Temitope Emokpae 9/16/2022

CSC 344

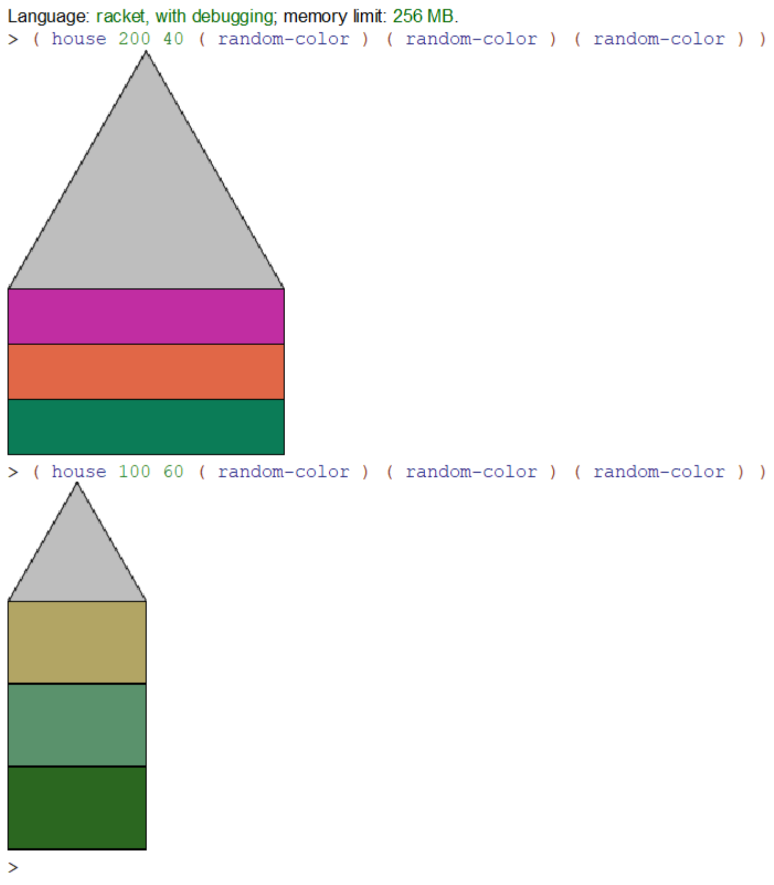
Racket Programming Assignment #2: Racket Functions and Recursion

Learning Abstract

This assignment features programs that generate images in the context of the 2htdp/image library, most of which are recursive in nature.

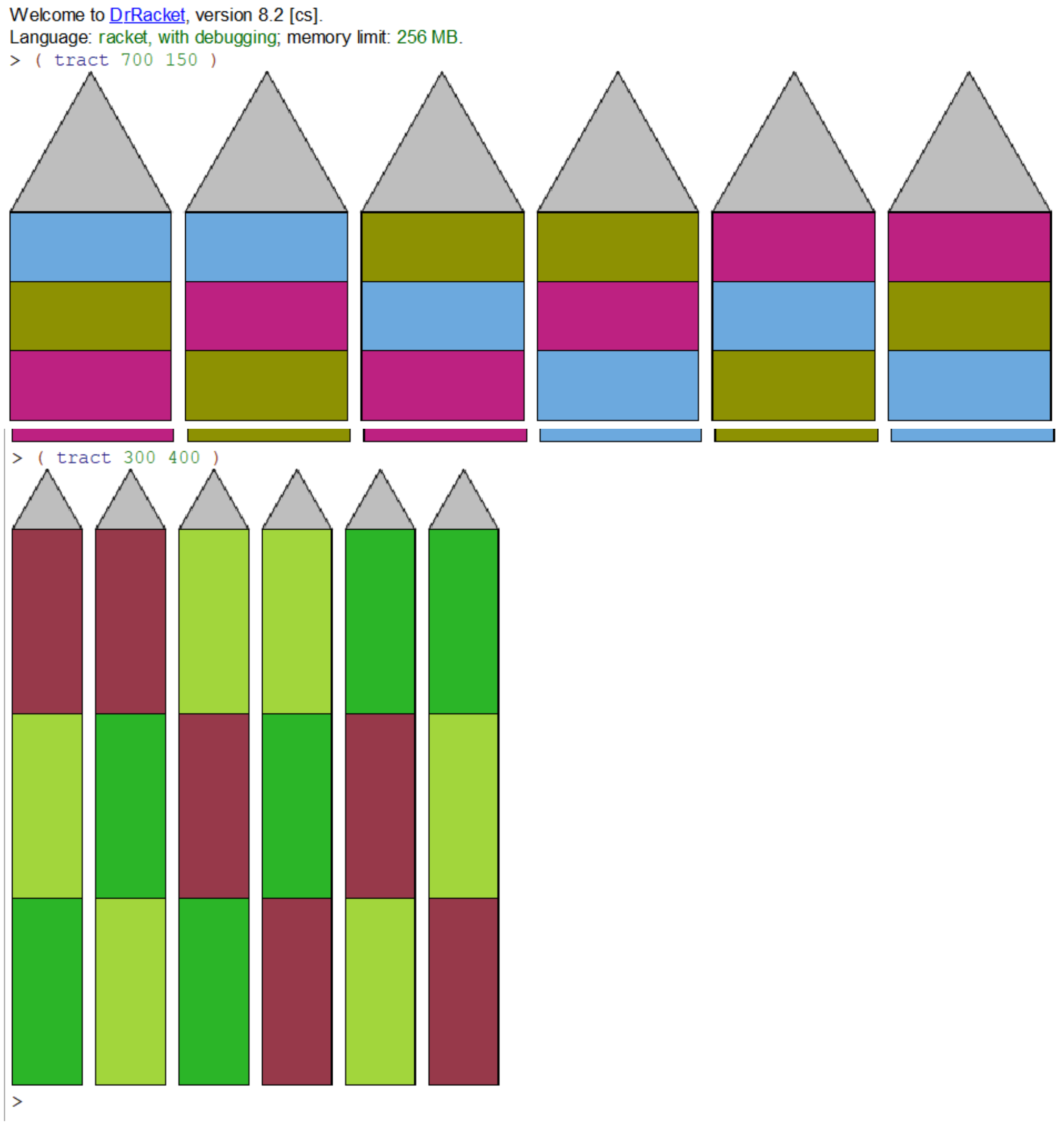
Task 1: Colorful Permutations of Tract Houses

