Second Racket Programming assignment

Learning Abstract

I learned how to instead of making a bunch of separate shapes, printing them separately, then printing them together, which takes up an unnecessary amount of time, I instead, put all the functions in the definitions table, then create a method using those functions. This cuts the time it takes to work on a problem in half!

Tile

Write a program called tile which takes four parameters, each presumed to represent a color, which creates an image representing a square tile of side 100 with background defined by the first color, on which are con-centrically piled a disk of diameter 90 of the second color, a disk of diameter 60 of the third color, and a disk of diameter 30 of the fourth color. If the words are overwhelming, just look at the examples presented in the accompanying demo.

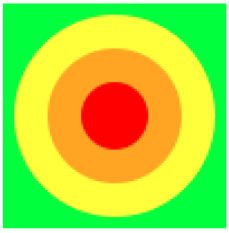
Definitions:

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

(define the-sqaure 100)
(define diameter 90)
(define diameter2 60)
(define diameter3 30)

(define radius ( / diameter 2))
(define radius2 ( / diameter2 2))
(define radius3 ( / diameter3 2))

(define (a-square h)
  (square the-sqaure "solid" h)
  )

(define (dot h)
  (circle radius "solid" h)
  )

(define (dot2 h)
  (circle radius2 "solid" h)
  )
```

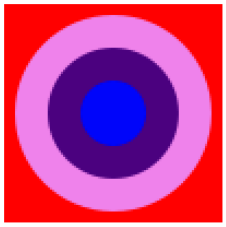
```
(define (dot3 h)
  (circle radius3 "solid" h)
  )

( define (tile a b c d)
   ( overlay
     ( dot3 a )
     ( dot2 b )
     ( dot c )
     ( a-square d)
     )
   )
```

Interactions:

```
> (tile "red" "orange" "yellow" "green")
```



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( tile "blue" "indigo" "violet" "red" )
```



```
>
```

Dots-permutations

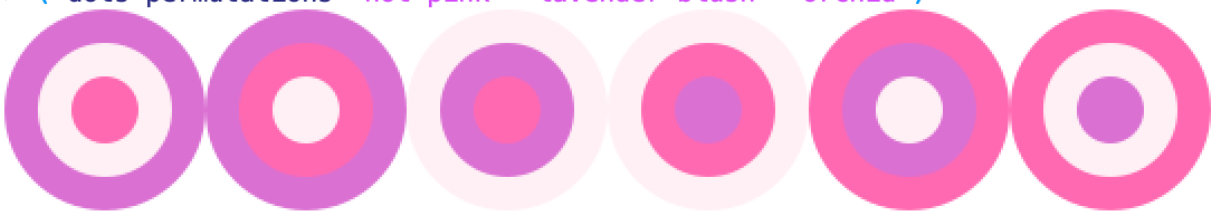
Write a program called dots-permutations taking three parameters, each presumed to represent a color, which creates a row of tiles representing the permutations of three colors, where each permutation is rendered as a stack of dots of diameters 90, 60, and 30. Please look to the accompanying demo for clarification.

Definitions:

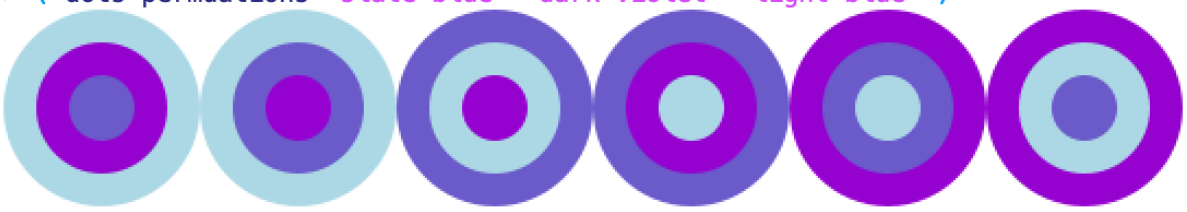
```
#lang racket
(require 2htdp/image)

(define the-sqaure 100)
(define diameter 90)
(define diameter2 60)
(define diameter3 30)

(define radius ( / diameter 2))
(define radius2 ( / diameter2 2))
(define radius3 ( / diameter3 2))

(define (dot h)
  (circle radius "solid" h)
  )
(define (dot2 h)
  (circle radius2 "solid" h)
  )
(define (dot3 h)
  (circle radius3 "solid" h)
  )

(define (target a b c )
  (overlay
   ( dot3 a )
   ( dot2 b )
   ( dot c )
   )
  )



( define ( dots-permuations a b c )
   ( beside
     ( target a b c )
     ( target b a c )
     ( target b c a )
     ( target c b a )
     ( target c a b )
     ( target a c b )
     )
   )
```

