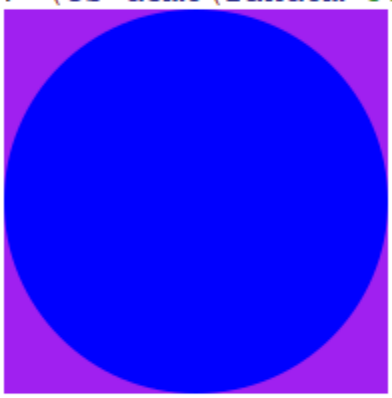
Cs-demo

Language: racket, with debugging; memory limit: 128 MB.

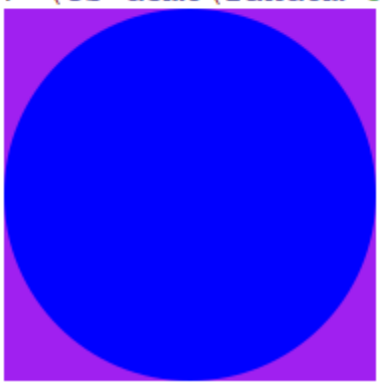
```
> (cs-demo(random 50 150))
```



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



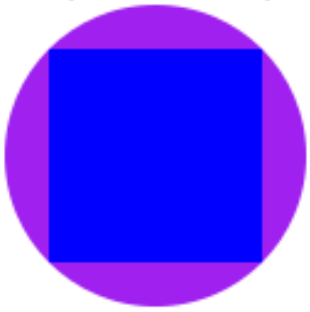
```
> (cs-demo(random 50 150))
```



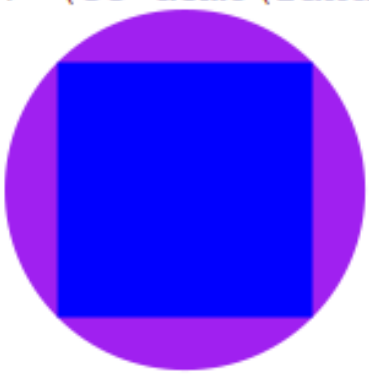
```
> |
```

CC Demo

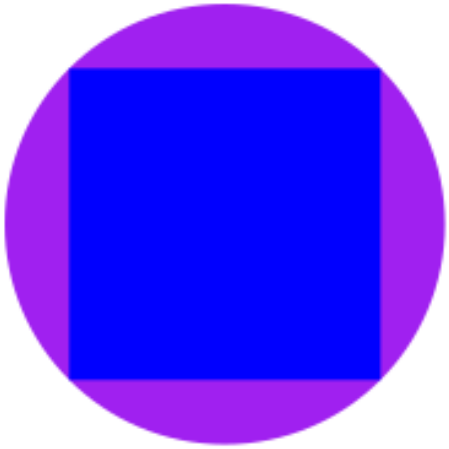
```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (cc-demo(random 50 150))
```



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



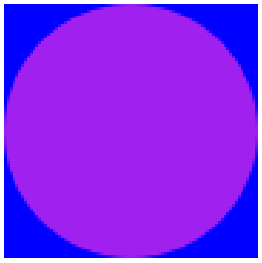
```
> (cc-demo(random 50 150))
```



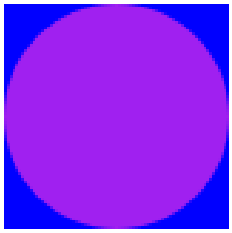
```
>
```

IC Demo

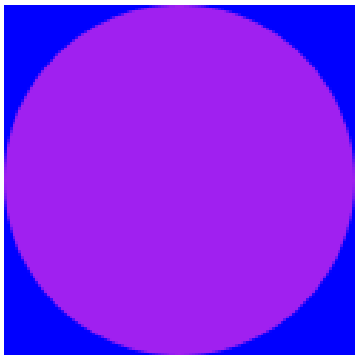
```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (ic-demo(random 50 150))
```



