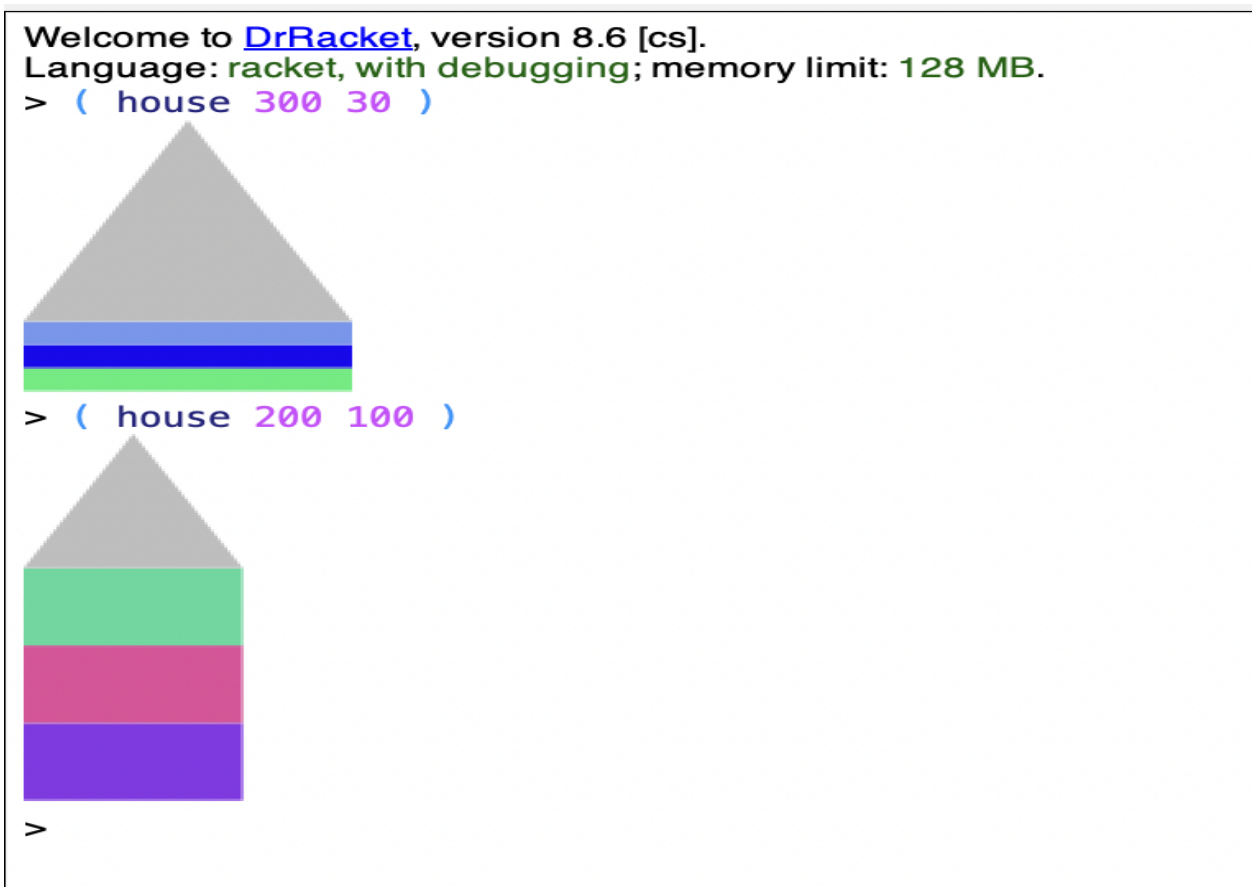
Racket Assignment #2: Racket Functions and Recursions

Learning Abstract:
This task highlights programs that produce images with the 2htdp library which the vast majority are recursive. Each aspect of the tasks provided features different shapes and numerical sequences. For example the use of the dice roll generates values that are ranged from even to odd numbers. This assignment widening the opportunity to experiment with a variety of elements to generate desired creation outcomes.