Interactions:

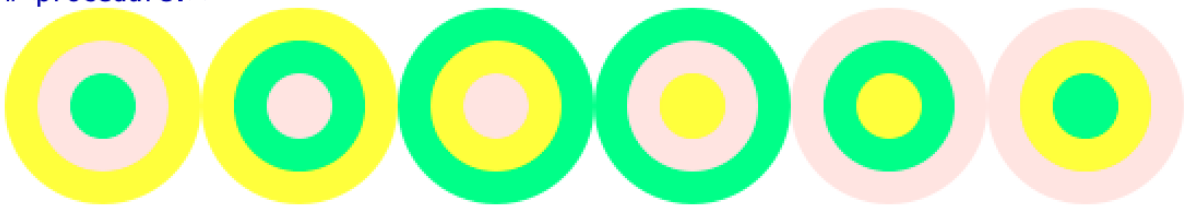
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( dots-permutations "hot pink" "lavender blush" "orchid")
```



```
>
> ( dots-permuations "slate blue" "dark violet" "light blue" )
```



```
> > ( dots-permuations "spring green" "misty rose" "yellow" )
#<procedure:>>
```



Number Sequences

Write a program called natural-sequence that behaves in the manner suggested by the function of the same name in the accompanying demo. Write a program called copies that behaves in the manner suggested by the function of the same name in the accompanying demo. Write a program called natural-sequence that behaves in the manner suggested by the function of the same name in the accompanying demo. Please generate a demo that is just like the demo provided, except that it includes two more applications of each of the three programs. Thus, your demo will have four applications of natural-sequence (two of mine and two of yours), four applications of copies (two of mine and two of yours), and four applications of special-natural-sequence (two of mine and two of yours).

Definitions:

```
#lang racket

( define ( natural a b )
   ( cond
      ( ( <= b a )
        ( display b )
        ( display " " )
        ( natural a ( + b 1 ) )
        )
      )
   )

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


( define ( copies a b )
   ( cond
     ( ( > b 0 )
      ( display a )
      ( display " " )
      ( copies a ( - b 1) ) )
   )
)
   )

( define (special-natural a b )
( cond
   ( ( <= b a )
     ( copies b a )
     ( special-natural a ( + b 1 ) )
     )
   )
)


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

Interactions

```
> ( 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 40 )
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
> ( natural-sequence 8 )
1 2 3 4 5 6 7 8
> ( 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 "k" 4)
k k k k
> ( copies 15 15 )
15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
>
```

