Racket Programming Assignment #2: Functions and Recursion

Learning Abstract:

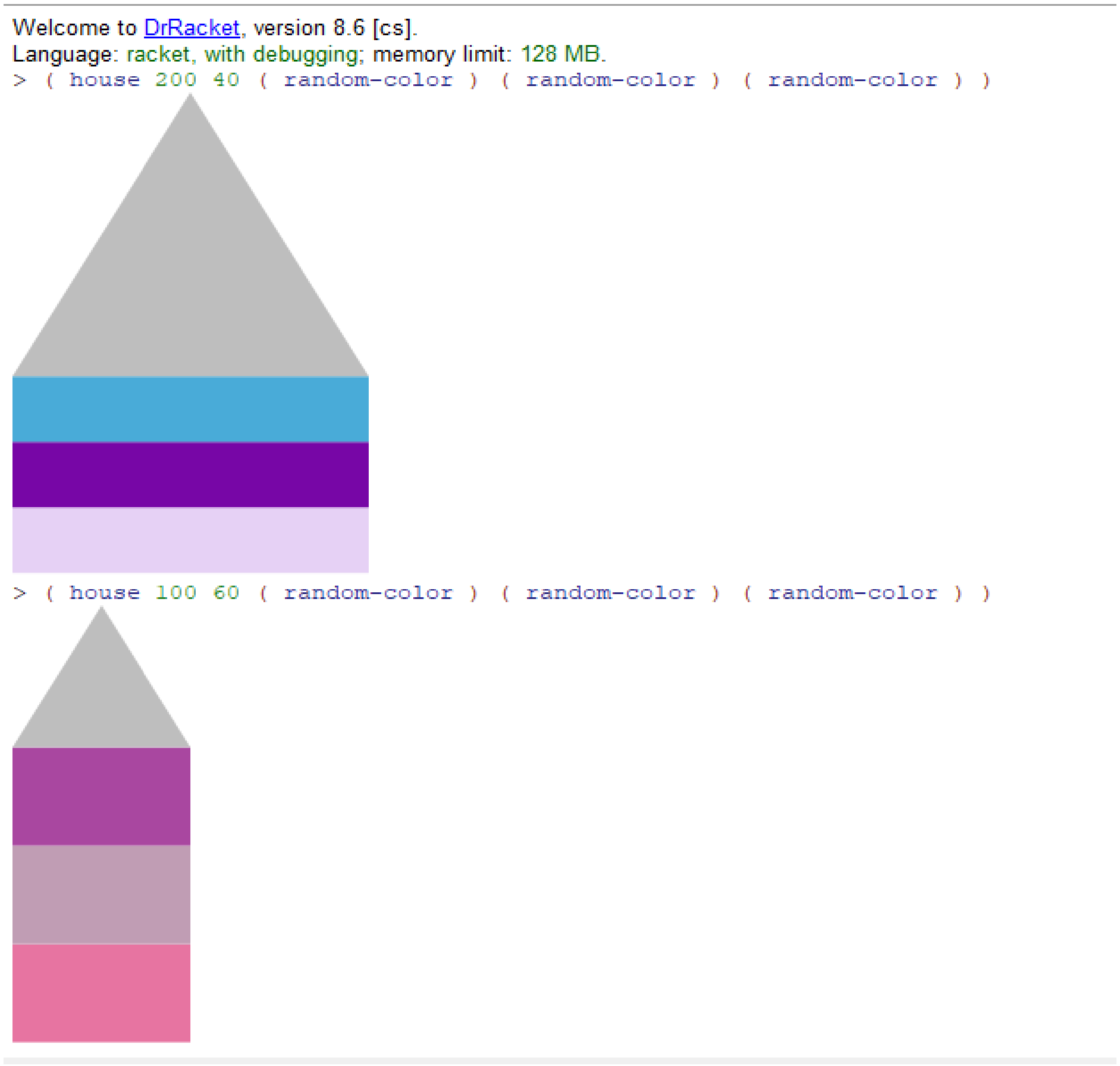
This assignment will cover functions and recursion using Racket.

1. Build a housing tract using a recursive function.
2. Complete several tasks related to rolling dice.
3. Perform mathematical operations and sequences of operations.
4. Create a Hirst Dot image using recursion.
5. Create a Frank Stella image.
6. Complete a given set of domino code.
7. Build a creation using the image library.

Housing Tract

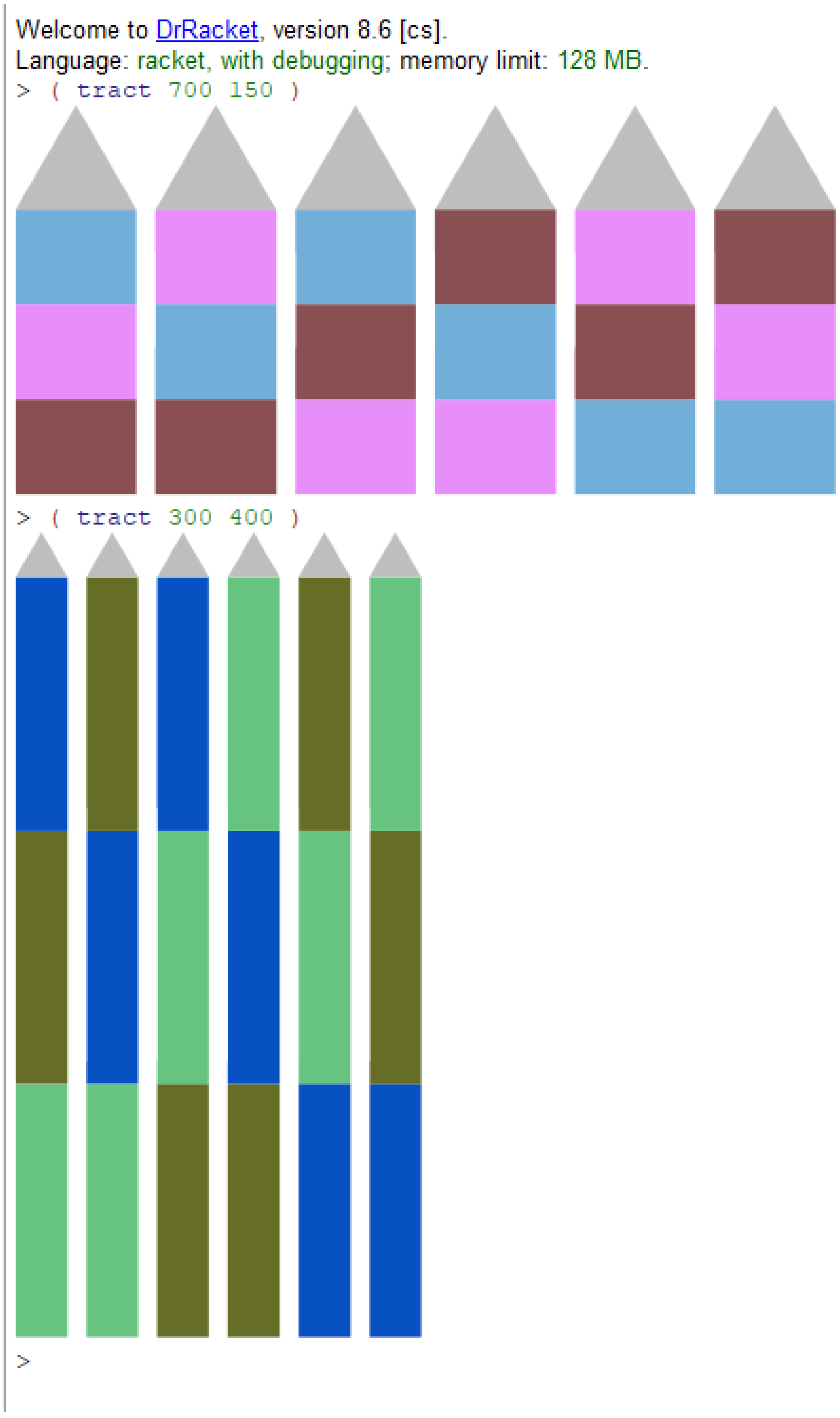
GitHub

A single house:



```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( house 200 40 ( random-color ) ( random-color ) ( random-color ) )
> ( house 100 60 ( random-color ) ( random-color ) ( random-color ) )
```

Tract:

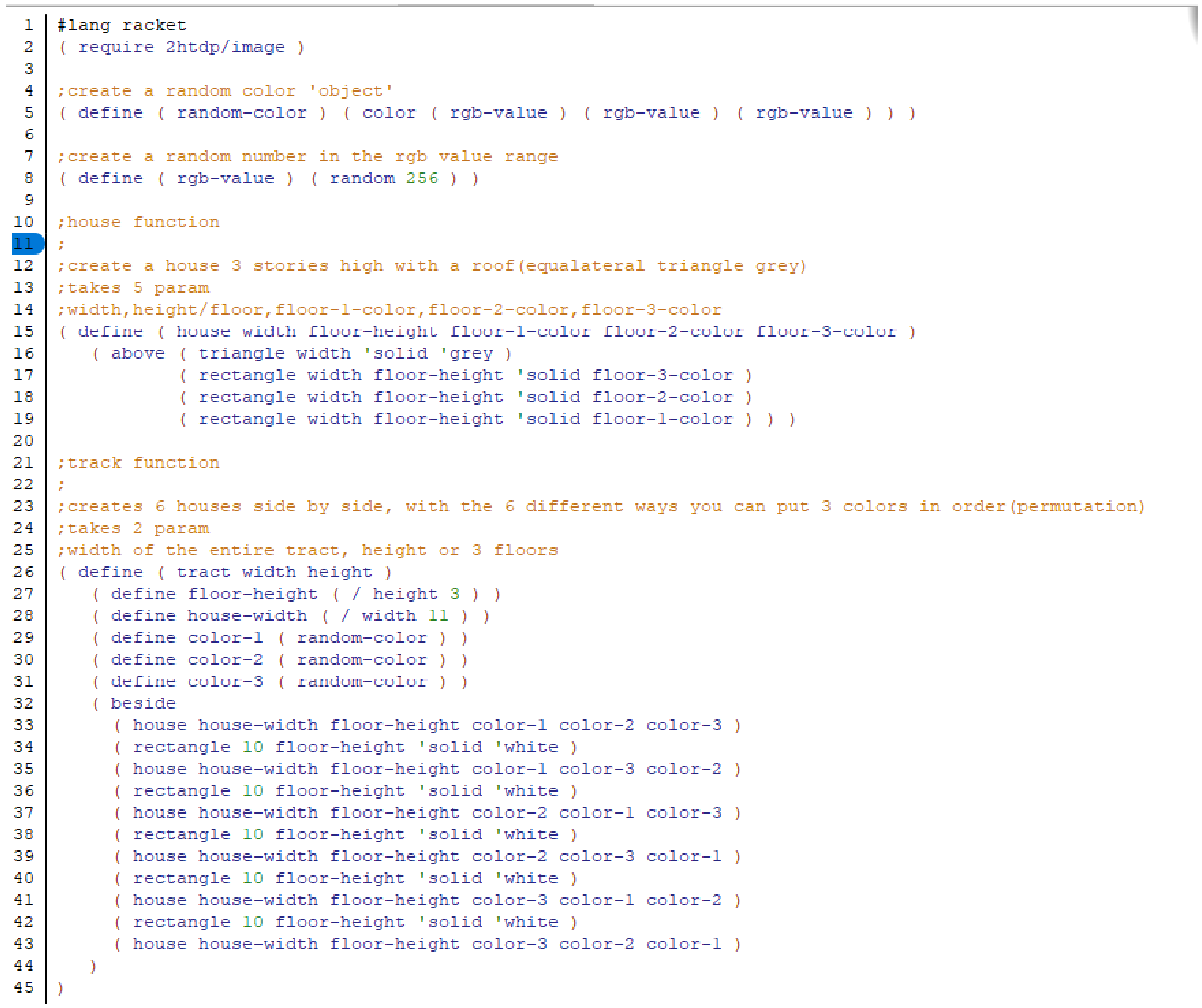


```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( tract 700 150 )
```

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

```
>
```

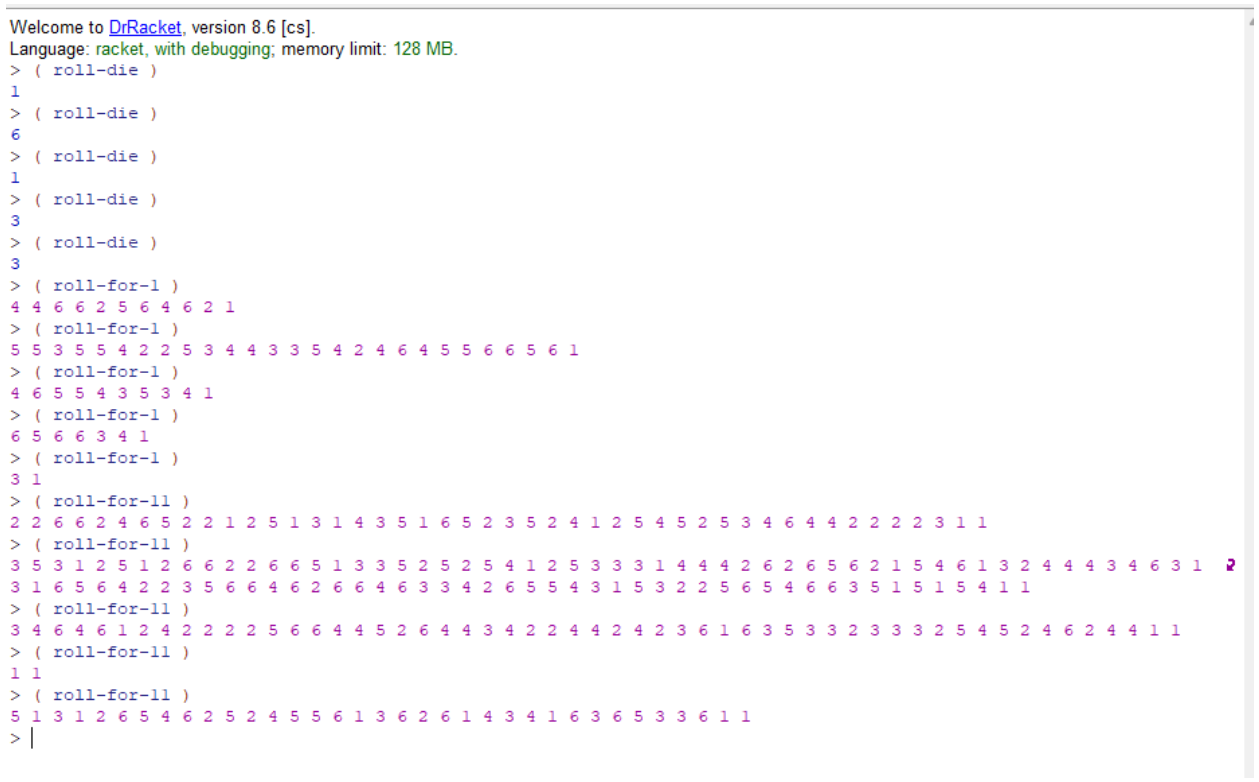
Code: (there is also a link to my Git Repo above)



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

;create a random color 'object'
( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )

;create a random number in the rgb value range
( define ( rgb-value ) ( random 256 ) )

;house function
;
;create a house 3 stories high with a roof(equalateral triangle grey)
;takes 5 param
;width,height/floor,floor-1-color,floor-2-color,floor-3-color
( define ( house width floor-height floor-1-color floor-2-color floor-3-color )
   ( above ( triangle width 'solid 'grey )
           ( rectangle width floor-height 'solid floor-3-color )
           ( rectangle width floor-height 'solid floor-2-color )
           ( rectangle width floor-height 'solid floor-1-color ) ) )