Demo for house



```
Language: racket, with debugging; memory limit: 256 MB.
> ( house 200 40 ( random-color ) ( random-color ) ( random-color ) )
> ( house 100 60 ( random-color ) ( random-color ) ( random-color ) )
>
```

Demo for tract



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( tract 700 150 )
```

```
> ( tract 300 400 )
```

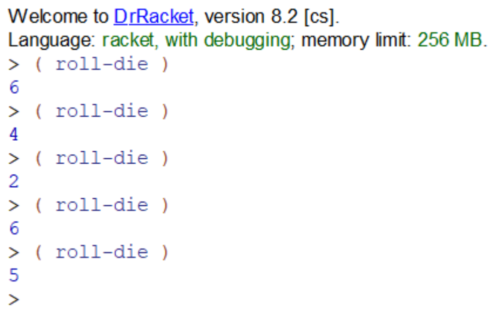
```
>
```

The code ...

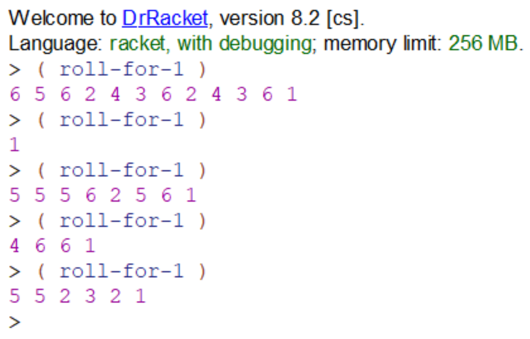
```
#lang racket
( require 2htdp/image )
( define ( random-color ) ( color ( random 256 )( random 256 ) ( random 256 ) ) )
( define ( house width height color1 color2 color3 )
   ( define roof
      ( overlay
        ( triangle width "outline" "black" )
        ( triangle width "solid" "gray" )
       )
      )
   ( define floor1
      ( overlay
        ( rectangle width height "outline" "black" )
        ( rectangle width height "solid" ( random-color ) )
       )
      )
   ( define floor2
      ( overlay
        ( rectangle width height "outline" "black" )
        ( rectangle width height "solid" ( random-color ) )
       )
      )
   ( define floor3
      ( overlay
        ( rectangle width height "outline" "black" )
        ( rectangle width height "solid" ( random-color ) )
       )
      )
   ( above roof floor1 floor2 floor3 )
   )