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( special-natural-sequence 5 )
1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5
> ( special-natural-sequence 20 )
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
12 12 12 12 12 12 12 12 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 13 13 13 13 13 13 14 14 14 14 14 14 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 15 15 15 15 16 16 16 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 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 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 10 )
1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5
5 5 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 8 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
> ( special-natural-sequence 8 )
1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 8 8 8 8 8 8 8
>
```

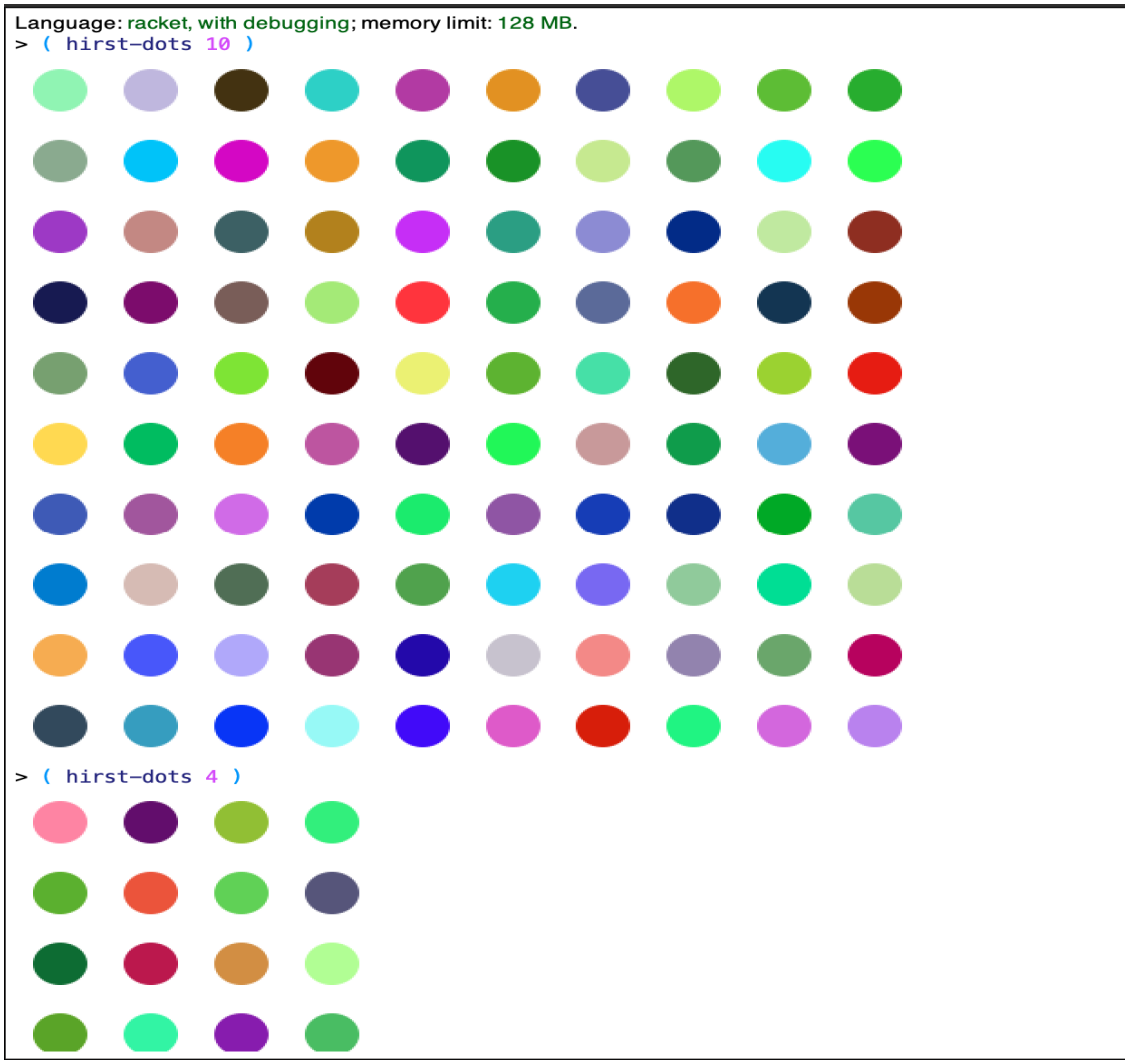
Hirst Dots

Write a function called hirst-dots to draw square arrangements of Hirst dots that are consistent with the accompanying demo. Please arrange for the diameter of the dots to be 30 pixels, and for each dot to be 20 pixels from its nearest dots. Please note that the parameter is presumed to be a non-negative integer, and that the number of dots in an image is the square of the value of the parameter.

Definitions:

```
#lang racket

(require 2htdp/image)
(define (rbg-value) ( random 256))
( define (random-color) (color (rbg-value) (rbg-value) (rbg-value)))


( define (dot color)
   (define background (square 50 "solid" "white"))
   ( define dot ( circle 15 "solid" color ))
   (overlay dot background )
   )