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



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

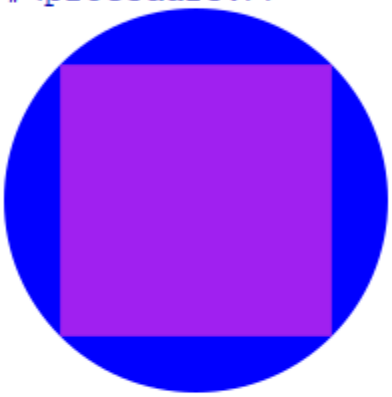


```
>
```

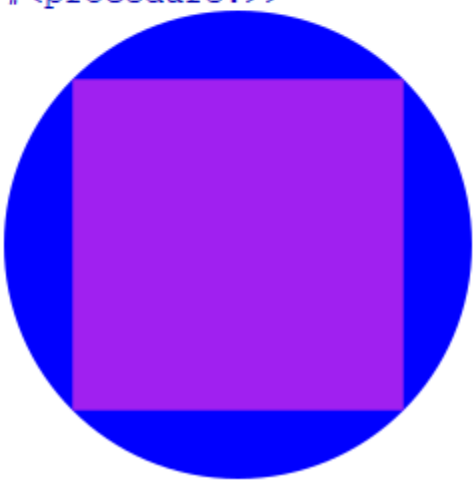
IS Demo

```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB
> (is-demo(random 50 150))
```

```
> > (is-demo(random 50 150))
#<procedure:>>
```



```
> > (is-demo(random 50 150))
#<procedure:>>
```



```
>
```

The Code

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

(define (cs radius)
  ( * radius 2 )
)
(define (cc siLength)
   ( / siLength (sqrt 2))
)
(define (ic siLength)
  (/ siLength 2)
)
(define (is radius)
  ( * radius (sqrt 2))
)
(define (cs-demo radius)
  (define build-square
    (square (cs radius) "solid""purple")
)
(define build-circle
  (circle radius "solid""blue") )
  (overlay build-circle build-square)
)
(define (cc-demo sLength )
  (define build-square
  (square sLength "solid""blue")
)

(define build-circle
  (circle (cc sLength) "solid""purple") )
  (overlay build-square build-circle)
)

