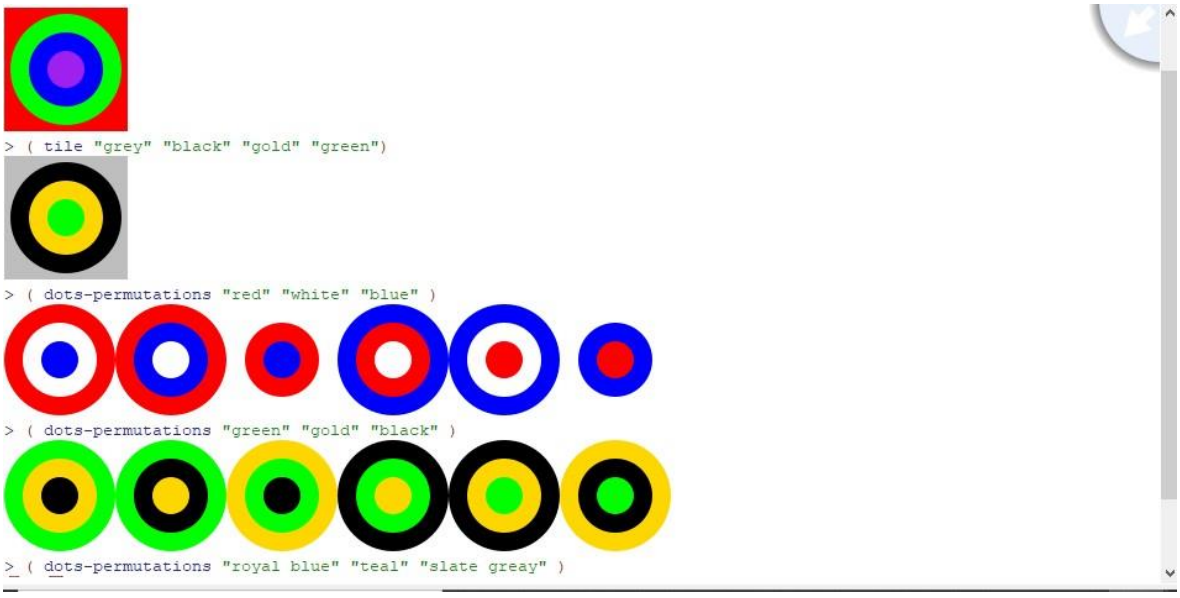
Second Racket Assignment

Learning Abstract: This assignment contained 5 tasks. The first was to create a program the displayed all the possible permutation of a circle with 3 layers. The second used recursive functions by creating number sequences. The third continued with the use of recursive functions to create a grid of dots. In the fourth I had to use the inspiration a Frank Stella to create a rhombus. The final task was using racket to create my own image.

Task 1 Permutations:

```
#lang racket
; Image library
 (require 2htdp/image)

; Make a function tile that takes four parameters
; All the parameters are colors
; The function outputs a square of the color of the first parameter
; There are 3 circles that are overlayed on the sqaure

( define ( colored-square color) (square 100 "solid" color ))
( define (tcircle1 color) (circle 45 "solid" color ))
( define (tcircle2 color) (circle 30 "solid" color ))
( define (tcircle3 color) (circle 15 "solid" color ))

(define (tile c1 c2 c3 c4)
  (overlay (overlay (tcircle3 c4) (tcircle2 c3))
           (overlay (tcircle1 c2) (colored-square c1))))


; Make a function called d0ts-permutations
; 3 color paramters
; create 6 images of 3 circles overlayed on each other
; every image has different order of colors of the 3 circles
( define (first-circle c) (circle 45 "solid" c ) )
( define (second-circle c) (circle 30 "solid" c ) )
( define (third-circle c) (circle 15 "solid" c ) )

( define (target c1 c2 c3)
   (overlay (third-circle c3)
            (overlay (second-circle c2)
                     (first-circle c1))))

( define (dots-permutations c1 c2 c3)
   (beside
    (target c1 c2 c3)
    (target c1 c3 c2)
    (target c2 c1 c3)
    (target c3 c1 c2)
    (target c3 c2 c1)
    (target c2 c3 c1)))
```