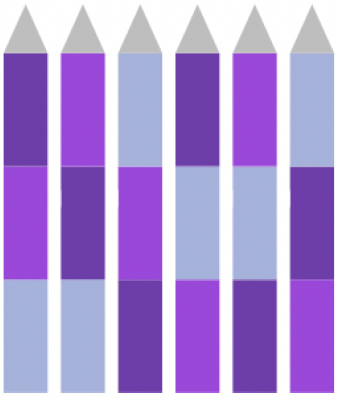
Task 1: Colorful Permutations of Tract House–

Demo for house

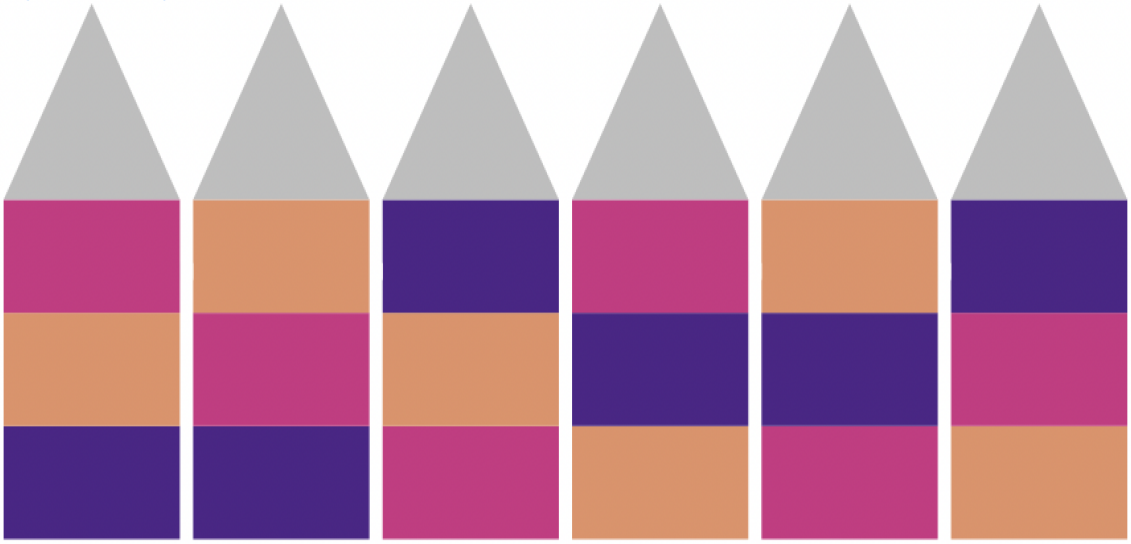


Demo for Tract

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( tract 100 200)
```



```
> ( tract 400 200)
```



```
>
```

The code...

```
#lang racket

( require 2htdp/image )


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

(define (house-rectangle width height)
  (rectangle width height "solid" ( random-color) )
  )



  ( define ( house width height )
( define width-of-house ( / width 3 ) )
( define height-of-house ( / height 3 ) )
( define first-floor ( house-rectangle width-of-house height-of-house ) )
( define second-floor ( house-rectangle width-of-house height-of-house ) )
( define third-floor ( house-rectangle width-of-house height-of-house ) )
( define roof-of-house ( triangle width-of-house "solid" "grey") )
( define full-house ( above roof-of-house first-floor second-floor third-floor ) )
 full-house
)


