Second Racket Programming assignment solution

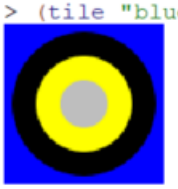
Second Racket Programming assignment solution Learning Abstract: In this assignment I hard coded a function that displayed all possible permutation of a three-layered circle (three disks). I also learned how to make recursive functions through creating number sequences and images that embody the art of two famous modern artist. I then applied that knowledge to make my own image using the 2htdp/image library.

Interactions: Solution to Task 1: Permutations

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

(define (tile squareColor color1 color2 color3)
  (define theSquare (square 100 "solid" squareColor))
  (define firstCircle (circle 45 "solid" color1))
  (define secondCircle (circle 30 "solid" color2))
  (define thirdCircle (circle 15 "solid" color3))
  (define theTile (overlay thirdCircle secondCircle firstCircle theSquare))
  theTile
  )
(define (dots-permutations color1 color2 color3)
  (define circle1 (tile "white" color1 color2 color3))
  (define circle2 (tile "white" color1 color3 color2))
  (define circle3 (tile "white" color2 color1 color3))
  (define circle4 (tile "white" color2 color3 color1))
  (define circle5 (tile "white" color3 color1 color2))
  (define circle6 (tile "white" color3 color2 color1))
  (define permutations (beside circle1 circle2 circle3 circle4 circle5 circle6))
  permutations
)
```

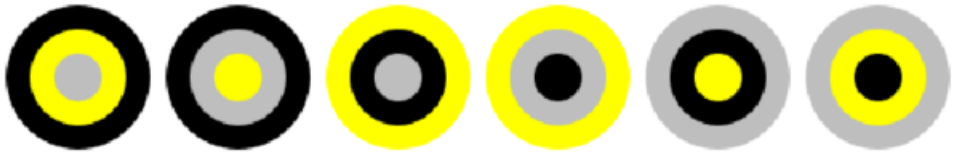
Interactions: Illustration of Task 1: Permutations

```
> (tile "blue" "black" "yellow" "grey")
```



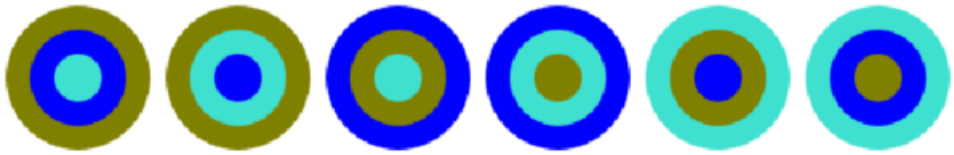
```
> (tile "teal" "orange" "pink" "purple")
```

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

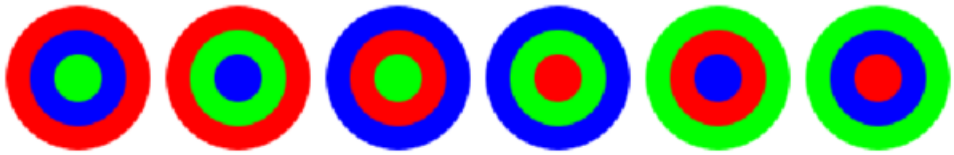


```
> (dots-permutations "orange" "pink" "purple")
```

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



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



```
>
```

Interactions: Solution to Task 2: Number Sequences

```
#lang racket
(define (natural-sequence countTo)
  (cond ((> countTo 0)
         (natural-sequence (- countTo 1))
         (display countTo)(display " ")
         )
   )
 )

(define (copies input numOfTimes)
  (cond ((> numOfTimes 0)
         (copies input (- numOfTimes 1))
         (display input)(display " ")
         )
   )
 )