(define (ic-demo sLength )
  (define build-square
    (square sLength "solid""blue")
)
(define build-circle
  (circle (ic sLength) "solid""purple") )
  (overlay build-circle build-square)
)
(define (is-demo radius )
  (define build-square
    ( square (is radius) "solid""purple")
)
(define build-circle
  (circle radius "solid""blue") )
  (overlay build-square build-circle)
)
```

Task 3: Inscribing/Circumscribing Images

Image 1 Demo

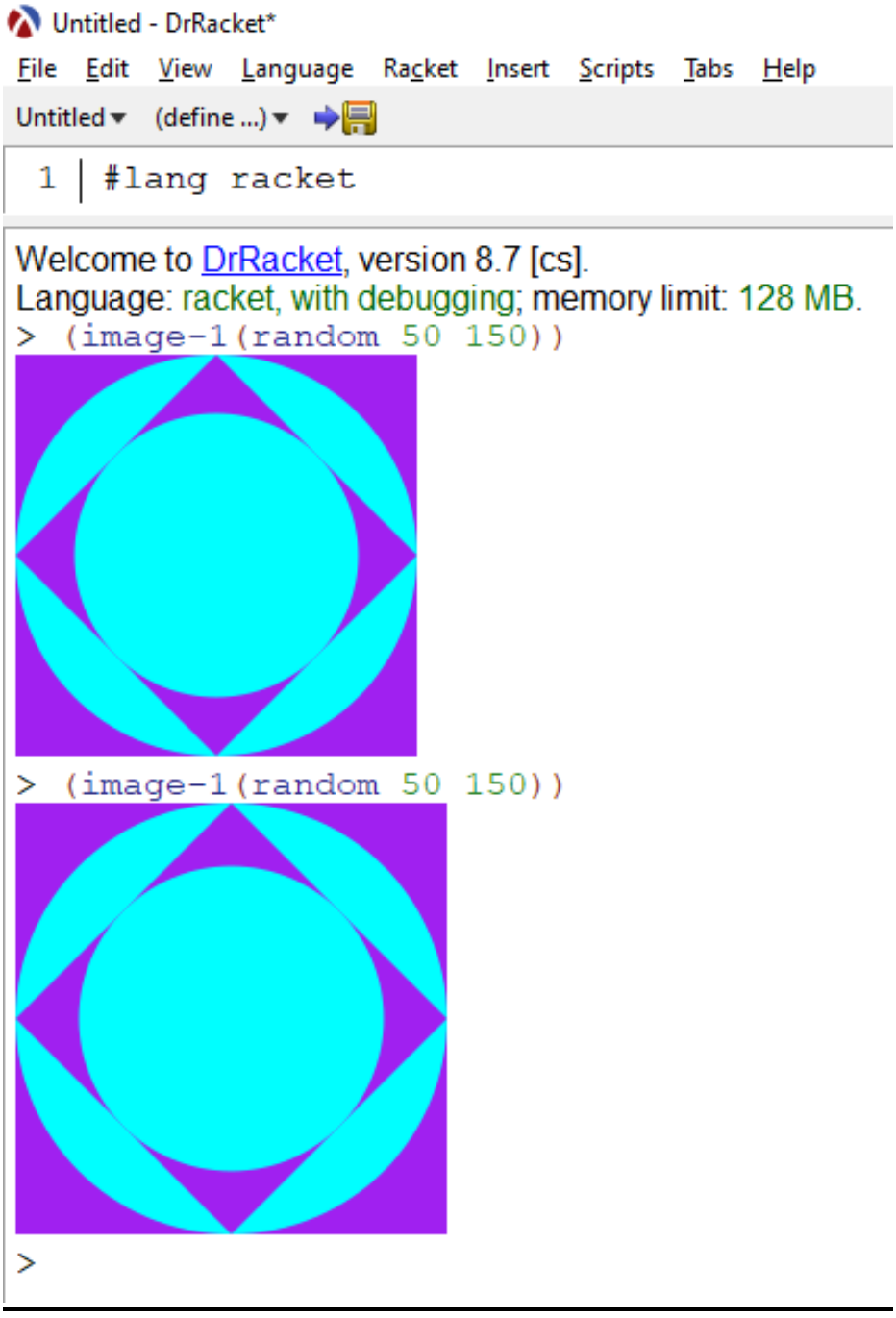
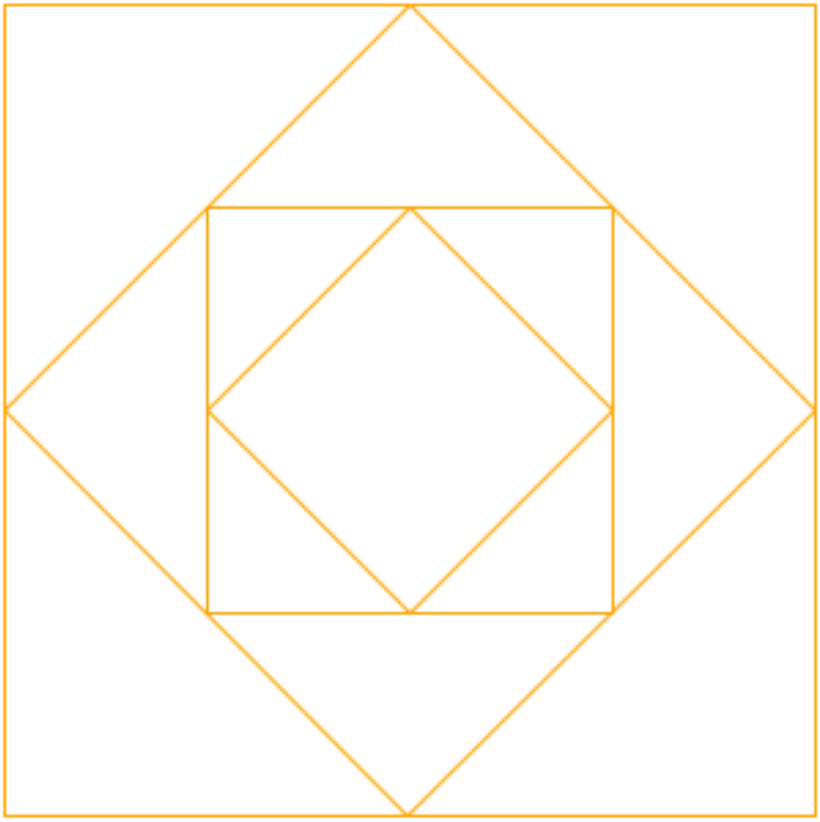


Image 2 Demo

Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> (image-2(random 50 150))
```