  ( define ( tract width height )
( define width-of-house ( / width 3 ) )
( define height-of-house ( / height 3 ) )
( define first-floor ( house-rectangle width-of-house height-of-house ) )
( define second-floor ( house-rectangle width-of-house height-of-house ) )
( define third-floor ( house-rectangle width-of-house height-of-house ) )
( define roof-of-house ( triangle width-of-house "solid" "grey") )
( define h1 ( above roof-of-house first-floor second-floor third-floor ))
( define h2 ( above roof-of-house second-floor first-floor third-floor ))
( define h3 ( above roof-of-house third-floor second-floor first-floor ))
( define h4 ( above roof-of-house first-floor third-floor second-floor ))
( define h5 ( above roof-of-house second-floor third-floor first-floor ))
( define h6 ( above roof-of-house third-floor first-floor second-floor ))
( define white-space ( square 10 "solid" "white" ))
( define full-tract ( beside h1 white-space h2 white-space h3 white-space h4 white-space h5 white-space h6 ) )
 full-tract
)
```

Task 2: Dice

Demo...

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( roll-die )
0
> ( roll-die )
4
> ( roll-die )
1
> ( roll-die )
0
> ( roll-die )
5
> ( roll-for-1)
1
> ( roll-for-1)
5 3 1
> ( roll-for-1)
3 1
> ( roll-for-1)
1
> ( roll-for-1)
1
> ( roll-for-1)
2 5 1
> ( roll-for-11 )
3 0 2 2 3 0 2 0 1 2 0 3 1 1
> ( roll-for-11 )
4 1 1
> ( roll-for-11 )
5 3 2 2 1 1
> ( roll-for-11 )
1 4 4 2 5 3 3 3 1 1
> ( roll-for-11 )
3 3 5 5 0 1 5 3 2 5 4 0 0 1 3 2 1 5 5 1 5 5 2 2 4 4 1 5 2 4 0 1 0 1 4 4 4 1 4
3 3 1 5 3 5 2 1 0 5 0 3 5 2 1 3 4 2 5 5 2 3 5 1 3 2 5 2 1 1
> ( roll-for-odd-even-odd )
2 2 2 3 5 1 3 0 3
> ( roll-for-odd-even-odd )
1 4 3
> ( roll-for-odd-even-odd )
3 0 4 0 2 1 0 1
> ( roll-two-dice-for-a-lucky-pair )
( 5 0 ) ( 2 3 ) ( 3 5 ) ( 0 2 ) ( 3 4 ) #t
> ( roll-two-dice-for-a-lucky-pair )
```

Code.....

1: Untitled | 2: Untitled 2 | 3: Untitled 4

```
#lang racket

( define ( roll-die ) ( random 6 ) )



( define ( roll-for-1 )
   ( define outcome ( roll-die ) )
   ( display outcome ) ( display " " )
   ( cond
      ( ( not ( eq? outcome 1 ) )
        ( roll-for-1 )
        )
      )
   )



( define ( roll-for-11 )
   ( roll-for-1 )
   ( define outcome ( roll-die ) )
   ( display outcome ) ( display " " )
   ( cond
      ( ( not ( eq? outcome 1 ) )
        ( roll-for-11 )
        )
      )
   )


( define ( roll-for-odd )
   ( define outcome ( roll-die ) )
   ( display outcome ) ( display " " )
   ( cond
      ( ( odd? outcome )
        ( roll-for-odd )
        )
      )
   )



( define ( roll-for-odd-even-odd )
   ( define outcome ( roll-die ) )
   ( display outcome ) ( display " " )
   ( cond
      ( ( even? outcome )
        ( roll-for-odd-even-odd ))
      ( else
        ( cond
           ( ( roll-for-odd )
             ( define outcome ( roll-die ) )
             ( display outcome ) ( display " " )
             ( cond
                ( ( even? outcome )
                  ( roll-for-odd-even-odd )
                  )
                )
             )
           )
        )
      )
   )


( define ( roll-two-dice-for-a-lucky-pair )
   ( define die-outcome ( roll-die ) )
   ( define die-outcome2 ( roll-die ) )
   ( display "( " ) ( display die-outcome ) ( display " " )
   ( display die-outcome2 ) ( display " ) " )
   ( cond
      ( ( eq? die-outcome die-outcome2 ) )
      ( else
        ( cond
           ( ( eq? ( + die-outcome die-outcome2 ) 7 ) )
           ( else
             ( cond
                ( ( eq? ( + die-outcome die-outcome2 ) 11 ) )
                ( else
                  ( cond
                   (( roll-two-dice-for-a-lucky-pair )))

                ) ) )
           )
        )
      )
   )


```

Task 3: Numeric Sequences

Preliminary Demo…

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( square 5 )
25
> ( square 10 )
100
> ( sequence square 15 )
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225
> ( cube 2 )
8
> ( cube 3 )
27
> ( sequence cube 15 )
1 8 27 64 125 216 343 512 729 1000 1331 1728 2197 2744 3375
>
```

Triangular Demo….

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( triangular 1 )
1
> ( triangular 2 )
3
> ( triangular 3 )
6
> ( triangular 4 )
10
> ( triangular 5 )
15
> ( sequence triangular 20 )
1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136 153 171 190 210
>
```

Sigma Demo...

