Second Racket Programming Assignment Specification

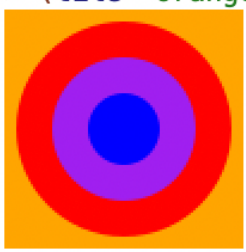
In this assignment I learned a little bit about the Racket 2htdp/image library. I also learned about calling recursive functions in Racket.

Task 1 - Permutations of Randomly Colored Stacked Dots

Programming constraint: For this part of your assignment, you are not permitted to use any form of repetition (recursion/iteration) or any form of conditional statement (e.g., if, cond).

```
> (define (tile c1 c2 c3 c4)
        (define s (square 100 "solid" c1))
        (define circle1 (circle 45 "solid" c2))
        (define circle2 (circle 30 "solid" c3))
        (define circle3 (circle 15 "solid" c4))
        (overlay circle3 circle2 circle1 s)
   )
> (tile "black" "red" "white" "blue")
```

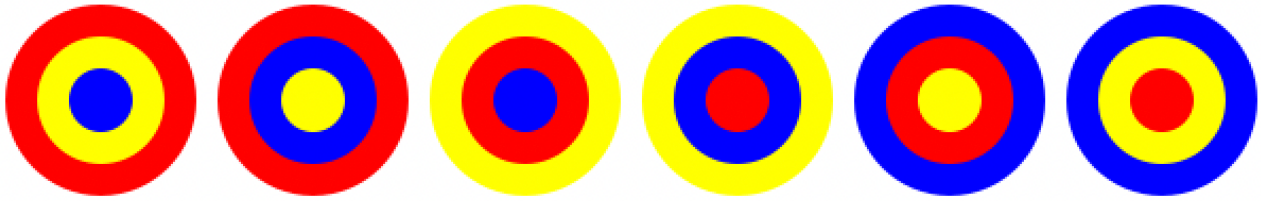
```
> (tile "orange" "red" "purple" "blue")
```



```
>
```

Task 1 (Continued) - Permutations of Randomly Colored Stacked Dots

```
> (define (dots-permutations c1 c2 c3)
    (define perm1 (tile "white" c1 c2 c3))
    (define perm2 (tile "white" c1 c3 c2))
    (define perm3 (tile "white" c2 c1 c3))
    (define perm4 (tile "white" c2 c3 c1))
    (define perm5 (tile "white" c3 c1 c2))
    (define perm6 (tile "white" c3 c2 c1))
    (beside perm1 perm2 perm3 perm4 perm5 perm6)
  )
> (dots-permutations "red" "yellow" "blue")
```



```
>
```

Task 2 - Number Sequences

Programming constraint: For this part of your assignment, you are are not permitted to use any form of iterative construct. Rather, you are required to use recursion.

```
> (define (natural-sequence n)
    (cond
      ((= n 1) (display 1) (display " "))
      ((> n 1) (natural-sequence (- n 1)) (display n) (display " "))
    )
  )
> (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
>
```

```
> (define (copies word num)
    (cond
      ((= num 1) (display word) (display " "))
      ((> num 1) (display word) (display " ") (copies word (- num 1)))
    )
  )
> (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
>
```

Task 2 (Continued) - Number Sequences

```
> (define (special-natural-sequence n)
    (cond
     ((= n 1) (display "1 "))
     ((> n 1) (special-natural-sequence (- n 1)) (copies n n))
    )
  )
> (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 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 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 16 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 18
18 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 20
```

Task 3 - Hirst Dots

Programming constraint: For this part of your assignment, you are are not permitted to use any form of iterative construct. Rather, you are required to use recursion.

```
> ( require 2htdp/image )
> ( define ( rgb-value ) ( random 256 ) )
> ( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )
> (define (square-with-dot)
    (define dot ( circle 15 "solid" ( random-color )))
    (overlay dot (square 40 "solid" "white"))
)

> (define (hirst-row num)
    (cond
      ((= num 1) (square-with-dot))
      ((> num 1) (beside (square-with-dot) (hirst-row (- num 1))))
    )
  )
> (define (hirst-box row col)
    (cond
      ((= col 1) (hirst-row row))
      ((> col 1) (above (hirst-box row (- col 1)) (hirst-row row)))
    )
  )
> (define (hirst-dots num)
    (hirst-box num num)
   )
> (hirst-dots 4)
```

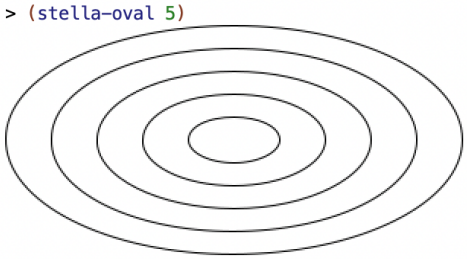
```
> (hirst-dots 10)
```

Task 4 - Stella Thing

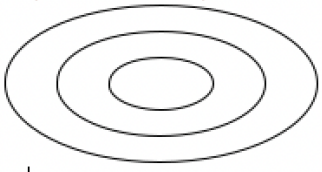
Programming constraint: For this part of your assignment, you are not permitted to use any form of iterative construct. Rather, you are required to use recursion.

```
> (require 2htdp/image)

> (define (stella-oval n)
    (cond
      ((= n 1) (ellipse 60 30 "outline" "black"))
      ((> n 1) (overlay (stella-oval (- n 1)) (ellipse (* 60 n) (* 30 n) "outline" "black")))
     )
  )
> (stella-oval 5)
```



```
> (stella-oval 3)
```



```
> 
```

Task 5 - Creation

Programming constraint: For this part of your assignment, your are not permitted to use any form of of iterative construct. Rather, you are required to use recursion for any repetition that you would like to accomplish.

```
> (define (rand-polygons n)
    (cond
      ((= n 1) (regular-polygon (* n 30) 5 "solid" (random-color)))
      ((> n 1) (overlay (rand-polygons (- n 1)) (regular-polygon (* n 30) 5 "solid" (random-color)))
      )
    )
   )
> (rand-polygons 5)
```