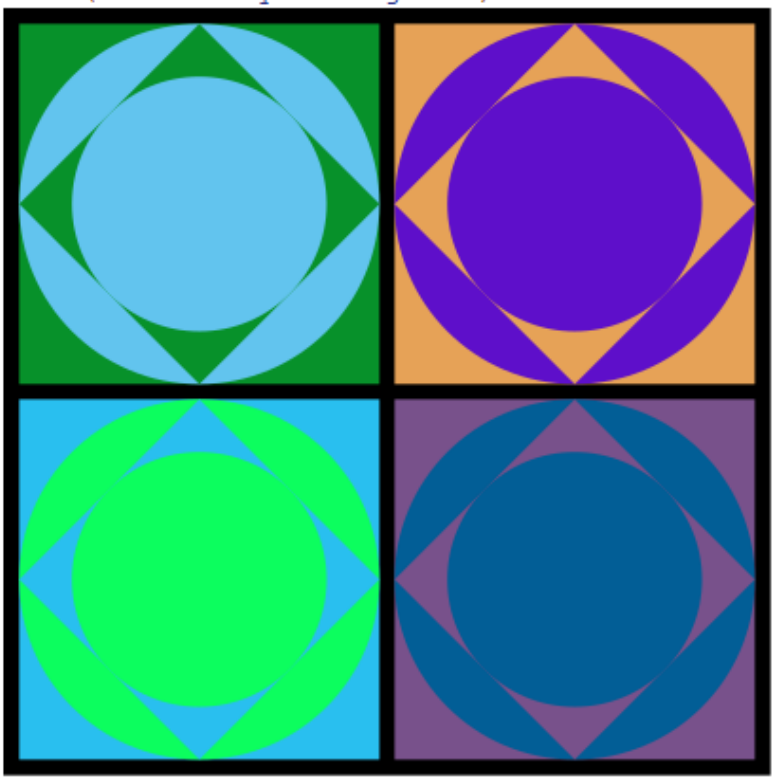
```
> (image-2(random 50 150))
```



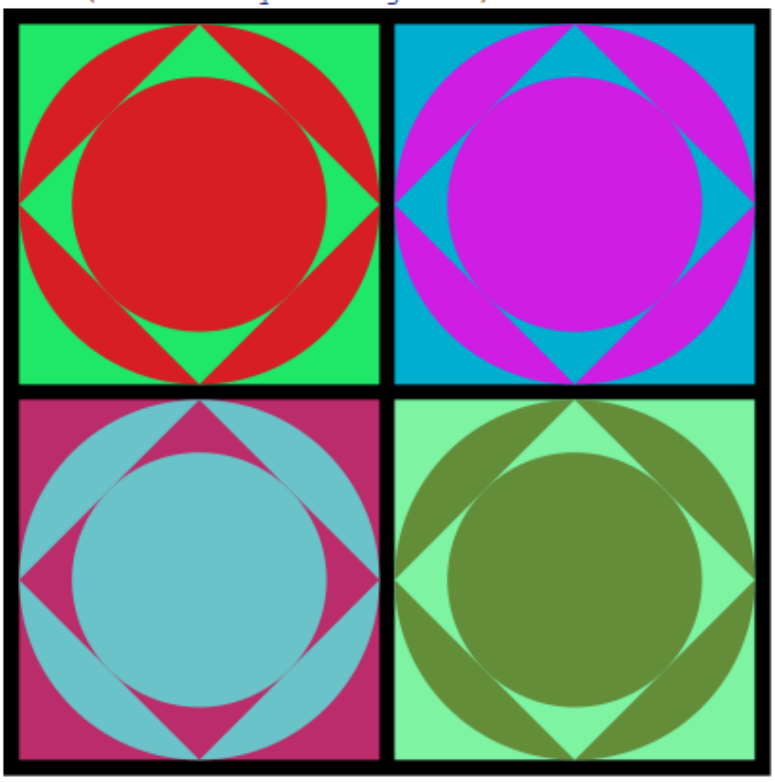
```
>
```

Warholesque Image

```
>    (warholesque-image 95)
```



```
>    (warholesque-image 95)
```



The Code

```
#lang racket
(require 2htdp/image)
(define (cs radius)
 ( * radius 2 ) )
( define (cc side)
 ( / side (sqrt 2) ) )
(define (ic side)
 (/ side 2) )
(define (is radius)
 ( * radius (sqrt 2) ) )
(define (image-1 radius)
  (define build-square( square (cs radius) "solid""purple") )
  (define build-circle(circle radius "solid""cyan") )
  (define build-square2(rotate 45(square (is radius) "solid""purple")))
  (define build-circle2(circle (cc radius) "solid""cyan"))
  (overlay build-circle2 (overlay build-square2(overlay build-circle build-square)))
)
(define (image-2 side)
  (define build-square( square (cs side) "outline""orange") )
  (define build-square2(square side "outline""orange") )
  (define build-square3(rotate 45(square (is side) "outline""orange")))
  (define build-square4(rotate 45(square (cc side) "outline""orange")))