#lang racket
( require 2htdp/image )
( define ( random-color ) ( color ( random 256 )( random 256 ) ( random 256 ) ) )
( define ( tract width height )
   ( define roof
      ( overlay
        ( triangle ( / width 6 ) "outline" "black" )
        ( triangle ( / width 6 ) "solid" "gray" )
       )
      )
   ( define floor1
      ( overlay
        ( rectangle ( / width 6 ) ( / height 3 ) "outline" "black" )
        ( rectangle ( / width 6 ) ( / height 3 ) "solid" ( random-color ) )
       )
      )
   ( define floor2
      ( overlay
        ( rectangle ( / width 6 ) ( / height 3 ) "outline" "black" )
        ( rectangle ( / width 6 ) ( / height 3 ) "solid" ( random-color ) )
       )
      )
   ( define floor3
      ( overlay
        ( rectangle ( / width 6 ) ( / height 3 ) "outline" "black" )
        ( rectangle ( / width 6 ) ( / height 3 ) "solid" ( random-color ) )
       )
      )
   ( define space
      ( rectangle 10 0 "solid" "white" ) )
   ( define h1
      ( above roof floor1 floor2 floor3 ) )
   ( define h2
      ( above roof floor1 floor3 floor2 ) )
   ( define h3
      ( above roof floor2 floor1 floor3 ) )
   ( define h4
      ( above roof floor2 floor3 floor1 ) )
   ( define h5
      ( above roof floor3 floor1 floor2 ) )
 ( define h6
    ( above roof floor3 floor2 floor1 ) )
  ( beside h1 space h2 space h3 space h4 space h5 space h6 ) )
```

Task 2: Dice

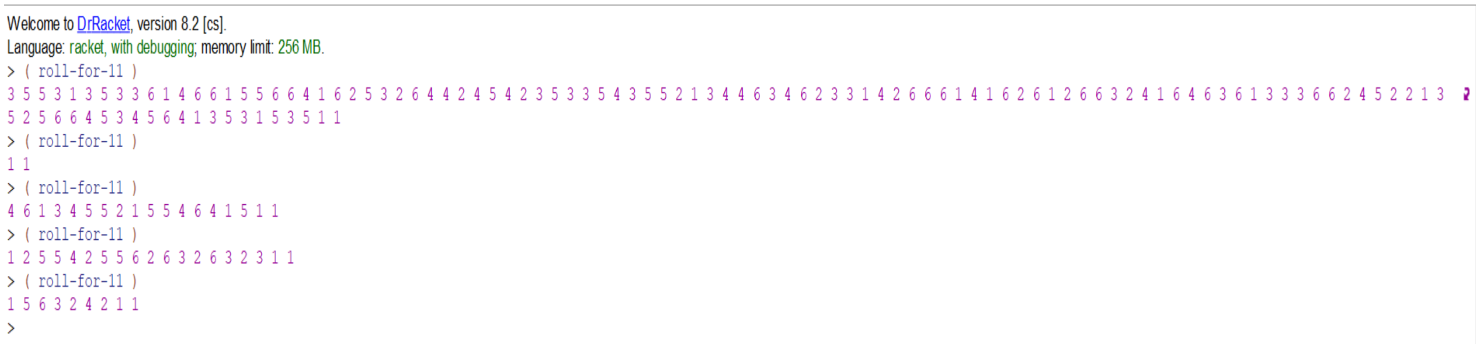
Demo ...



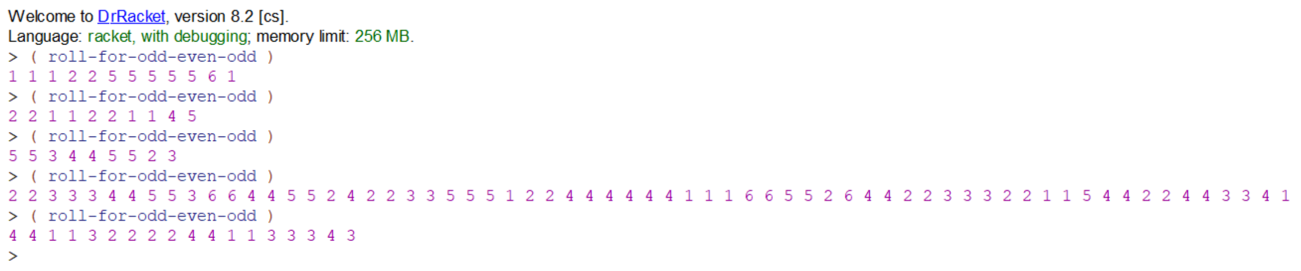
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( roll-die )
6
> ( roll-die )
4
> ( roll-die )
2
> ( roll-die )
6
> ( roll-die )
5
>
```



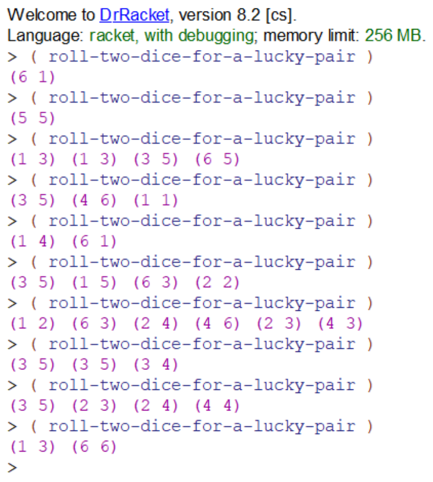
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( roll-for-1 )
6 5 6 2 4 3 6 2 4 3 6 1
> ( roll-for-1 )
1
> ( roll-for-1 )
5 5 5 6 2 5 6 1
> ( roll-for-1 )
4 6 6 1
> ( roll-for-1 )
5 5 2 3 2 1
>
```



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( roll-for-11 )
3 5 5 3 1 3 5 3 3 6 1 4 6 6 1 5 5 6 6 4 1 6 2 5 3 2 6 4 4 2 4 5 4 2 3 5 3 3 5 4 3 5 5 2 1 3 4 4 6 3 4 6 2 3 3 1 4 2 6 6 6 1 4 1 6 2 6 1 2 6 6 3 2 4 1 6 4 6 3 6 1 3 3 3 6 6 2 4 5 2 2 1 3
5 2 5 6 6 4 5 3 4 5 6 4 1 3 5 3 1 5 3 5 1 1
> ( roll-for-11 )
1 1
> ( roll-for-11 )
4 6 1 3 4 5 5 2 1 5 5 4 6 4 1 5 1 1
> ( roll-for-11 )
1 2 5 5 4 2 5 5 6 2 6 3 2 6 3 2 3 1 1
> ( roll-for-11 )
1 5 6 3 2 4 2 1 1
>
```