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( sigma 1 )
1
> ( sigma 2 )
3
> ( sigma 3 )
4
> ( sigma 4 )
7
> ( sigma 5 )
6
> ( sequence  sigma 20 )
1 3 4 7 6 12 8 15 13 18 12 28 14 24 24 31 18 39 20 42
>
```

Code...

```
#lang racket

( define ( square n )
   ( * n n )
)
( define ( cube n )
   ( * n n n )
)
( define ( sequence name n )
   ( cond
( ( = n 1 )
  ( display ( name 1 ) ) ( display " " )
)
( else
  ( sequence name ( - n 1 ) )
  ( display ( name n ) ) ( display " " )
)))


( define (triangular n)
  ( / ( * n ( + n 1 )) 2 )
)
( define (sigma n )
   (sigma-figure n n )
   )

( define (sigma-figure s n )
   ( cond
      ( ( = n 1 ) 1 )
      ( else
        ( cond
           ( ( = ( remainder s n) 0 )
            ( + ( sigma-figure s ( - n 1 ) ) n ) )
               (else
                (sigma-figure s (- n 1 ) )
                )))))
```

Task 4: Hirst Dot

Demo...

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( hirst-dots 10 dot)
```

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

```
>
```

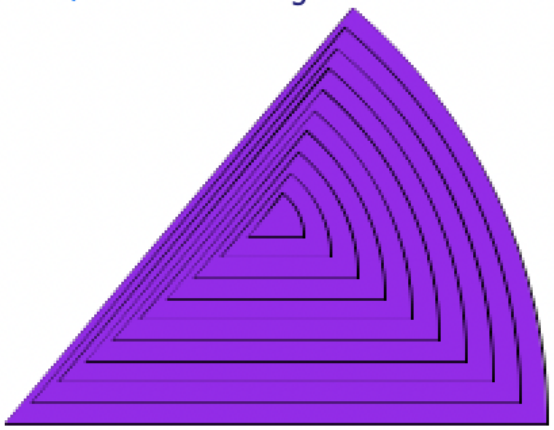
Code...

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

( define ( random-color )
   ( color( rgb-value ) ( rgb-value ) ( rgb-value ) )
   )

( define ( rgb-value ) (random 255 )

   )
( define white-space ( square 20 "solid" "white" ))

(define ( dot ) (circle 10 "solid" ( random-color)))


( define ( row-of-circles n dot )
   ( cond
(( = n 0) empty-image
) (( > n 0)
( beside ( row-of-circles ( - n 1 ) dot ) white-space  ( dot ) ) )
) )


( define ( rectangle-of-circles r c dot)
   ( cond
((= r 0)
 empty-image
)
((> r 0)
( above

( rectangle-of-circles ( - r 1 ) c dot ) white-space ( row-of-circles c dot) ) )
) )

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

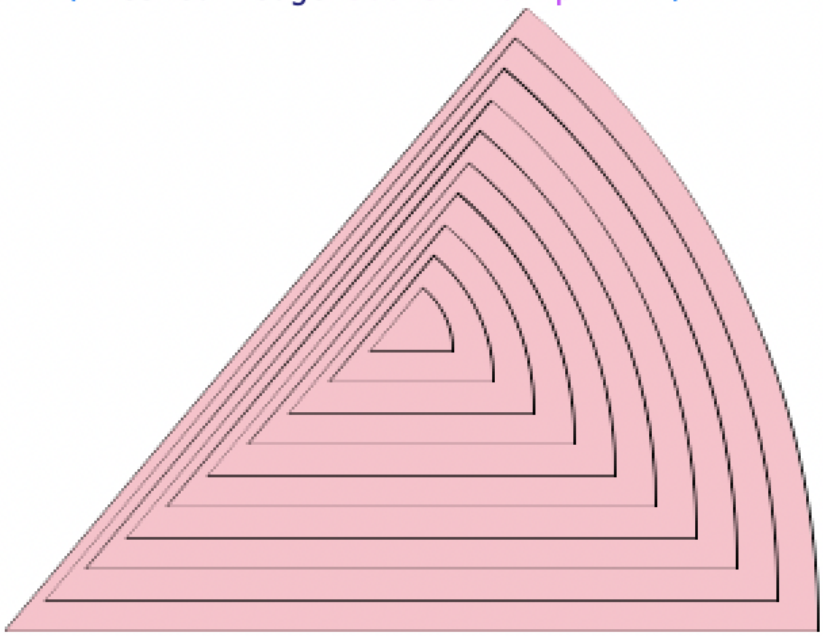
Task 5: Channeling Frank Stella

Demo...