;track function
;
;creates 6 houses side by side, with the 6 different ways you can put 3 colors in order(permutation)
;takes 2 param
;width of the entire tract, height or 3 floors
( define ( tract width height )
   ( define floor-height ( / height 3 ) )
   ( define house-width ( / width 11 ) )
   ( define color-1 ( random-color ) )
   ( define color-2 ( random-color ) )
   ( define color-3 ( random-color ) )
   ( beside
     ( house house-width floor-height color-1 color-2 color-3 )
     ( rectangle 10 floor-height 'solid 'white )
     ( house house-width floor-height color-1 color-3 color-2 )
     ( rectangle 10 floor-height 'solid 'white )
     ( house house-width floor-height color-2 color-1 color-3 )
     ( rectangle 10 floor-height 'solid 'white )
     ( house house-width floor-height color-2 color-3 color-1 )
     ( rectangle 10 floor-height 'solid 'white )
     ( house house-width floor-height color-3 color-1 color-2 )
     ( rectangle 10 floor-height 'solid 'white )
     ( house house-width floor-height color-3 color-2 color-1 )
   )
)
```

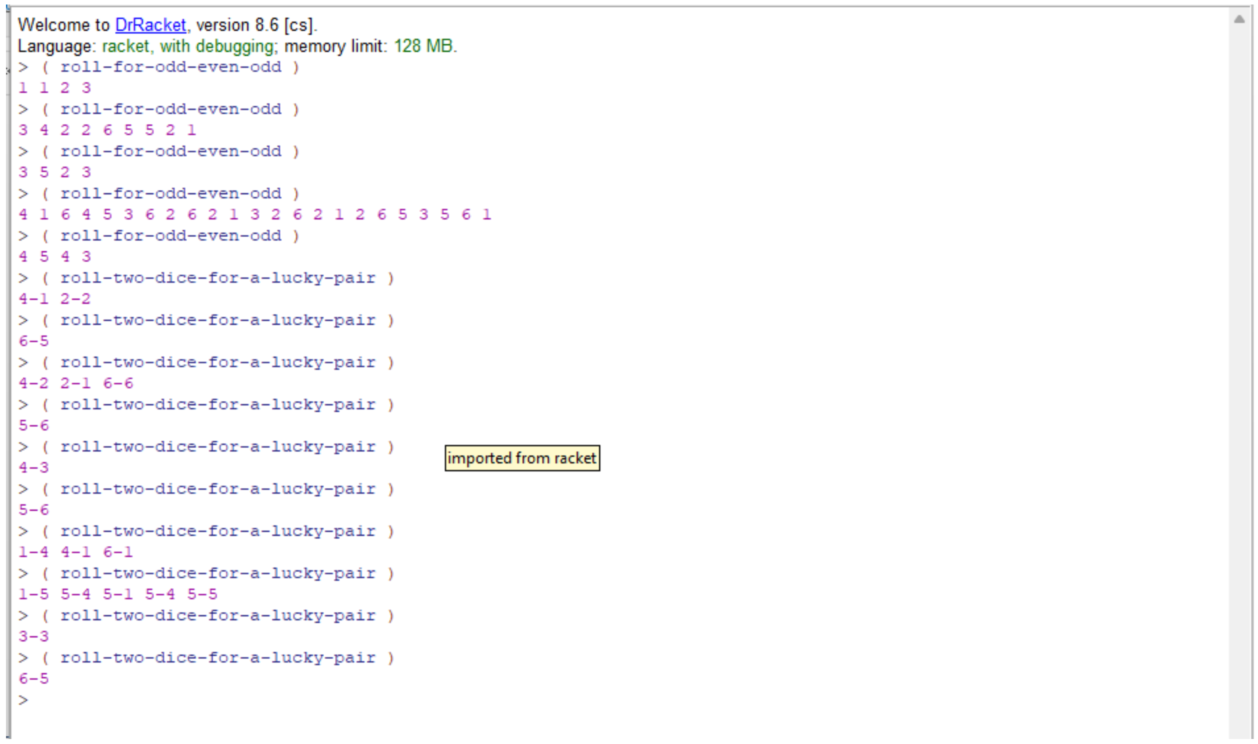
Dice Rolling   GitHUB

Part 1:



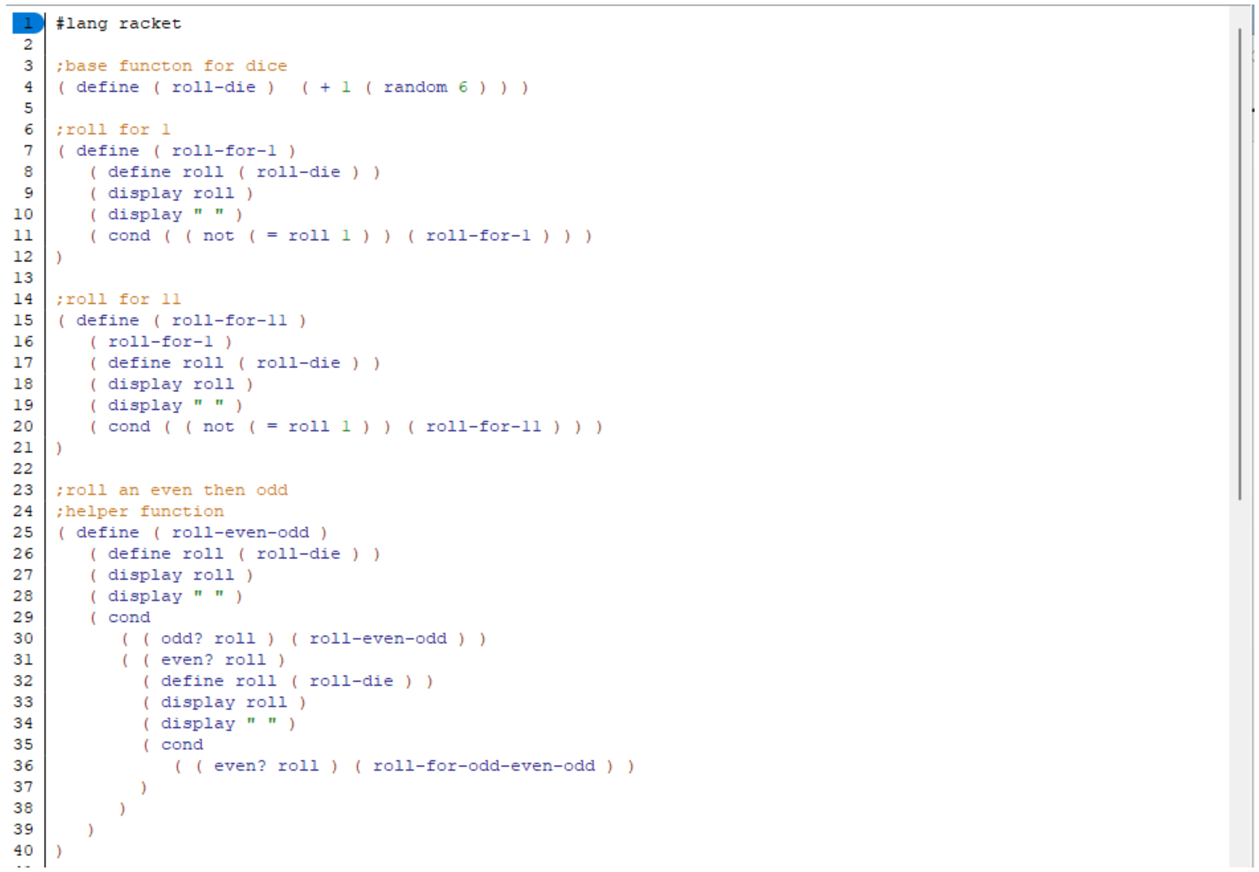
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( roll-die )
1
> ( roll-die )
6
> ( roll-die )
1
> ( roll-die )
3
> ( roll-die )
3
> ( roll-for-1 )
4 4 6 6 2 5 6 4 6 2 1
> ( roll-for-1 )
5 5 3 5 5 4 2 2 5 3 4 4 3 3 5 4 2 4 6 4 5 5 6 6 5 6 1
> ( roll-for-1 )
4 6 5 5 4 3 5 3 4 1
> ( roll-for-1 )
6 5 6 6 3 4 1
> ( roll-for-1 )
3 1
> ( roll-for-11 )
2 2 6 6 2 4 6 5 2 2 1 2 5 1 3 1 4 3 5 1 6 5 2 3 5 2 4 1 2 5 4 5 2 5 3 4 6 4 4 2 2 2 2 3 1 1
> ( roll-for-11 )
3 5 3 1 2 5 1 2 6 6 2 2 6 6 5 1 3 3 5 2 5 2 5 4 1 2 5 3 3 3 1 4 4 4 2 6 2 6 5 6 2 1 5 4 6 1 3 2 4 4 4 3 4 6 3 1
3 1 6 5 6 4 2 2 3 5 6 6 4 6 2 6 6 4 6 3 3 4 2 6 5 5 4 3 1 5 3 2 2 5 6 5 4 6 6 3 5 1 5 1 5 4 1 1
> ( roll-for-11 )
3 4 6 4 6 1 2 4 2 2 2 2 5 6 6 4 4 5 2 6 4 4 3 4 2 2 4 4 2 4 2 3 6 1 6 3 5 3 3 2 3 3 3 2 5 4 5 2 4 6 2 4 4 1 1
> ( roll-for-11 )
1 1
> ( roll-for-11 )
5 1 3 1 2 6 5 4 6 2 5 2 4 5 5 6 1 3 6 2 6 1 4 3 4 1 6 3 6 5 3 3 6 1 1
> |
```

Part 2:



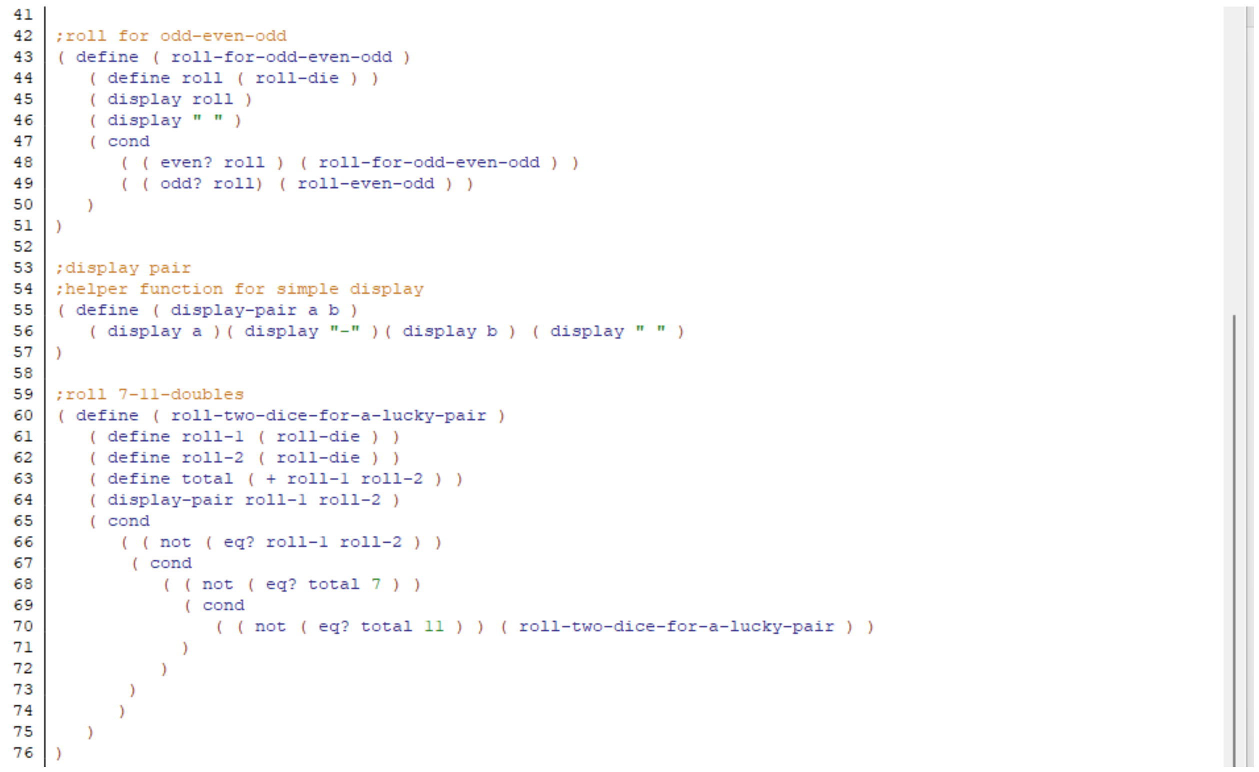
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( roll-for-odd-even-odd )
1 1 2 3
> ( roll-for-odd-even-odd )
3 4 2 2 6 5 5 2 1
> ( roll-for-odd-even-odd )
3 5 2 3
> ( roll-for-odd-even-odd )
4 1 6 4 5 3 6 2 6 2 1 3 2 6 2 1 2 6 5 3 5 6 1
> ( roll-for-odd-even-odd )
4 5 4 3
> ( roll-two-dice-for-a-lucky-pair )
4-1 2-2
> ( roll-two-dice-for-a-lucky-pair )
6-5
> ( roll-two-dice-for-a-lucky-pair )
4-2 2-1 6-6
> ( roll-two-dice-for-a-lucky-pair )
5-6
> ( roll-two-dice-for-a-lucky-pair )
4-3
> ( roll-two-dice-for-a-lucky-pair )
5-6
> ( roll-two-dice-for-a-lucky-pair )
1-4 4-1 6-1
> ( roll-two-dice-for-a-lucky-pair )
1-5 5-4 5-1 5-4 5-5
> ( roll-two-dice-for-a-lucky-pair )
3-3
> ( roll-two-dice-for-a-lucky-pair )
6-5
>
```

Code Part 1:



```
#lang racket

;base functon for dice
( define ( roll-die )  ( + 1 ( random 6 ) ) )

;roll for 1
( define ( roll-for-1 )
   ( define roll ( roll-die ) )
   ( display roll )
   ( display " " )
   ( cond ( ( not ( = roll 1 ) ) ( roll-for-1 ) ) )
)

;roll for 11
( define ( roll-for-11 )
   ( roll-for-1 )
   ( define roll ( roll-die ) )
   ( display roll )
   ( display " " )
   ( cond ( ( not ( = roll 1 ) ) ( roll-for-11 ) ) )
)

;roll an even then odd
;helper function
( define ( roll-even-odd )
   ( define roll ( roll-die ) )
   ( display roll )
   ( display " " )
   ( cond
      ( ( odd? roll ) ( roll-even-odd ) )
      ( ( even? roll )
        ( define roll ( roll-die ) )
        ( display roll )
        ( display " " )
        ( cond
           ( ( even? roll ) ( roll-for-odd-even-odd ) )
        )
      )
   )
)
```

Code Part 2:

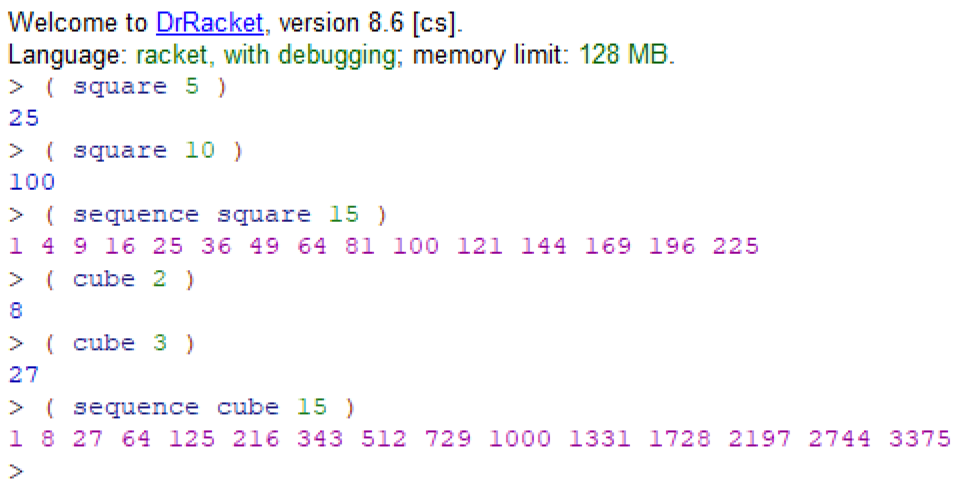


```

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

;display pair
;helper function for simple display
( define ( display-pair a b )
   ( display a )( display "-" )( display b ) ( display " " )
)

;roll 7-11-doubles
( define ( roll-two-dice-for-a-lucky-pair )
   ( define roll-1 ( roll-die ) )
   ( define roll-2 ( roll-die ) )
   ( define total ( + roll-1 roll-2 ) )
   ( display-pair roll-1 roll-2 )
   ( cond
      ( ( not ( eq? roll-1 roll-2 ) )
       ( cond
          ( ( not ( eq? total 7 ) )
            ( cond
               ( ( not ( eq? total 11 ) ) ( roll-two-dice-for-a-lucky-pair ) )
            )
          )
       )
      )
   )
)
```

Number Sequences

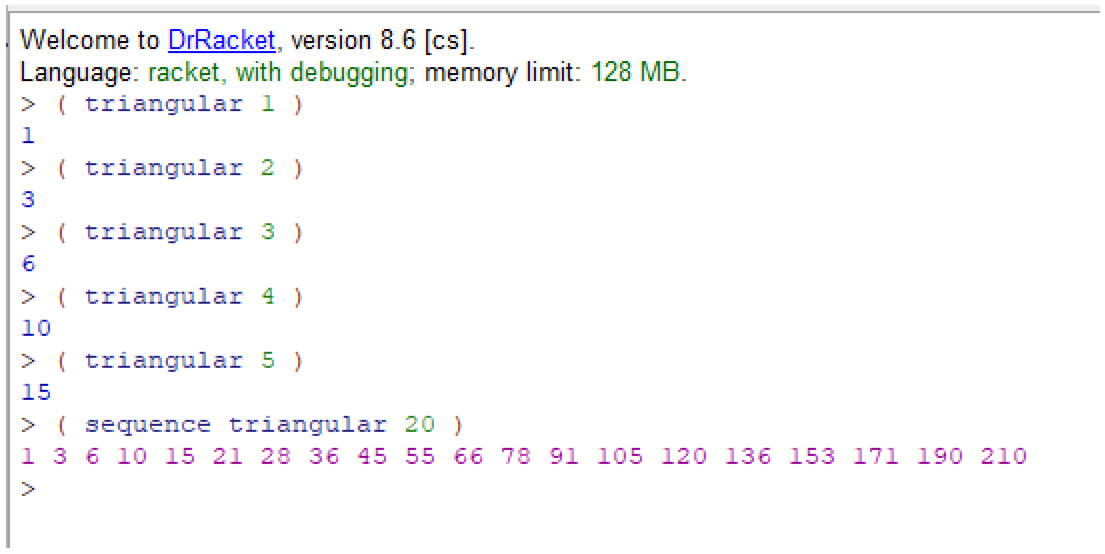
GitHUB

Preliminary Demo:



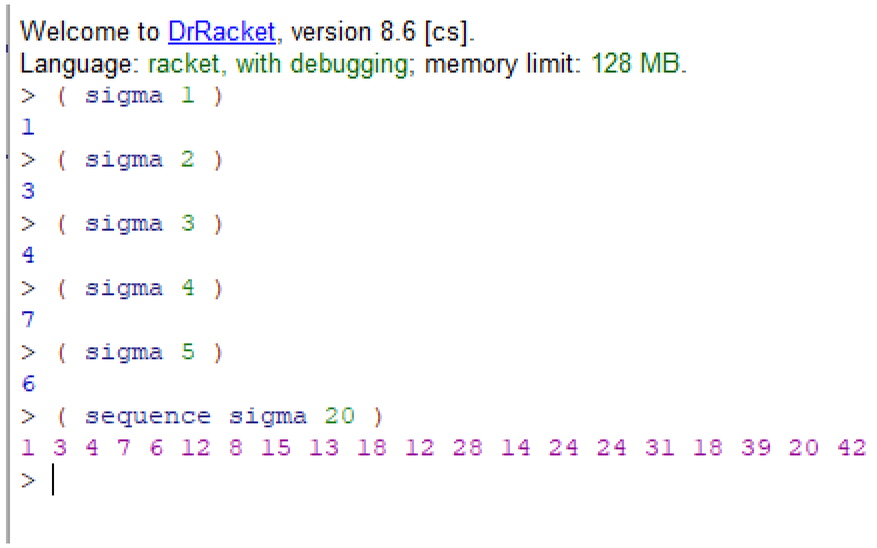
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( square 5 )
25
> ( square 10 )
100
> ( sequence square 15 )
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225
> ( cube 2 )
8
> ( cube 3 )
27
> ( sequence cube 15 )
1 8 27 64 125 216 343 512 729 1000 1331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

Triangular Demo:



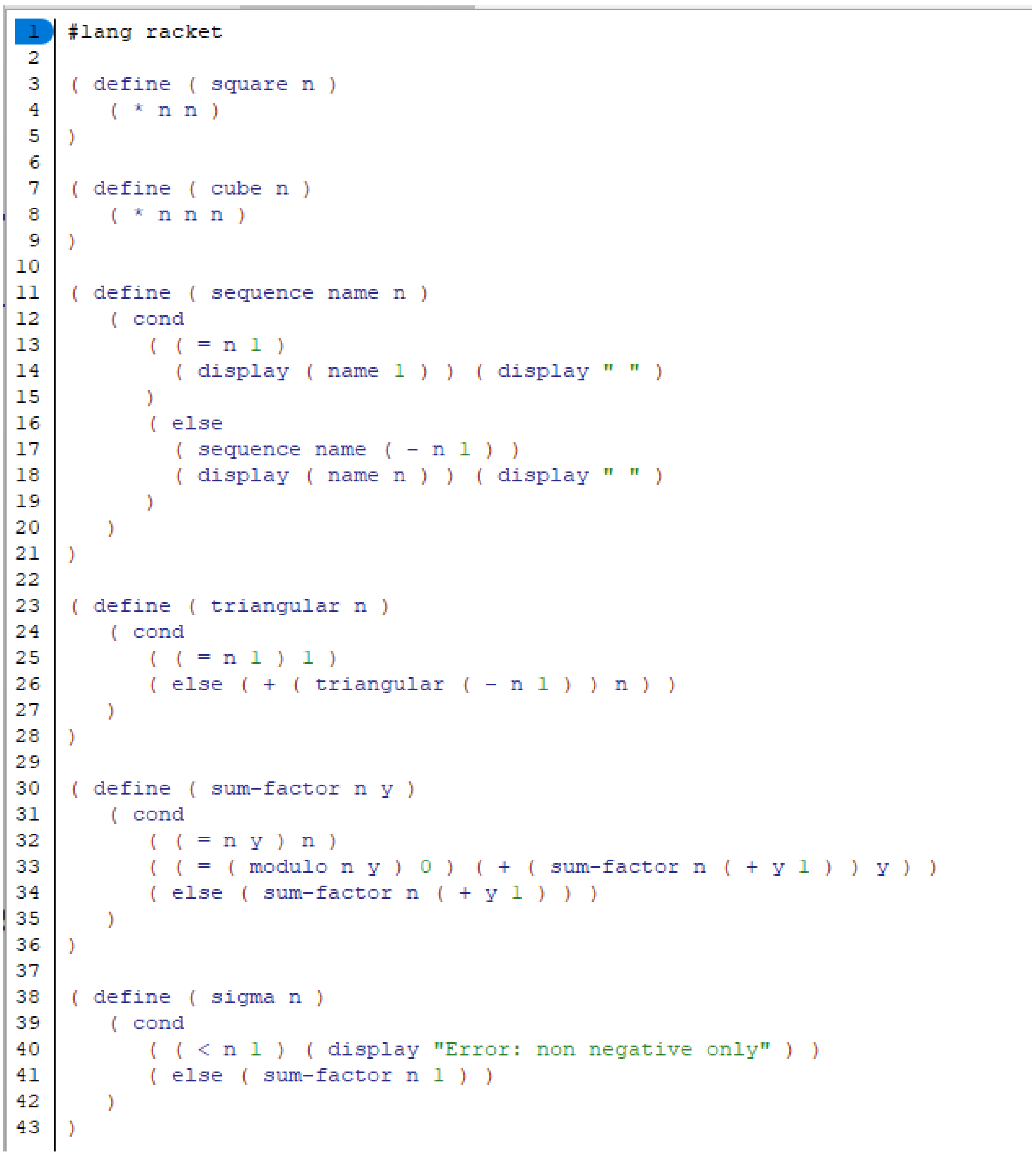
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( triangular 1 )
1
> ( triangular 2 )
3
> ( triangular 3 )
6
> ( triangular 4 )
10
> ( triangular 5 )
15
> ( sequence triangular 20 )
1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136 153 171 190 210
>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

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
> 
```