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( roll-for-odd-even-odd )
1 1 1 2 2 5 5 5 5 5 6 1
> ( roll-for-odd-even-odd )
2 2 1 1 2 2 1 1 4 5
> ( roll-for-odd-even-odd )
5 5 3 4 4 5 5 2 3
> ( roll-for-odd-even-odd )
2 2 3 3 3 4 4 5 5 3 6 6 4 4 5 5 2 4 2 2 3 3 5 5 5 1 2 2 4 4 4 4 4 4 1 1 1 6 6 5 5 2 6 4 4 2 2 3 3 3 2 2 1 1 5 4 4 2 2 4 4 3 3 4 1
> ( roll-for-odd-even-odd )
4 4 1 1 3 2 2 2 2 4 4 1 1 3 3 3 4 3
>
```



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( roll-two-dice-for-a-lucky-pair )
(6 1)
> ( roll-two-dice-for-a-lucky-pair )
(5 5)
> ( roll-two-dice-for-a-lucky-pair )
(1 3) (1 3) (3 5) (6 5)
> ( roll-two-dice-for-a-lucky-pair )
(3 5) (4 6) (1 1)
> ( roll-two-dice-for-a-lucky-pair )
(1 4) (6 1)
> ( roll-two-dice-for-a-lucky-pair )
(3 5) (1 5) (6 3) (2 2)
> ( roll-two-dice-for-a-lucky-pair )
(1 2) (6 3) (2 4) (4 6) (2 3) (4 3)
> ( roll-two-dice-for-a-lucky-pair )
(3 5) (3 5) (3 4)
> ( roll-two-dice-for-a-lucky-pair )
(3 5) (2 3) (2 4) (4 4)
> ( roll-two-dice-for-a-lucky-pair )
(1 3) (6 6)
>
```

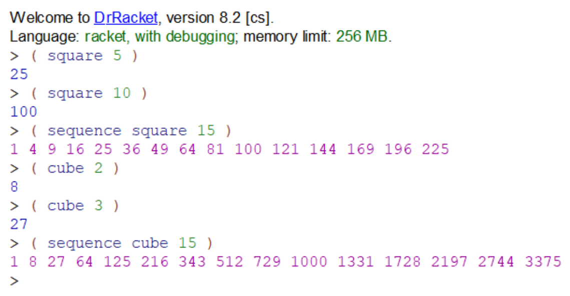
Code ...

```
#lang racket
( define ( roll-die )
   ( define outcome ( random 6 ) )
     ( cond
       ( ( = outcome 0 ) 1 )
       ( ( = outcome 1 ) 2 )
       ( ( = outcome 2 ) 3 )
       ( ( = outcome 3 ) 4 )
       ( ( = outcome 4 ) 5 )
       ( ( = outcome 5 ) 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-even-odd )
   ( define outcome ( roll-die ) )
   ( display outcome ) ( display " " )
   ( cond
      ( ( or ( eq? outcome 1 ) ( eq? outcome 3 ) ( eq? outcome 5 ) )
        ( display outcome ) ( display " " )
        ( set! outcome ( roll-die ) )
        ( cond
           ( ( or ( eq? outcome 2 ) ( eq? outcome 4 ) ( eq? outcome 6 ) )
             ( display outcome ) ( display " " )
             ( set! outcome ( roll-die ) )
        ( cond
           ( ( or ( eq? outcome 1 ) ( eq? outcome 3 ) ( eq? outcome 5 ) )
             ( display outcome )
            )
           ( ( or ( eq? outcome 2 ) ( eq? outcome 4 ) ( eq? outcome 6 ) )
             ( display outcome ) ( display " " )
             ( roll-for-odd-even-odd )
             )
           )
        )
           ( ( or ( eq? outcome 1 ) ( eq? outcome 3 ) ( eq? outcome 5 ) )
             ( display outcome ) ( display " " )
             ( roll-for-odd-even-odd )
             )
           )
        )
      ( ( or ( eq? outcome 2 ) ( eq? outcome 4 ) ( eq? outcome 6 ) )
        ( display outcome ) ( display " " )
        ( roll-for-odd-even-odd )
       )
      )
   )



( define ( roll-two-dice-for-a-lucky-pair )
   ( define dice1 ( roll-die ) )
   ( define dice2 ( roll-die ) )
   ( cond
      ( ( or ( eq? ( + dice1 dice2 ) 7 ) ( eq? ( + dice1 dice2 ) 11 ) ( eq? dice1 dice2 ) )
        ( display "(") ( display dice1 ) ( display " " ) ( display dice2 ) ( display ")" )
        )
      ( else
        ( display "(") ( display dice1 ) ( display " " ) ( display dice2 ) ( display ")" ) ( display " " )
        ( roll-two-dice-for-a-lucky-pair )
        )
      )
   )
```