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( nested-wedge 200 50 10 "purple" )
```



```
> ( nested-wedge 300 50 10 "pink" )
```



```
>
```

Code

```
#lang racket


( require 2htdp/image )
( define ( nested-wedge side angle count color ) ( define unit ( / side count ) ) )
( paint-nested-wedge 1 count angle unit color )
)
( define ( paint-nested-wedge from to angle unit color)
( define side-length ( * from unit ) )
   ( cond
( ( = from to )
( framed-wedge side-length angle color )
)
( ( < from to )
( overlay
( framed-wedge side-length angle color )
( paint-nested-wedge ( + from 1 ) to angle unit color )
) )
) )
( define ( framed-wedge side-length angle color )
   ( overlay
( wedge side-length angle "solid" color )
( wedge side-length angle "outline" "black" ) )
)
```

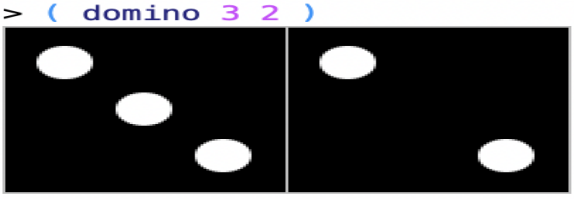
Task 6: Dominos

Final demo...

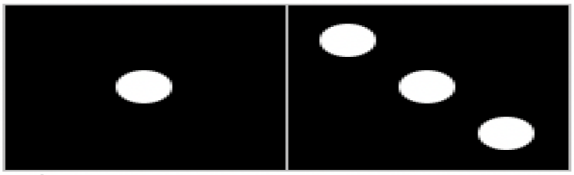
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( domino 0 1 )
```

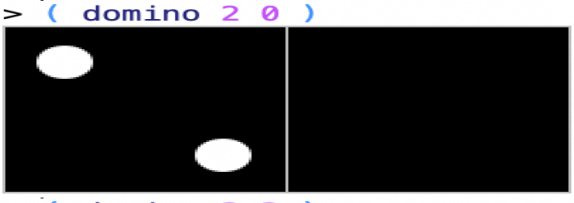
```
> ( domino 3 2 )
```



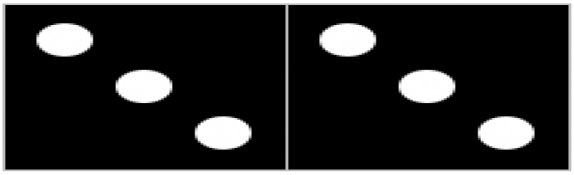
```
> ( domino 1 3 )
```



```
> ( domino 2 0 )
```



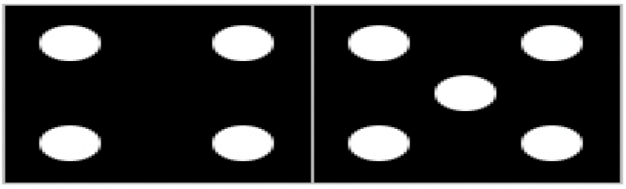
```
> ( domino 3 3 )
```



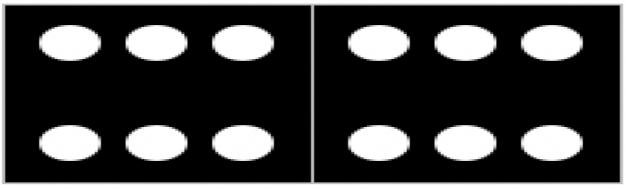
```
> |
```

Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.