  (overlay build-square4 (overlay build-square3(overlay build-square2 build-square)))
)
(define (image-3 radius)
  (define (rgb)(random 0 256))
  (define (randColor) (color (rgb)(rgb)(rgb)))
  (define squareColor(randColor))
  (define circleColor(randColor))
  (define build-square( square (cs radius)"solid"squareColor) )
  (define build-circle(circle radius "solid"circleColor) )
  (define build-square2(rotate 45(square (is radius) "solid"squareColor)))
  (define build-circle2(circle (cc radius) "solid"circleColor))
  (overlay build-circle2 (overlay build-square2(overlay build-circle build-square)))
)
(define (brdImage1 size)
  (define border(square (cs (+ size 4)) "solid" "black"))
  (overlay (image-3 size) border)
)
(define (warholesque-image size)
  (define border(square (cs (* size 2.13)) "solid" "black"))
  (overlay(above(beside (brdImage1 size)(brdImage1 size))(beside (brdImage1 size)(brdImage1 size))) border)
)

```

Task 4: Permutations of Randomly Colored Stacks of Dots

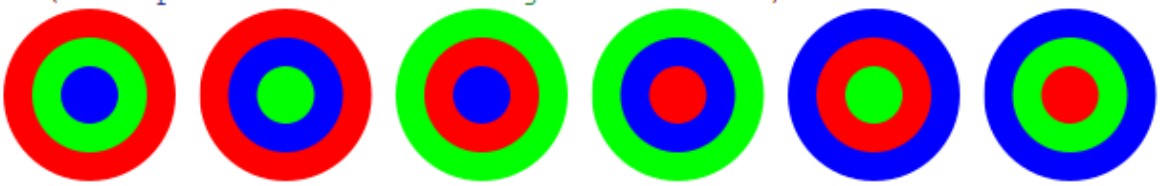
Demo

Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.