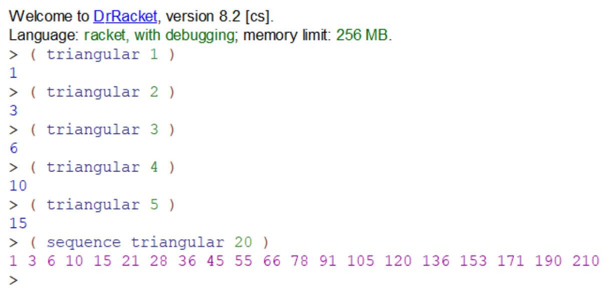
Task 3: Number Sequences

Preliminary demo ...



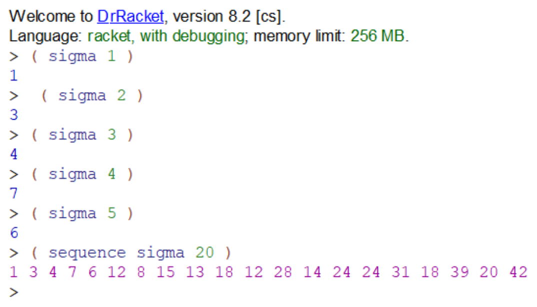
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( square 5 )
25
> ( square 10 )
100
> ( sequence square 15 )
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225
> ( cube 2 )
8
> ( cube 3 )
27
> ( sequence cube 15 )
1 8 27 64 125 216 343 512 729 1000 1331 1728 2197 2744 3375
>
```

Triangular demo ...



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 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 ...



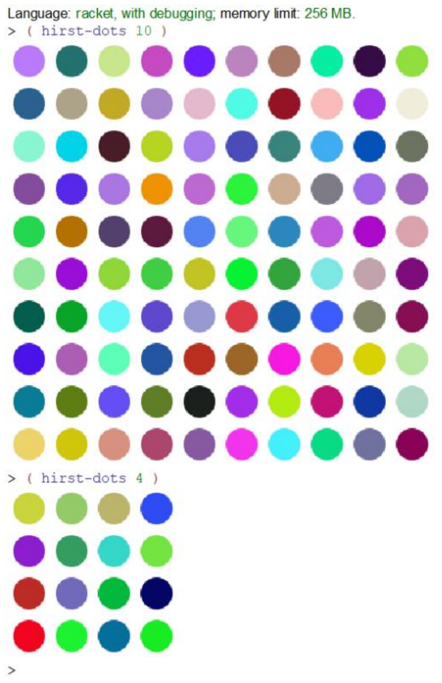
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( sigma 1 )
1
>  ( sigma 2 )
3
> ( sigma 3 )
4
> ( sigma 4 )
7
> ( sigma 5 )
6
> ( sequence sigma 20 )
1 3 4 7 6 12 8 15 13 18 12 28 14 24 24 31 18 39 20 42
>
```

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 )
   ( cond
      ( ( > n 0 )
        ( / ( * n  ( + n 1 ) ) 2 )
        )
      )
    )

( define ( sigma n )
   ( define ( prime n a )
      ( cond
         ( ( = n 1 )
        1
        )
      ( ( eq? ( modulo a n ) 0 )
        ( + n ( prime ( - n 1 ) a ) )
        )
      ( else
        ( prime ( - n 1 ) a )
        )
      )
   )

 ( prime n n )

)
```

Task 4: Hirst Dots

Demo ...



```
Language: racket, with debugging; memory limit: 256 MB.
> ( hirst-dots 10 )
> ( hirst-dots 4 )
>
```

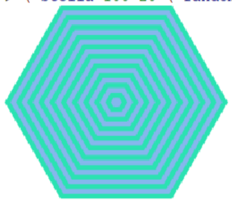
Code ...

```
#lang racket
( require 2htdp/image )
( define ( square-grid ) ( square 40 "solid" "white" ) )
( define ( random-dot ) ( circle 15 "solid" ( random-color ) ) )
( define ( rgb-value ) ( random 256 ) )
( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )
( define ( the-dots ) ( overlay ( random-dot ) ( square-grid ) ) )

( define ( row-of-dots n )
   ( cond
      ( ( = n 0)
        empty-image
        )
      ( ( > n 0 )
        ( beside ( row-of-dots ( - n 1 ) ) ( the-dots ) )
        )
      )
    )

(define ( grid-of-dots height width )
    ( cond
      ( ( = height 0 )
        empty-image
        )
      ( ( > height 0 )
        ( above ( grid-of-dots ( - height 1 ) width ) ( row-of-dots width ) )
        )
      )
    )

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

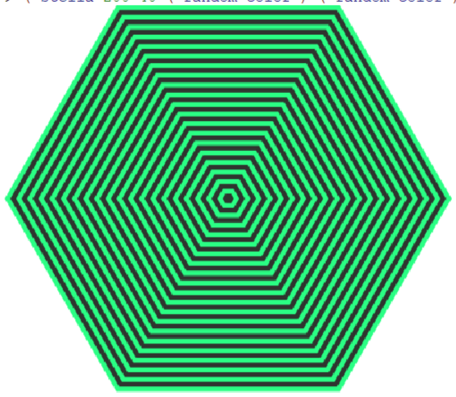
Task 5: Chanelling Frank Stella

Demo ...