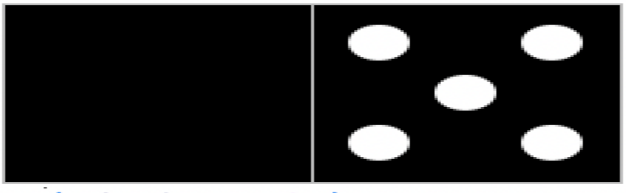
```
> ( domino 4 5 )
```



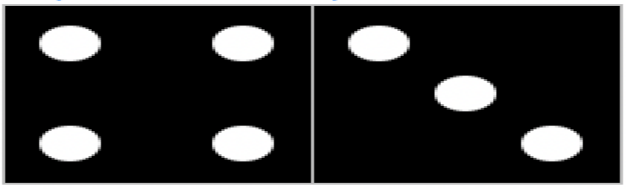
```
> ( domino 6 6 )
```



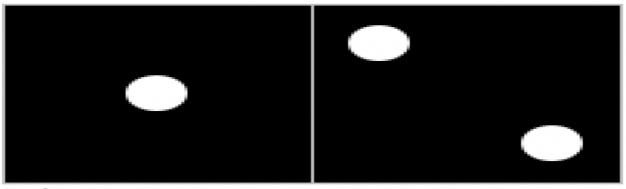
```
> ( domino 0 5 )
```



```
> ( domino 4 3 )
```



```
> ( domino 1 2 )
```