Task 2 Number Sequences:

```
#lang racket
( define ( natural-number n)
   ( cond
      ( ( = n 0 )
        ( display "\n")
      )

      ( ( > n 0)
        n
      )
   )
)

( define ( natural-sequence n )
   ( cond
      ( (> n 0)
        ( natural-sequence ( - n 1 ) )
        ( display ( natural-number n) ) ( display " " )
      )
    )
 )

( define ( copies s n )
   ( cond
      ( ( = n 0 )
        ( display "" )
       )

      ( ( > n 0)
        ( display s ) ( display " " )
        ( copies s ( - n 1 ) )
      )
    )
)


( define ( special-natural-sequence n )
   ( cond
      ( ( = n 0 )
        ( display "\n" )
       )
      ( ( > n 0 )
        ( special-natural-sequence (- n 1 ) )
        ( copies n n )

       )
      )
   )
```

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( natural-sequence 15 )
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
> (natural-sequence 7)
1 2 3 4 5 6 7
> (copies "g" 4 )
g g g g
> (copies "A" 7 )
A A A A A A A
> ( special-natural-sequence 4)

1 2 2 3 3 3 4 4 4 4
> ( special-natural-sequence 40)

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 ↵
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 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 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 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 22 22 22 22 22 22 22 22 22 22 22 22 ↵
22 22 22 22 22 22 22 22 22 22 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 24 24 24 24 ↵
24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 ↵
25 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 27 27 27 27 ↵
27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 ↵
28 28 28 28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 ↵
29 29 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 31 31 31 31 31 ↵
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 32 32 32 32 32 32 32 32 32 32 32 32 32 32 ↵
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 ↵
33 33 33 33 33 33 33 33 33 33 33 33 33 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 ↵
34 34 34 34 34 34 34 34 34 34 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 35 ↵
35 35 35 35 35 35 35 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 ↵
36 36 36 36 36 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 ↵
37 37 37 37 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 ↵
38 38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 ↵
39 39 39 39 39 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 ↵
40 40 40 40 40 40 40
>
```

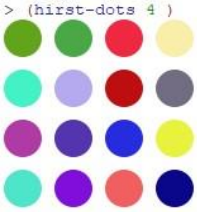
Task 3 Hirst Dots:

```
#lang racket

( require 2htdp/image )

( define ( row-of-dots n )
   ( cond
      ( ( = n 0 )
        empty-image
        )
      ( ( > n 0 )
        ( beside (row-of-dots ( - n 1 ) ) ( random-color-dot ) (square 10 "solid" "white"))
        )
      )
   )

( define ( rgb-value ) ( random 256 ) )
( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )
( define ( random-color-dot ) ( circle 15 "solid" ( random-color ) ) )


( define ( rectangle-of-dots r c )
   ( cond
      ( ( = r 0 )
       empty-image
      )
      ( ( > r 0 )
        ( above ( rectangle-of-dots ( - r 1 ) c ) ( row-of-dots c ) (square 10 "solid" "white" ) )
        )
      )
   )

( define ( hirst-dots n )
   ( rectangle-of-dots n n )
)
```

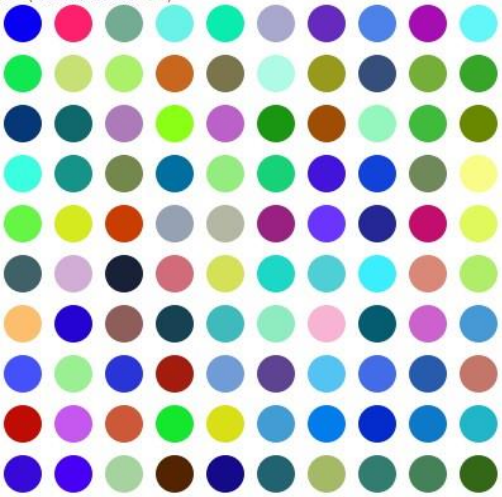
1 bound occurrence

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

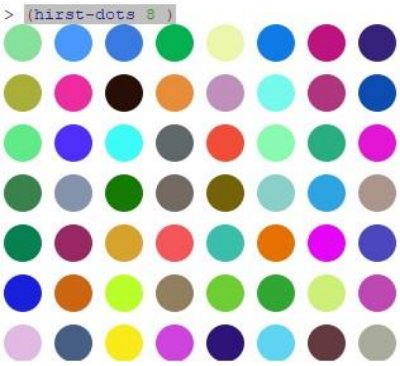
```
> (hirst-dots 4 )
```



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