(define (special-natural-sequence countTo)
  (cond ((> countTo 0)
         (special-natural-sequence (- countTo 1))
         (copies countTo countTo)
         )
   )
 )
```

Interactions: Illustration to Task 2: Number Sequences

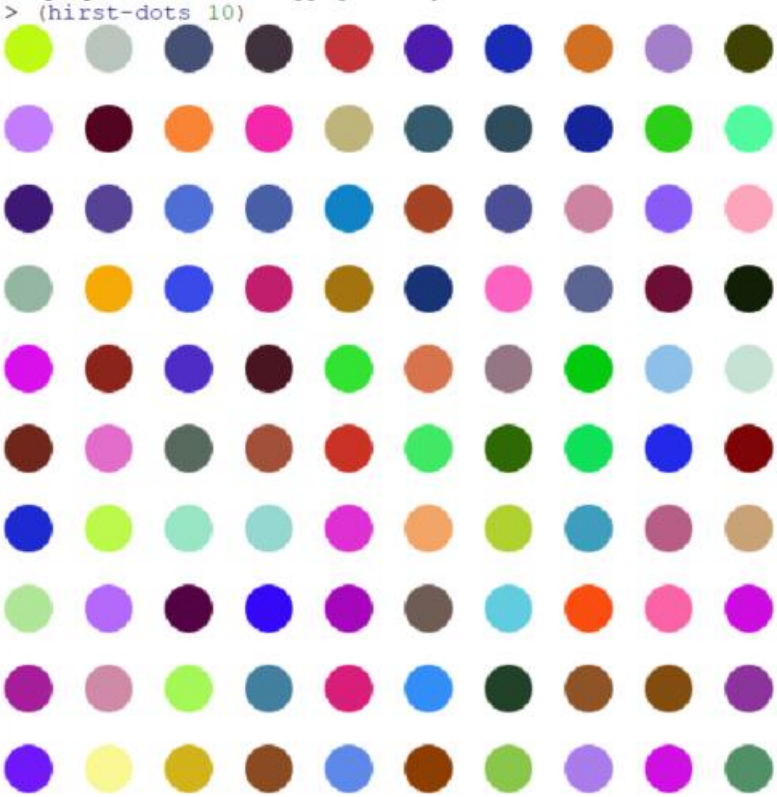
```
>  ( natural-sequence 5 )
1 2 3 4 5
> ( natural-sequence 18 )
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
> ( natural-sequence 10 )
1 2 3 4 5 6 7 8 9 10
> ( natural-sequence 12 )
1 2 3 4 5 6 7 8 9 10 11 12
>  ( copies "a" 11 )
a a a a a a a a a a a
>  ( copies 9 9 )
9 9 9 9 9 9 9 9 9
>  ( copies "hello " 3 )
hello  hello  hello
>  ( copies "goodbye" 4 )
goodbye goodbye goodbye goodbye
>  ( special-natural-sequence 5 )
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
>  ( special-natural-sequence 20 )
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 ↵
10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 11 11 12 12 12 12 12 12 12 12 12 ↵
12 12 12 13 13 13 13 13 13 13 13 13 13 13 13 13 14 14 14 14 14 14 14 14 14 14 14 14 14 14 ↵
15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 ↵
16 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 18 18 18 18 18 18 18 18 18 18 18 ↵
18 18 18 18 18 18 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 20 20 20 20 20 ↵
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
>  ( special-natural-sequence 6 )
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6
>  ( special-natural-sequence 9 )
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9
>
```

Interactions: Solution to Task 3: Hirst Dots

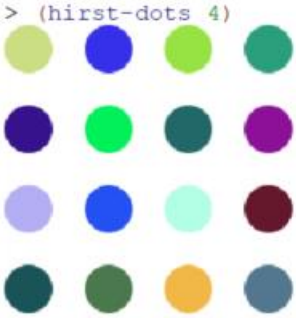
```
#lang racket
(require 2htdp/image)

(define (dot color)
  (circle 15 "solid" color)
 )

(define (rgb-value)
  (random 256)
 )