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( stella 100 20 ( random-color ) ( random-color ) )
```



```
> ( stella 200 40 ( random-color ) ( random-color ) )
```



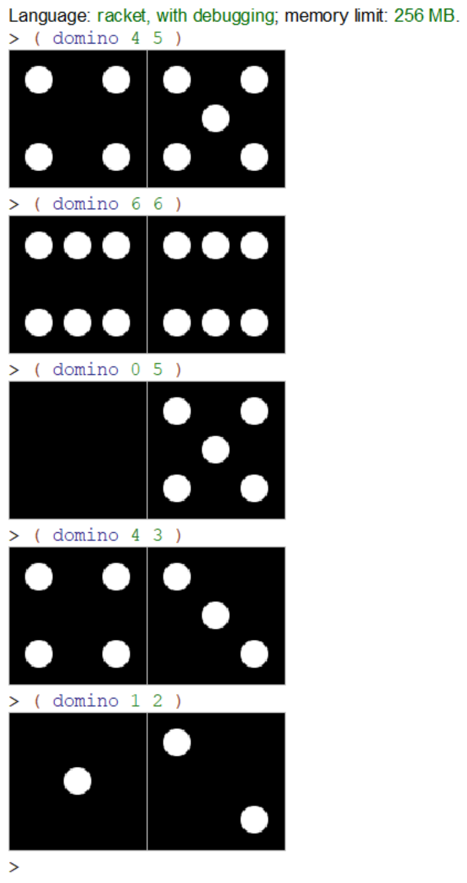
```
>
```

Code ...

```
#lang racket
( require 2htdp/image )
( define ( random-color ) ( color ( random 256 )( random 256 ) ( random 256 ) ) )

( define ( stella side count color1 color2 )
    ( define delta ( / side count ) )
    ( paint-nested-hexagon 1 count delta color1 color2 )
    )

( define ( paint-nested-hexagon from to delta color1 color2 )
    ( define side-length ( * from delta ) )
    ( cond
      ( ( = from to )
        ( if ( even? from )
            ( regular-polygon side-length 6 "solid" color1 )
            ( regular-polygon side-length 6 "solid" color2 )
            )
        )
      ( ( < from to )
        ( if ( even? from )
            ( overlay
              ( regular-polygon side-length 6 "solid" color1 )
              ( paint-nested-hexagon ( + from 1 ) to delta color1 color2 )
              )
            ( overlay
              ( regular-polygon side-length 6 "solid" color2 )
              ( paint-nested-hexagon ( + from 1 ) to delta color1 color2 )
              )
            )
        )
      )
    )
```

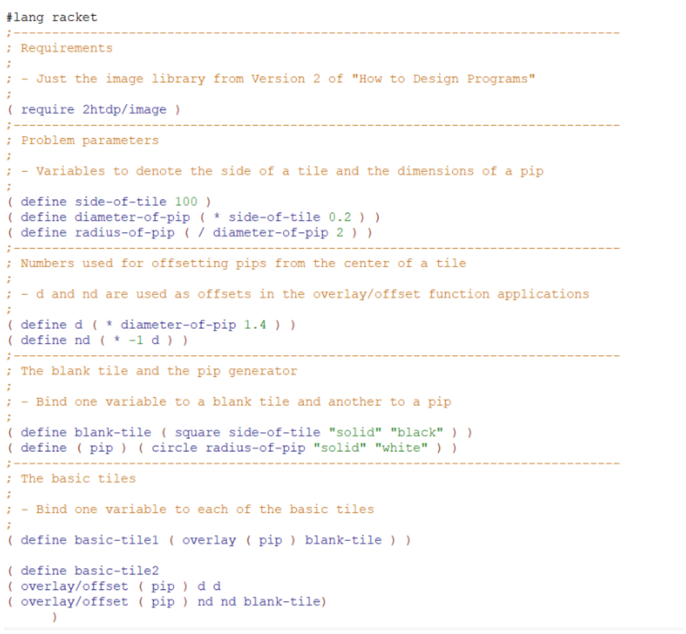
Task 6: Dominos

Final demo ...

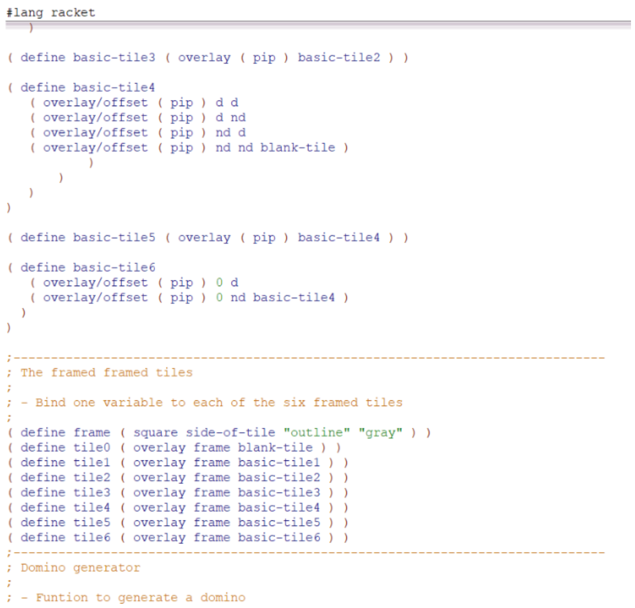