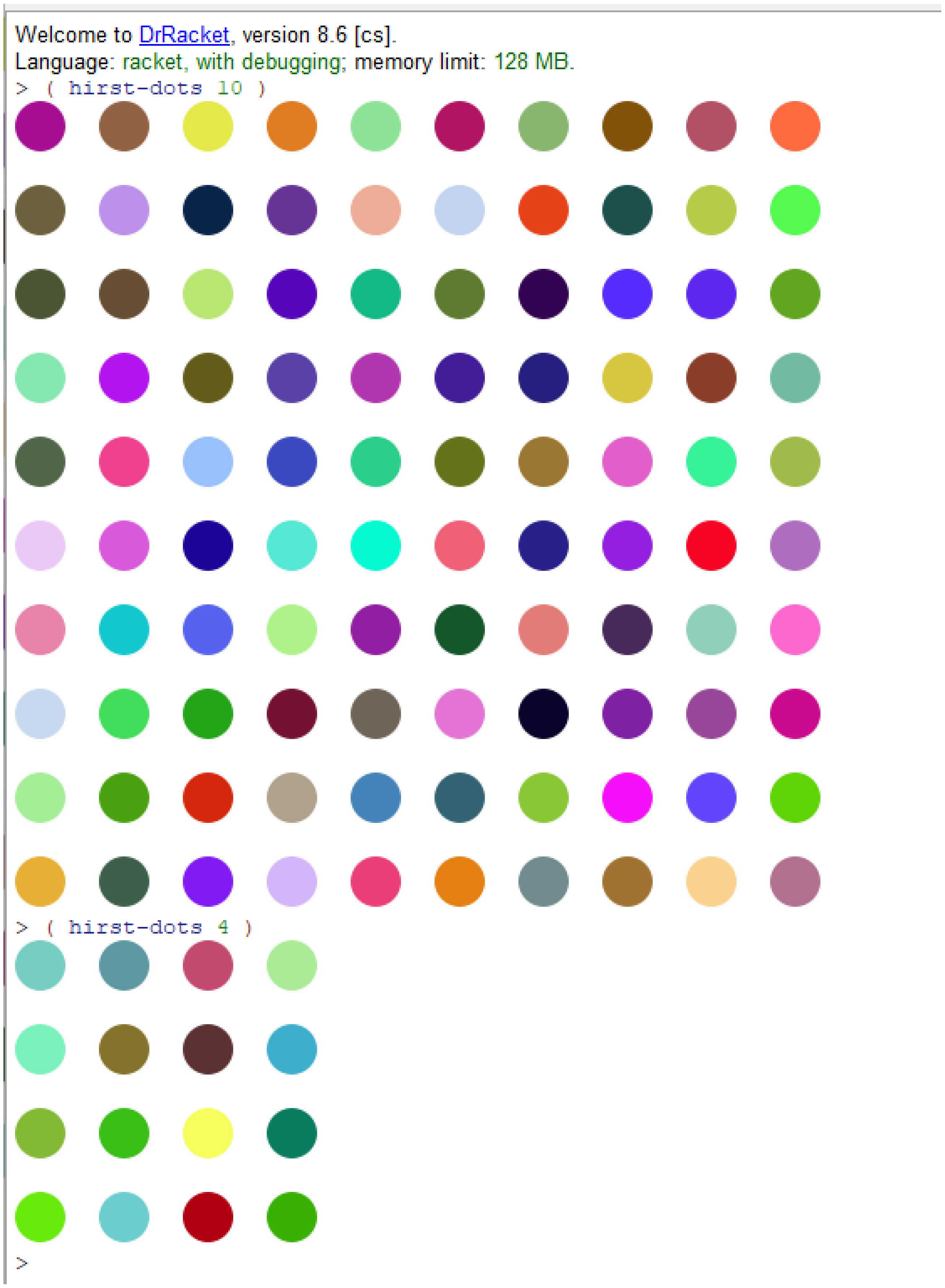
Code:



```racket
#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 1 ) 1 )
      ( else ( + ( triangular ( - n 1 ) ) n ) )
   )
)

( define ( sum-factor n y )
   ( cond
      ( ( = n y ) n )
      ( ( = ( modulo n y ) 0 ) ( + ( sum-factor n ( + y 1 ) ) y ) )
      ( else ( sum-factor n ( + y 1 ) ) )
   )
)

( define ( sigma n )
   ( cond
      ( ( < n 1 ) ( display "Error: non negative only" ) )
      ( else ( sum-factor n 1 ) )
   )
)
```