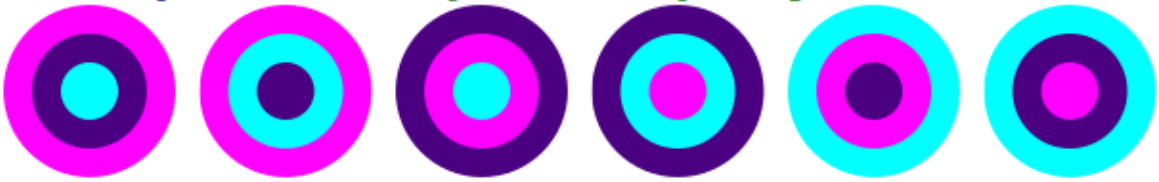
```
> (tile "black" "blue" "teal" "turquoise")
```

```
> (tile "silver" "magenta" "purple" "lavender")
```

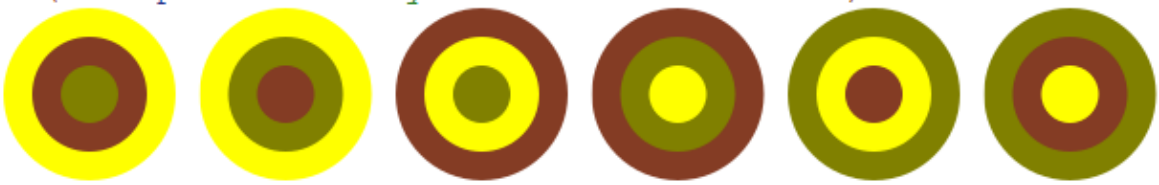
```
> (dots-permutation "red" "green" "blue")
```



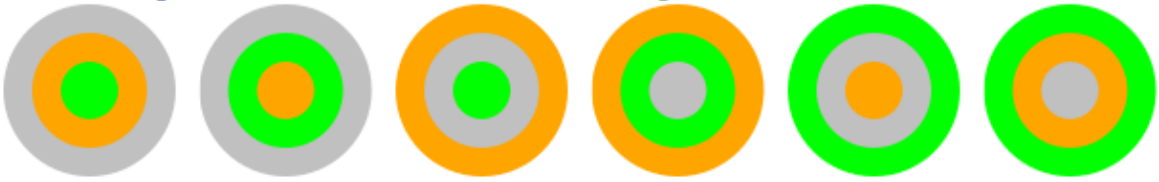
```
> (dots-permutation "magenta" "indigo" "cyan")
```



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



```
> (dots-permutation "silver" "orange" "lime")
```



```
>
```

The Code

```
1  #lang racket
2  (require 2htdp/image)
3
4  (define side 100)
5  (define radius1 (* side .45))
6  (define radius2 (* side .30))
7  (define radius3 (* side .15))
8  (define gap (/ side 8) )
9  (define placeGap(square gap "solid" "white"))
10
11 (define (tile color color1 color2 color3)
12  (overlay
13   (circle radius3 "solid" color3)
14   (circle radius2 "solid" color2)
15   (circle radius1 "solid" color1)
16   (square side "solid" color)
17  )
18 )
19
20
21 (define (dots-permutation color1 color2 color3)
22  (define permutation1
23   (overlay
24    (circle radius3 "solid" color3)
25    (circle radius2 "solid" color2)
26    (circle radius1 "solid" color1)
27   )
28  )
29 (define permutation2
30   (overlay
31    (circle radius3 "solid" color2)
32    (circle radius2 "solid" color3)
33    (circle radius1 "solid" color1)
34   )
35  )
36 (define permutation3
37   (overlay
38    (circle radius3 "solid" color3)
39    (circle radius2 "solid" color1)
40    (circle radius1 "solid" color2)
41   )
42  )
```

```
43 (define permutation4
44   (overlay
45    (circle radius3 "solid" color1)
46    (circle radius2 "solid" color3)
47    (circle radius1 "solid" color2)
48   )
49  )
50 (define permutation5
51   (overlay
52    (circle radius3 "solid" color2)
53    (circle radius2 "solid" color1)
54    (circle radius1 "solid" color3)
55   )
56  )
57 (define permutation6
58   (overlay
59    (circle radius3 "solid" color1)
60    (circle radius2 "solid" color2)
61    (circle radius1 "solid" color3)
62   )
63  )
64 (beside permutation1 placeGap permutation2 placeGap
65         permutation3 placeGap permutation4 placeGap
66         permutation5 placeGap permutation6
67 )
68
69 )
```