```
Language: racket, with debugging; memory limit: 256 MB.
> ( domino 4 5 )
> ( domino 6 6 )
> ( domino 0 5 )
> ( domino 4 3 )
> ( domino 1 2 )
>
```

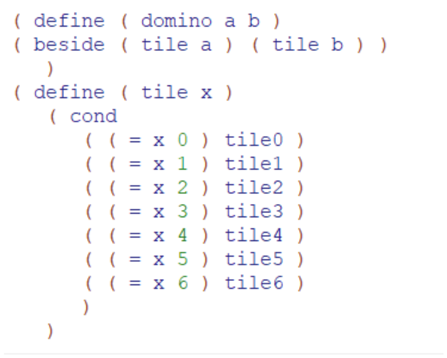
Collected code ...



```racket
#lang racket
;------------------------------------------------------------------------------
; Requirements
;
; - Just the image library from Version 2 of "How to Design Programs"
;
( require 2htdp/image )
;------------------------------------------------------------------------------
; Problem parameters
;
; - Variables to denote the side of a tile and the dimensions of a pip
;
( define side-of-tile 100 )
( define diameter-of-pip ( * side-of-tile 0.2 ) )
( define radius-of-pip ( / diameter-of-pip 2 ) )
;------------------------------------------------------------------------------
; Numbers used for offsetting pips from the center of a tile
;
; - d and nd are used as offsets in the overlay/offset function applications
;
( define d ( * diameter-of-pip 1.4 ) )
( define nd ( * -1 d ) )
;------------------------------------------------------------------------------
; The blank tile and the pip generator
;
; - Bind one variable to a blank tile and another to a pip
;
( define blank-tile ( square side-of-tile "solid" "black" ) )
( define ( pip ) ( circle radius-of-pip "solid" "white" ) )
;------------------------------------------------------------------------------
; The basic tiles
;
; - Bind one variable to each of the basic tiles
;
( define basic-tile1 ( overlay ( pip ) blank-tile ) )

( define basic-tile2
( overlay/offset ( pip ) d d
( overlay/offset ( pip ) nd nd blank-tile)
      )
```



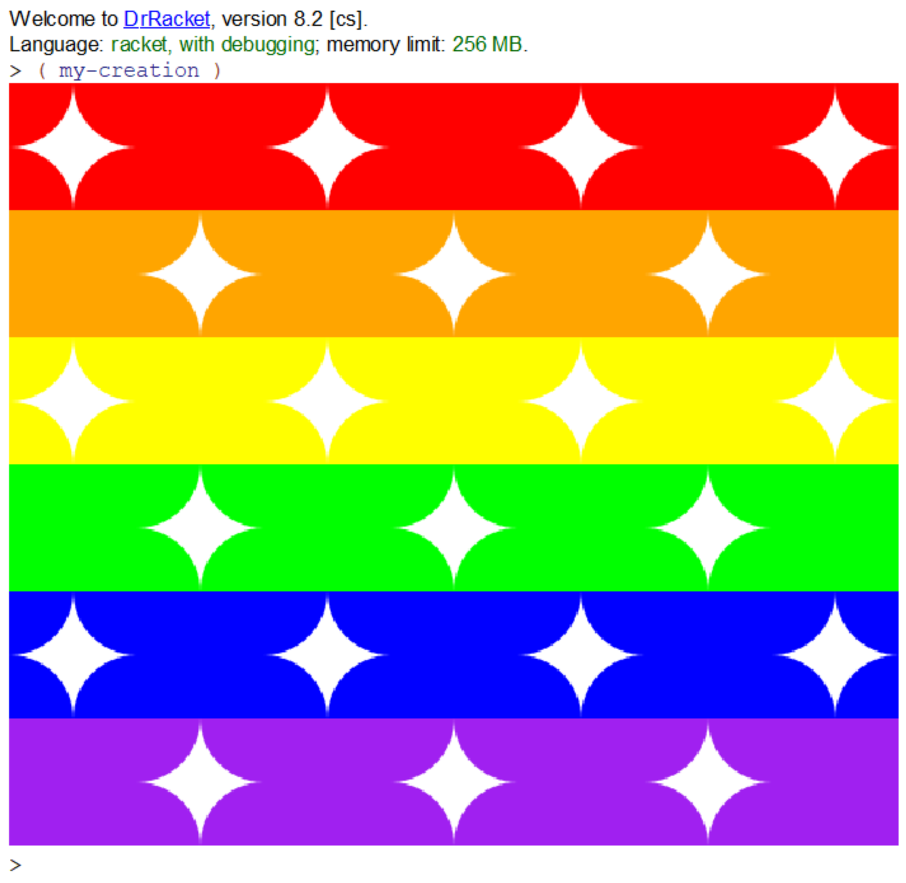
```racket
#lang racket
   )