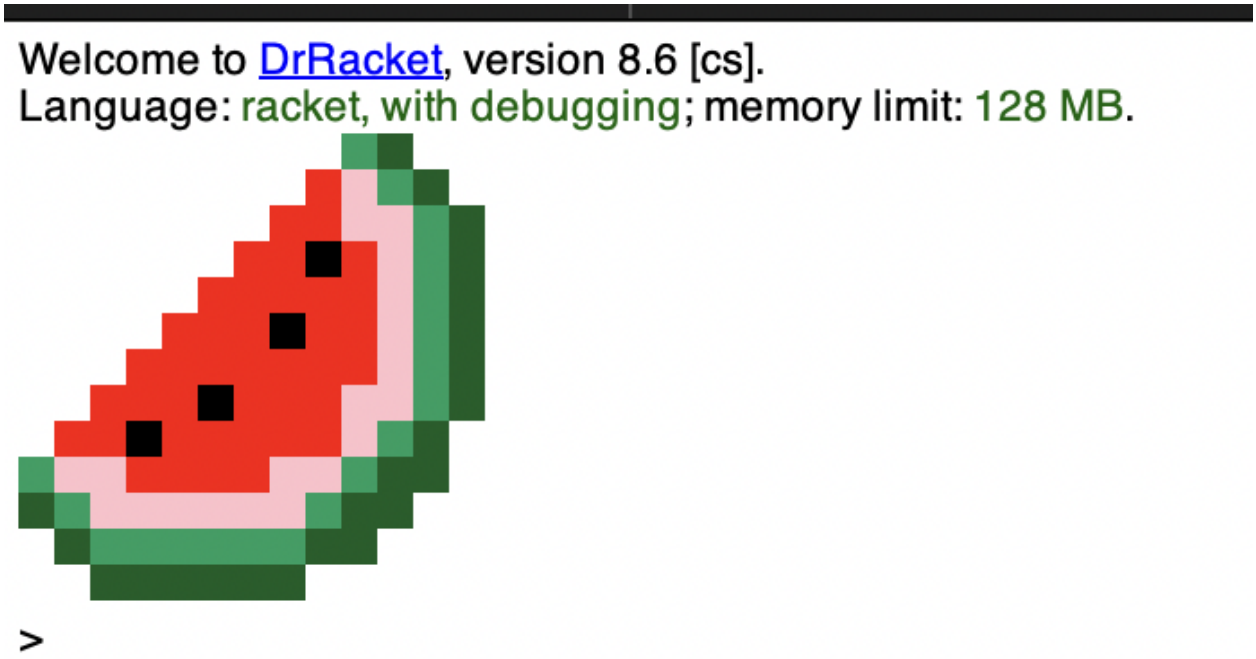
```
>
```

Code...

```
#lang racket


( define d ( * diameter-of-pip 1.4 ) ) ( define nd ( * -1 d ) )

( define blank-tile ( square side-of-tile "solid" "black" ) )
( define ( pip ) ( circle radius-of-pip "solid" "white" ) )

( define basic-tile1( overlay ( pip ) blank-tile ) )
( define basic-tile2 ( overlay/offset (pip) d d
                                    ( overlay/offset(pip) nd nd blank-tile )
) )
( define basic-tile3( overlay ( pip ) basic-tile2 ) )
( define basic-tile4
   ( overlay/offset (pip) d d
                    ( overlay/offset(pip) d nd
                              ( overlay/offset(pip) nd d
                                                ( overlay/offset(pip) nd nd blank-tile)))
                    ))




( define basic-tile5 (overlay ( pip ) basic-tile4 ))
( define basic-tile6
   ( overlay/offset(pip) 0 nd
                  ( overlay/offset(pip) 0 d basic-tile4)))

( define frame ( square side-of-tile "outline" "gray" ) )
( define tile0 ( overlay frame blank-tile ) )
( define tile1 ( overlay frame basic-tile1 ) )
( define tile2 ( overlay frame basic-tile2 ) )
( define tile3 ( overlay frame basic-tile3 ) )
( define tile4 ( overlay frame basic-tile4 ) )
( define tile5 ( overlay frame basic-tile5 ) )
( define tile6 ( overlay frame basic-tile6 ) )



( define ( domino a b )
( beside ( tile a ) ( tile b ) )
)
( define ( tile x ) ( cond

( ( = x 0 ) tile0 )
    ( ( = x 1 ) tile1 )
    ( ( = x 2 ) tile2 )
    ( ( = x 3 ) tile3 )
    ( ( = x 4 ) tile4 )
    ( ( = x 5 ) tile5 )
    ( ( = x 6 ) tile6 )
) )
```

Task 7: Creation

Creation (image)



Code...

```
#lang racket
(require 2htdp/image )
(overlay/xy(overlay/xy(overlay/xy(overlay/xy
(overlay/xy(overlay/xy(overlay/xy(overlay/xy
(overlay/xy(overlay/xy(overlay/xy(overlay/xy
(rectangle 60 10 "solid" (make-color 16 94 38))
-10
-10
(beside(rectangle 10 10 "solid" (make-color 16 94 38)) (rectangle 60 10 "solid" (make-color
       (rectangle 20 10 "solid" (make-color 16 94 38))))
           -10
           -10
(beside(rectangle 10 10 "solid" (make-color 16 94 38))
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 60 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 20 10 "solid" (make-color 16 94 38))
))
           0
           -10
       (beside(rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 20 10 "solid" "pink" )
       (rectangle 40 10 "solid" "red")
       (rectangle 20 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 20 10 "solid" (make-color 16 94 38))))
           10
           -10


(beside(rectangle 20 10 "solid" "red")
       (rectangle 10 10 "solid" "black")
       (rectangle 50 10 "solid" "red")
       (rectangle 10 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
           20
           -10


(beside(rectangle 30 10 "solid" "red")
       (rectangle 10 10 "solid" "black")
       (rectangle 30 10 "solid" "red")
       (rectangle 20 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
30
-10


(beside(rectangle 70 10 "solid" "red")
       (rectangle 10 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
```

```
       (rectangle 10 10 "solid" (make-color 16 94 38))))
           40
           -10

       (beside(rectangle 30 10 "solid" "red")
       (rectangle 10 10 "solid" "black")
       (rectangle 20 10 "solid" "red")
       (rectangle 10 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
50
-10
(beside(rectangle 50 10 "solid" "red")
       (rectangle 10 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
60
-10
(beside(rectangle 20 10 "solid" "red")
       (rectangle 10 10 "solid" "black")
       (rectangle 10 10 "solid" "red")
       (rectangle 10 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
70
-10
(beside(rectangle 20 10 "solid" "red")
       (rectangle 20 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
           80
           -10
       (beside(rectangle 10 10 "solid" "red")
       (rectangle 10 10 "solid" "pink" )
       (rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
90
-10
       (beside(rectangle 10 10 "solid" (make-color 0 158 96))
       (rectangle 10 10 "solid" (make-color 16 94 38))))
```