```
> (hirst-dots 8 )
```

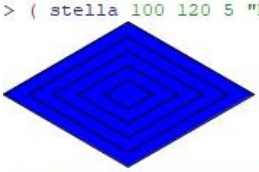


Task 4 Stella:

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

(define ( stella side angle count color )
  ( define unit ( / side count ) )
  ( paint-rhombus 1 count unit angle color )
  )

( define ( paint-rhombus from to unit angle color)
   ( define side-length ( * from unit ) )
   ( cond
      ( ( = from to )
        ( framed-rhombus side-length angle color )
       )
      ( ( < from to )
        ( overlay
          ( framed-rhombus side-length angle color )
          ( paint-rhombus ( + from 1 ) to unit angle color )
         )
       )
      )
   )



( define ( framed-rhombus side-length angle color )
   ( overlay
     (rhombus side-length angle "outline" "black" )
     ( rhombus side-length angle "solid" color )
     )
   )
```

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( stella 100 120 5 "blue" )
```



```
> ( stella 200 45 "red" )
stella: arity mismatch;
 the expected number of arguments does not match the given number
  expected: 4
  given: 3
> (stella 200 45 50 "red" )
```

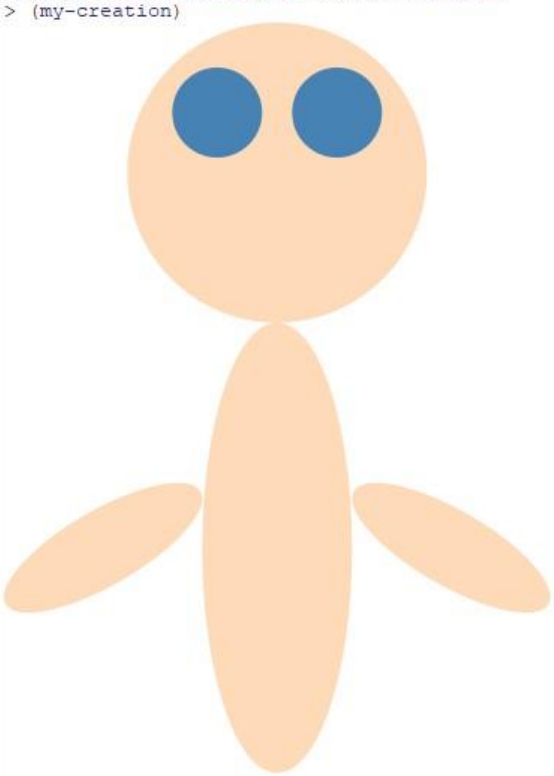
```
>
```

Task 5 My Creation:

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

(define ( head ) ( underlay/offset ( circle 100 "solid" "peach puff" ) 0 -40
                                   (underlay/offset (circle 30 "solid" "steel blue")
                                                    -80 0
                                                    (circle 30 "solid" "steel blue")
                                                    )
                                   )
  )

(define ( body ) (beside ( rotate 30 (ellipse 150 50 "solid" "peach puff" ) )
                         (ellipse 100 300 "solid" "peach puff" )
                         ( rotate -30 (ellipse 150 50 "solid" "peach puff" ) )
                         )
  )

(define (my-creation)
  (above (head) (body))
  )
```

Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> (my-creation)
```



>