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

;------------------------------------------------------------------------------
; The framed framed tiles
;
; - Bind one variable to each of the six framed tiles
;
( 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 ) )
;------------------------------------------------------------------------------
; Domino generator
;
; - Funtion to generate a domino
```



```racket
( 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) ...



```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( my-creation )
>
```

Code ...

```racket
#lang racket
( require 2htdp/image )
( define ( red-tile ) ( square 80 "solid" "red" ) )
( define red-spot ( above
                        ( beside ( crop/align "right" "bottom" 40 40 ( circle 40 "solid" "red" ) )
            ( crop/align "left" "bottom" 40 40 (circle 40 "solid" "red" ) ) )
   ( beside ( crop/align "right" "top" 40 40 (circle 40 "solid" "red" ) )
            ( crop/align "left" "top" 40 40 (circle 40 "solid" "red" ) )
            )
   ) )
( define ( red-row ) ( beside red-spot ( red-tile ) red-spot ( red-tile ) red-spot ( red-tile ) red-spot ) )
( define ( orange-tile ) ( square 80 "solid" "orange" ) )
( define orange-spot ( above
                        ( beside (crop/align "right" "bottom" 40 40 ( circle 40 "solid" "orange" ) )
            ( crop/align "left" "bottom" 40 40 ( circle 40 "solid" "orange" ) ) )
   (beside ( crop/align "right" "top" 40 40 ( circle 40 "solid" "orange" ) )
            ( crop/align "left" "top" 40 40 ( circle 40 "solid" "orange" ) )
            )
   ))
( define ( orange-row ) ( beside ( orange-tile ) orange-spot ( orange-tile ) orange-spot ( orange-tile ) orange-spot ( orange-tile ) ) )
( define ( yellow-tile ) ( square 80 "solid" "yellow"))
( define yellow-spot ( above
                        ( beside ( crop/align "right" "bottom" 40 40 ( circle 40 "solid" "yellow" ) )
            ( crop/align "left" "bottom" 40 40 (circle 40 "solid" "yellow" ) ) )
   (beside ( crop/align "right" "top" 40 40 (circle 40 "solid" "yellow" ) )
            ( crop/align "left" "top" 40 40 ( circle 40 "solid" "yellow" ) )
            )
   ))
( define ( yellow-row ) ( beside yellow-spot ( yellow-tile ) yellow-spot ( yellow-tile ) yellow-spot ( yellow-tile ) yellow-spot ) )
( define ( green-tile ) ( square 80 "solid" "green" ) )
( define green-spot ( above
                        ( beside ( crop/align "right" "bottom" 40 40 ( circle 40 "solid" "green" ) )
            ( crop/align "left" "bottom" 40 40 (circle 40 "solid" "green" ) ) )
   (beside ( crop/align "right" "top" 40 40 (circle 40 "solid" "green") )
            ( crop/align "left" "top" 40 40 (circle 40 "solid" "green") )
            )
   ) )
( define ( green-row ) ( beside ( green-tile ) green-spot ( green-tile ) green-spot ( green-tile ) green-spot ( green-tile ) ) )
( define ( blue-tile ) ( square 80 "solid" "blue"))
( define blue-spot ( above
                       ( beside ( crop/align "right" "bottom" 40 40 ( circle 40 "solid" "blue" ) )
            ( crop/align "left" "bottom" 40 40 (circle 40 "solid" "blue" ) ) )
   ( beside ( crop/align "right" "top" 40 40 (circle 40 "solid" "blue" ) )
            ( crop/align "left" "top" 40 40 ( circle 40 "solid" "blue" ) )
            )
   ))
( define ( blue-row ) ( beside blue-spot ( blue-tile ) blue-spot ( blue-tile ) blue-spot ( blue-tile ) blue-spot ) )
( define ( purple-tile ) ( square 80 "solid" "purple" ) )
( define purple-spot ( above
                       ( beside ( crop/align "right" "bottom" 40 40 ( circle 40 "solid" "purple" ) )
            ( crop/align "left" "bottom" 40 40 (circle 40 "solid" "purple" ) ) )
   (beside ( crop/align "right" "top" 40 40 (circle 40 "solid" "purple") )
            ( crop/align "left" "top" 40 40 (circle 40 "solid" "purple") )
            )
   ) )
( define ( purple-row ) ( beside ( purple-tile ) purple-spot ( purple-tile ) purple-spot ( purple-tile ) purple-spot ( purple-tile ) ) )
( define ( my-creation ) ( above ( red-row ) ( orange-row ) ( yellow-row ) ( green-row ) ( blue-row ) ( purple-row ) ) )
```