(define (randomColorDot)
  (dot (color (rgb-value) (rgb-value) (rgb-value)))
 )
(define space ( square 20 "solid" "white"))

(define (dot-row numOfTimes)
  (cond ((> numOfTimes 0)
         (beside (randomColorDot) space (dot-row (- numOfTimes 1)) )
         )
        ((= numOfTimes 0)
         empty-image
        )
   )
 )

(define (hirst height length)
   (cond ((> height 0)
          (above (dot-row length) space (hirst (- height 1) length ))
          )
         ((= height 0)
          empty-image
          )
    )
 )
(define (hirst-dots gridLength)
  (hirst gridLength gridLength)
   )
```

Interactions: Illustration to Task 3: Hirst Dots

```
> (hirst-dots 10)
```



```
> (hirst-dots 4)
```



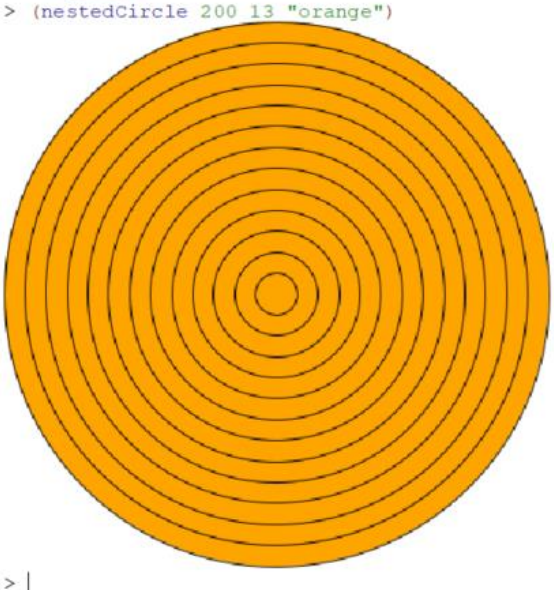
Interactions: Solution to Task 4: Stella Thing

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

(define (framedCircle radius color)
  (overlay
   (circle radius "outline" "black")
   (circle radius "solid" color)
  )
)

(define (paintNestedCircle from to unit color)
  (define radius (* from unit))
  (cond (( = from to)
           (framedCircle radius color)
         )
        ((< from to)
           (overlay
             (framedCircle radius color)
             (paintNestedCircle (+ from 1) to unit color)
            )
         )
   )
 )

(define (nestedCircle side count color)
  (define unit (/ side count))
  (paintNestedCircle 1 count unit color)
 )
```

Interactions: Illustration to Task 4: Stella Thing

```
> (nestedCircle 70 5 "teal")
```

```
> (nestedCircle 200 13 "orange")
```



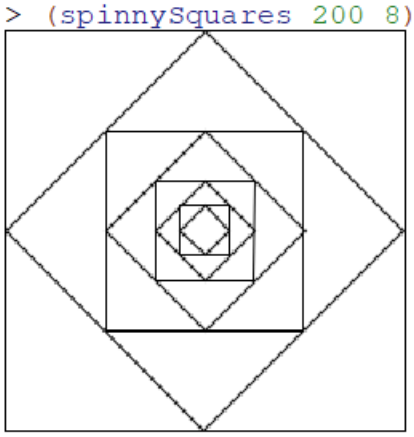
```
> |
```

Interactions: Solution to Task 5: My Creation

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

(define (singleSquare side)
  (square side "outline" "black")
 )

(define (spinnySquares sideLength numOfSquares)
  (cond ((> numOfSquares 0)
         (overlay
          (singleSquare sideLength) (rotate 45 (spinnySquares (/ sideLength (sqrt 2)) (- numOfSquares 1)))
          )
         )
        ((= numOfSquares 0)
          empty-image
         )
   )
 )
```

Interactions: Illustration to Task 5: My Creation

```
> (spinnySquares 200 8)
```



```
>
```