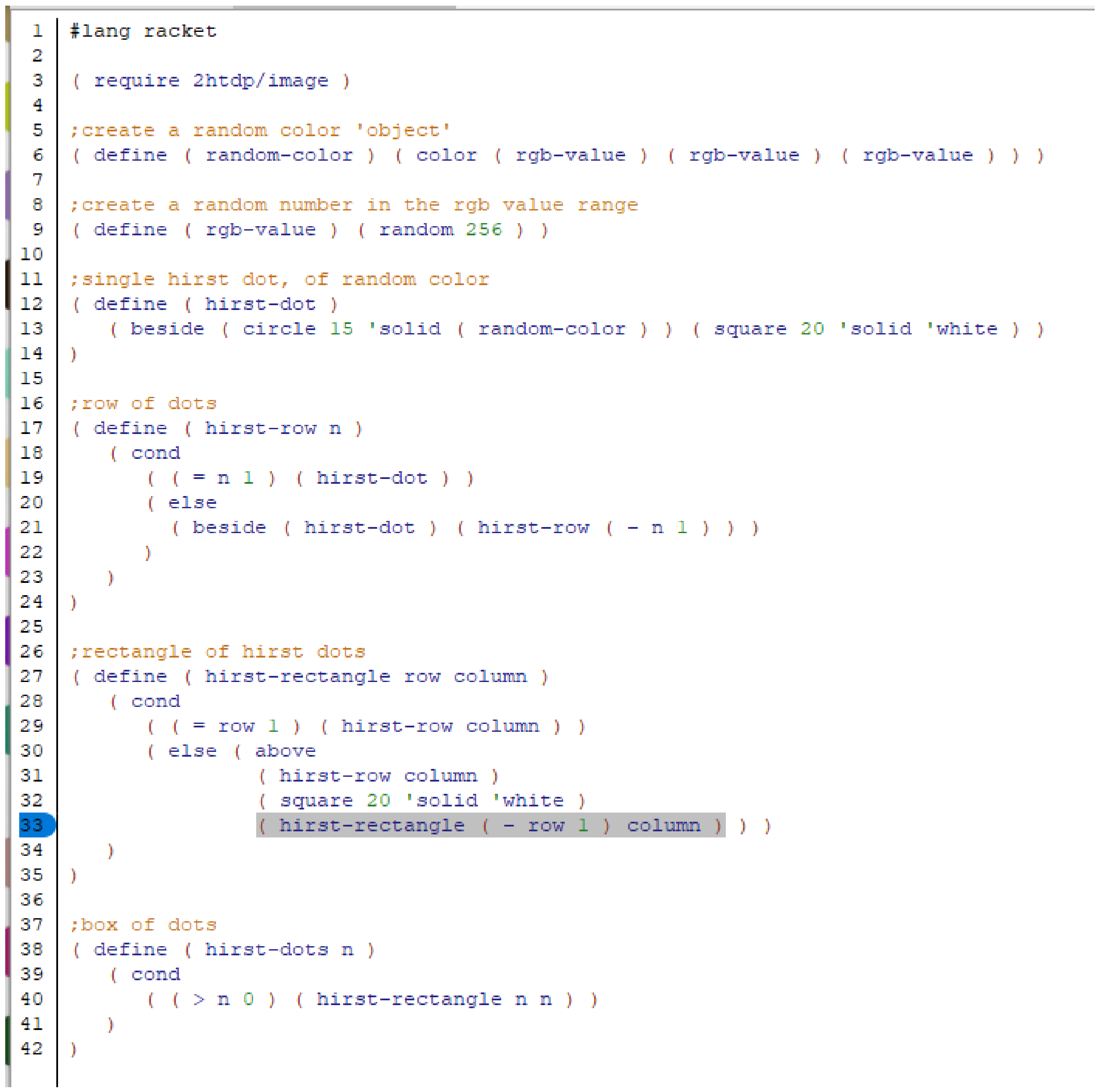
Hirst Dots   GitHUB

Demo:



```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( hirst-dots 10 )
> ( hirst-dots 4 )
>
```

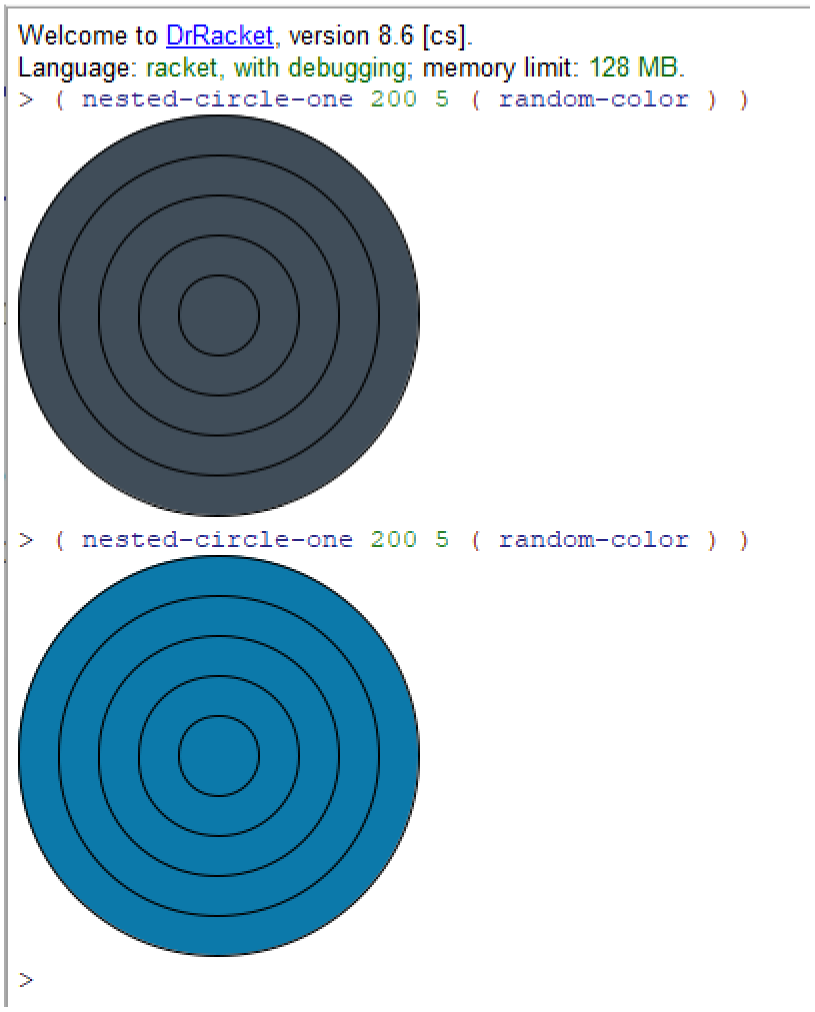
Code:



```racket
#lang racket

( require 2htdp/image )

;create a random color 'object'
( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )

;create a random number in the rgb value range
( define ( rgb-value ) ( random 256 ) )

;single hirst dot, of random color
( define ( hirst-dot )
   ( beside ( circle 15 'solid ( random-color ) ) ( square 20 'solid 'white ) )
)

;row of dots
( define ( hirst-row n )
   ( cond
      ( ( = n 1 ) ( hirst-dot ) )
      ( else
        ( beside ( hirst-dot ) ( hirst-row ( - n 1 ) ) )
      )
   )
)

;rectangle of hirst dots
( define ( hirst-rectangle row column )
   ( cond
      ( ( = row 1 ) ( hirst-row column ) )
      ( else ( above
               ( hirst-row column )
               ( square 20 'solid 'white )
               ( hirst-rectangle ( - row 1 ) column ) ) )
   )
)

;box of dots
( define ( hirst-dots n )
   ( cond
      ( ( > n 0 ) ( hirst-rectangle n n ) )
   )
)
```

Frank Stella GitHUB

Demo:

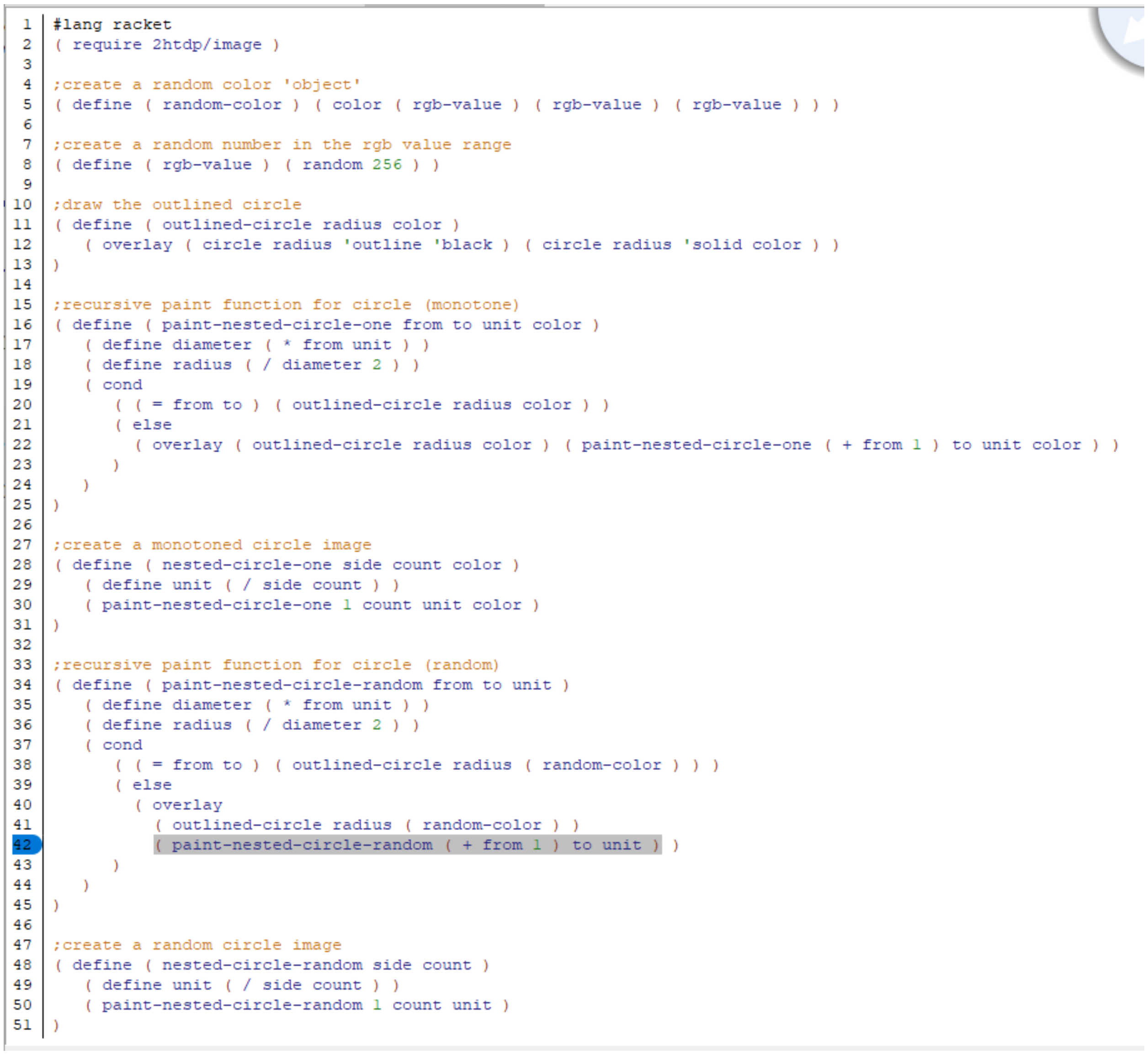


```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( nested-circle-one 200 5 ( random-color ) )
```

```
> ( nested-circle-one 200 5 ( random-color ) )
```

```
>
```