( define (hirst-row n)
   (cond
     ( ( = n 0 )
          empty-image
          )
      (( > n 0 )
       (beside (hirst-row ( - n 1 ) ) ( dot (random-color ) ) )
       )
      )
     )

   ( define (hirst-rectangle r c )
   ( cond
      (( = r 0 )
       empty-image
       )

       (( > r 0)
        ( above (hirst-rectangle ( - r 1 ) c ) ( hirst-row c ) )
        )
       )
   )


   ( define (hirst-dots a)
      (hirst-rectangle a a )
      )
```

Interactions:



Stella thing

Write a function called stella to display graphical images in the spirit of Frank Stella subject to the following constraints:
Your program must be based on a shape other than either of those that I used in the Stella section Lesson

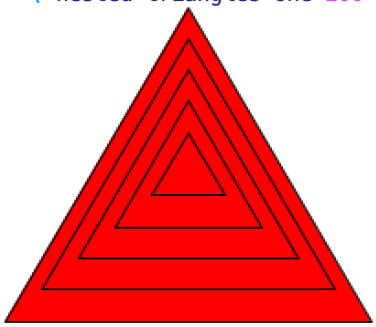
Thus, you must not use either a square or a star for it. Perhaps you would like to use a circle, or an ellipse, or a wedge, or a triangle, or rhombus, or a regular polygon, or a star-polygon. Maybe something else! Please find your way to the documentation for the 2htdp/image library and find some functionality with which to do the deed:

Definitions:

```
#lang racket

(require 2htdp/image )

( define ( nested-triangles-one side count color )
   ( define unit ( / side count ) )
   ( paint-nested-triangles-one 1 count unit color )
   )
( define ( paint-nested-triangles-one from to unit color)
   ( define side-length ( * from unit ) )
   ( cond
      ( ( = from to )
        ( framed-triangle side-length color )
        )
      ( ( < from to )
        ( overlay
          ( framed-triangle side-length color )
          ( paint-nested-triangles-one ( + from 1 ) to unit color )
          )
        )
      )
   )
( define ( framed-triangle side-length color )
   ( overlay
     ( triangle side-length "outline" "black" )
     ( triangle side-length "solid" color )
     )
   )
```

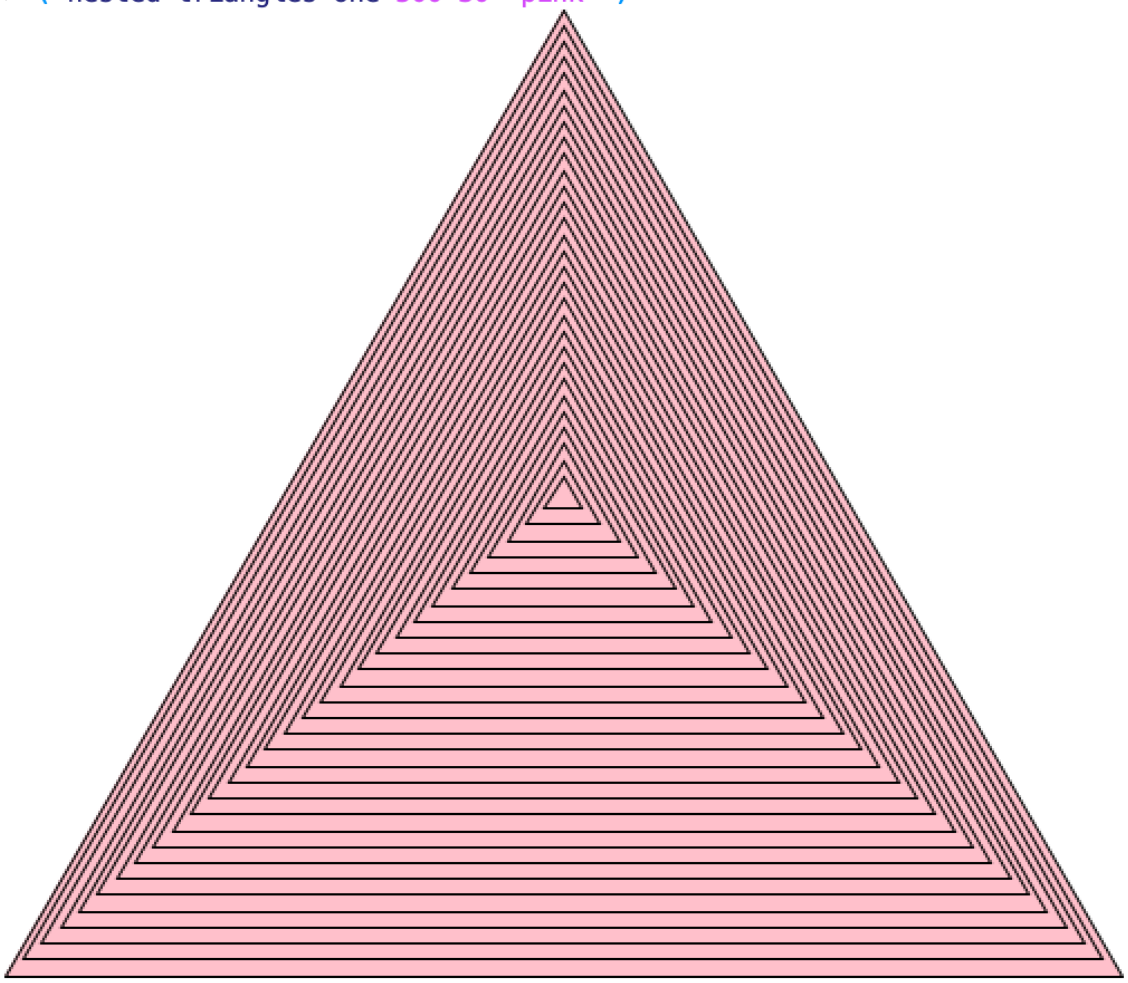
Interactions