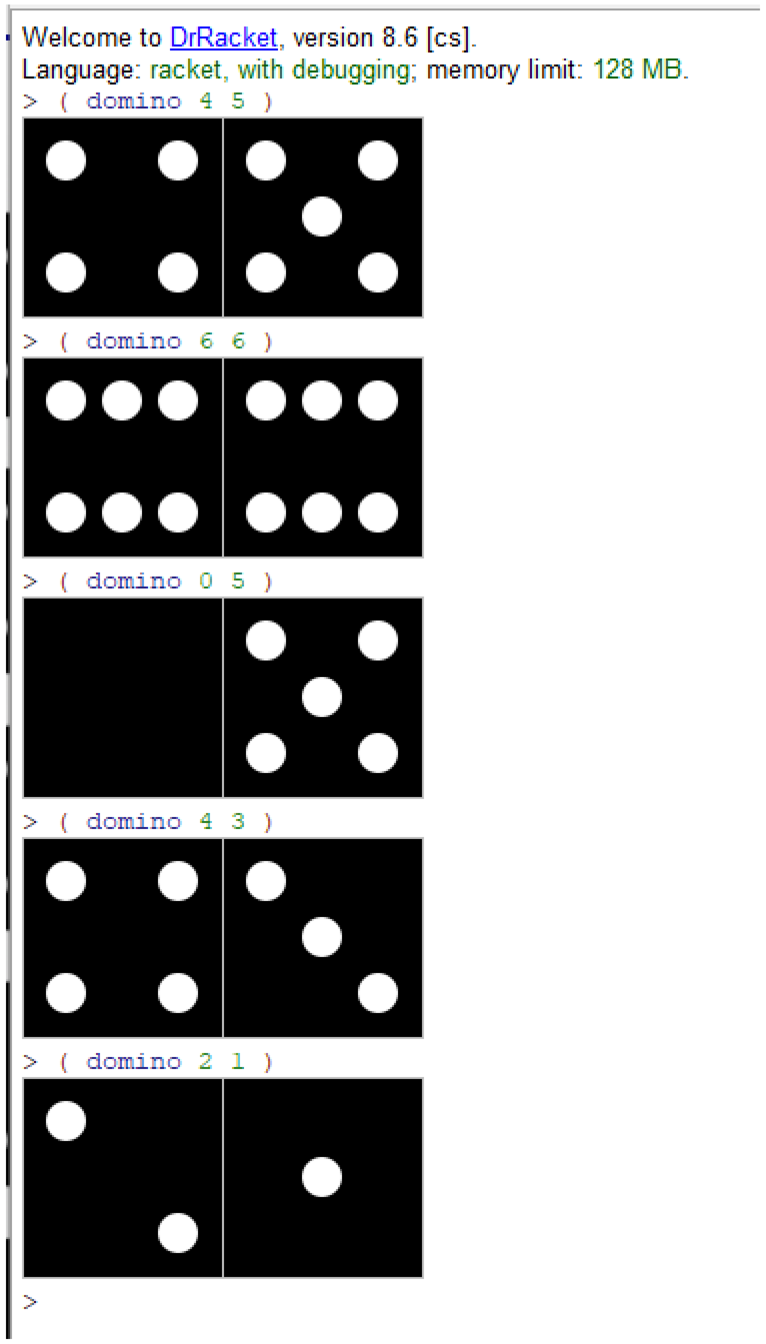
Code:



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

;create a random color 'object'
( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )

;create a random number in the rgb value range
( define ( rgb-value ) ( random 256 ) )

;draw the outlined circle
( define ( outlined-circle radius color )
   ( overlay ( circle radius 'outline 'black ) ( circle radius 'solid color ) )
)

;recursive paint function for circle (monotone)
( define ( paint-nested-circle-one from to unit color )
   ( define diameter ( * from unit ) )
   ( define radius ( / diameter 2 ) )
   ( cond
      ( ( = from to ) ( outlined-circle radius color ) )
      ( else
        ( overlay ( outlined-circle radius color ) ( paint-nested-circle-one ( + from 1 ) to unit color ) )
      )
   )
)

;create a monotoned circle image
( define ( nested-circle-one side count color )
   ( define unit ( / side count ) )
   ( paint-nested-circle-one 1 count unit color )
)

;recursive paint function for circle (random)
( define ( paint-nested-circle-random from to unit )
   ( define diameter ( * from unit ) )
   ( define radius ( / diameter 2 ) )
   ( cond
      ( ( = from to ) ( outlined-circle radius ( random-color ) ) )
      ( else
        ( overlay
          ( outlined-circle radius ( random-color ) )
          ( paint-nested-circle-random ( + from 1 ) to unit ) )
      )
   )
)

;create a random circle image
( define ( nested-circle-random side count )
   ( define unit ( / side count ) )
   ( paint-nested-circle-random 1 count unit )
)
```

Dominos GitHUB

Demo:



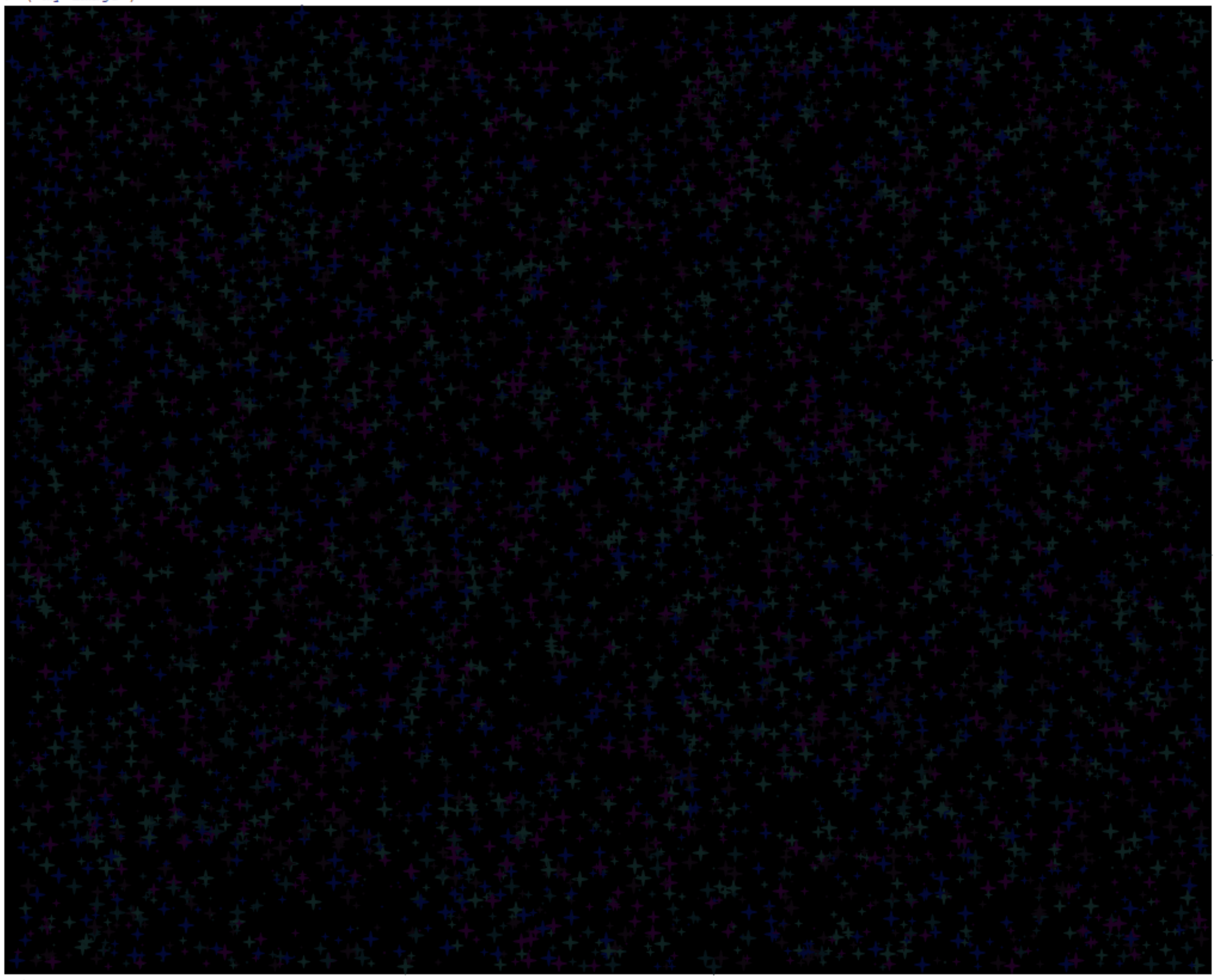
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( domino 4 5 )
> ( domino 6 6 )
> ( domino 0 5 )
> ( domino 4 3 )
> ( domino 2 1 )
>
```

Code:

```
#lang racket

;------------------------------------------------------------
;------------------   DOMINOS   -----------csc344 fall22----
;------------------------------------------------------------
( require 2htdp/image )

;------------------------------------------------------------
;----------------   VARIABLES   ---------------------------

;tile size variables
( define side-of-tile 100 )
( define diameter-of-pip ( * side-of-tile 0.2 ) )
( define radius-of-pip ( / diameter-of-pip 2 ) )
;offsets
( define d ( * diameter-of-pip 1.4 ) )
( define nd ( * -1 d ) )

;------------------------------------------------------------
;blank tile and a pip builder
( define blank-tile ( square side-of-tile 'solid 'black ) )
( define ( pip ) ( circle radius-of-pip 'solid 'white ) )

;------------------------------------------------------------
;---------------  TILE CODE  ------------------------------
;------------------------------------------------------------

;tile 1 ( tile with a single pip )
( define basic-tile1 ( overlay ( pip ) blank-tile ) )

;tile 2 ( tile with 2 pips )
( define basic-tile2
   ( overlay/offset ( pip ) d d
     ( overlay/offset ( pip ) nd nd blank-tile ) )
)

;tile 3 ( tile with 3 pips )
( define basic-tile3 ( overlay ( pip ) basic-tile2 ) )

;tile 4 ( tile with 4 pips )
( define basic-tile4 ( overlay/offset ( pip ) nd d ( overlay/offset ( pip ) d nd basic-tile2 ) ) )

;tile 5 ( tile with 5 pips )
( define basic-tile5 ( overlay ( pip ) basic-tile4 ) )

;tile 6 ( tile with 6 pips )
( define basic-tile6 ( overlay/offset ( pip ) 0 nd ( overlay/offset ( pip ) 0 d basic-tile4 ) ) )

;------------------------------------------------------------
;tile frames
( define frame ( square side-of-tile 'outline 'grey ) )

( 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  -----------------------
;------------------------------------------------------------

( 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 )
   )
)
```