```
Language: racket, with debugging; memory limit: 128 MB.
> ( nested-triangles 200 5 "red")
nested-triangles: undefined;
 cannot reference an identifier before its definition
> ( nested-triangles-one 200 5 "red" )
```



```
> ( nested-triangles-one 500 30 "pink" )
```

```
> ( nested-triangles-one 500 30 "pink" )
```



Creation

Please define a function called my-creation to produce an interesting image using some of the more involved functionality in the 2htdp/image library. Perhaps use some of the more elaborate overlay commands (maybe overlay/offset or overlay/xy), or maybe some of the rotating, scaling, flipping, cropping, and framing functionality.

Definitions

```
#lang racket

(require 2htdp/image )

( define the-square 500)

( define (background h)
   ( square the-square "solid" h)
   )

( define (moon h)
   ( radial-star 40 30 90 "solid" h )
   )

( define (the-star a )
   ( star-polygon 30 5 2 "solid" a ))

( define (the-star2 a)
   ( star-polygon 30 5 2 "solid" a ))
```

```
( define (the-star3 a)
   ( star-polygon 30 5 2 "solid" a ))

( define (the-star4 a)
   ( star-polygon 30 5 2 "solid" a ))

( define (the-star5 a)
   ( star-polygon 30 5 2 "solid" a ))

( define (the-star6 a)
   ( star-polygon 30 5 2 "solid" a ))

( define (the-star7 a)
   ( star-polygon 30 5 2 "solid" a ))

( define (the-star8 a)
   ( star-polygon 30 5 2 "solid" a ))
```

```
( define (stars a b)
   ( beside
     (the-star a)
     ( the-star2 b)
      (the-star3 a)
      (the-star4 b)
      (the-star5 a)
      (the-star6 b)
      (the-star7 a)
      (the-star8 b)
      )
   )

( define ( objects a b c)
   ( above
     ( stars a b )
     ( moon c )
     )
   )

( define (creation a b c d)

   ( overlay
     (objects a b c)
     ( background d)
     )
   )
```

Interactions

```
> ( creation "pink" "green" "yellow" "blue" )
```