My Creation (Starry Sky) GitHUB

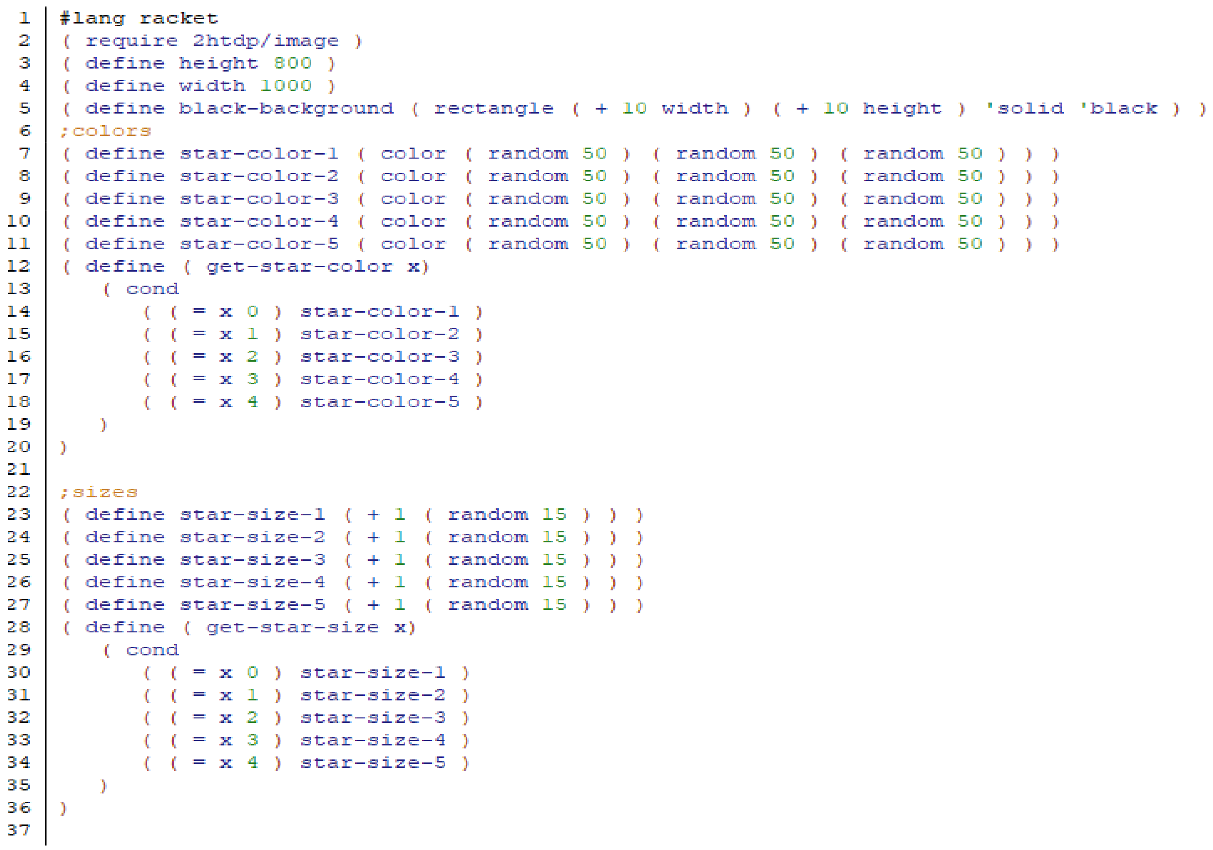
```
> ( my-image )
```



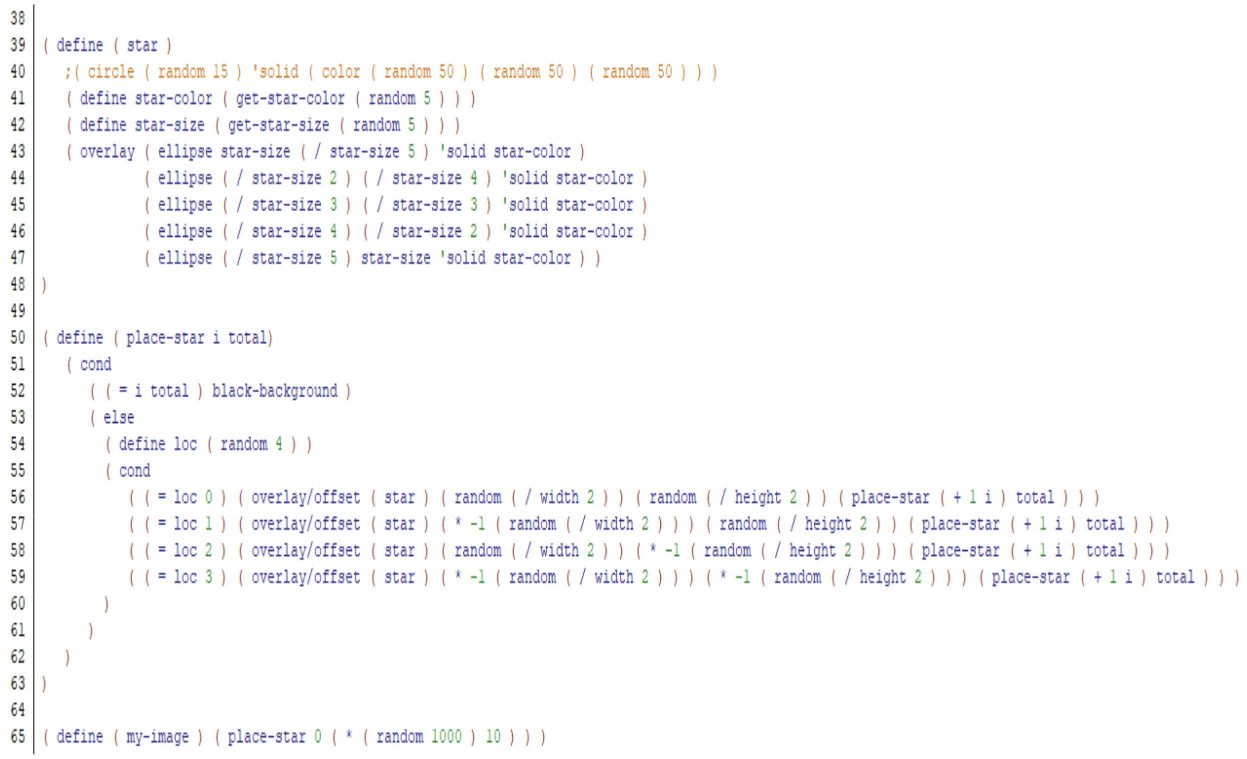
```
> ( my-image )
```

```
>
```

Code:



```
#lang racket
( require 2htdp/image )
( define height 800 )
( define width 1000 )
( define black-background ( rectangle ( + 10 width ) ( + 10 height ) 'solid 'black ) )
;colors
( define star-color-1 ( color ( random 50 ) ( random 50 ) ( random 50 ) ) )
( define star-color-2 ( color ( random 50 ) ( random 50 ) ( random 50 ) ) )
( define star-color-3 ( color ( random 50 ) ( random 50 ) ( random 50 ) ) )
( define star-color-4 ( color ( random 50 ) ( random 50 ) ( random 50 ) ) )
( define star-color-5 ( color ( random 50 ) ( random 50 ) ( random 50 ) ) )
( define ( get-star-color x)
   ( cond
      ( ( = x 0 ) star-color-1 )
      ( ( = x 1 ) star-color-2 )
      ( ( = x 2 ) star-color-3 )
      ( ( = x 3 ) star-color-4 )
      ( ( = x 4 ) star-color-5 )
   )
)

;sizes
( define star-size-1 ( + 1 ( random 15 ) ) )
( define star-size-2 ( + 1 ( random 15 ) ) )
( define star-size-3 ( + 1 ( random 15 ) ) )
( define star-size-4 ( + 1 ( random 15 ) ) )
( define star-size-5 ( + 1 ( random 15 ) ) )
( define ( get-star-size x)
   ( cond
      ( ( = x 0 ) star-size-1 )
      ( ( = x 1 ) star-size-2 )
      ( ( = x 2 ) star-size-3 )
      ( ( = x 3 ) star-size-4 )
      ( ( = x 4 ) star-size-5 )
   )
)

```



```

( define ( star )
   ;( circle ( random 15 ) 'solid ( color ( random 50 ) ( random 50 ) ( random 50 ) ) )
   ( define star-color ( get-star-color ( random 5 ) ) )
   ( define star-size ( get-star-size ( random 5 ) ) )
   ( overlay ( ellipse star-size ( / star-size 5 ) 'solid star-color )
             ( ellipse ( / star-size 2 ) ( / star-size 4 ) 'solid star-color )
             ( ellipse ( / star-size 3 ) ( / star-size 3 ) 'solid star-color )
             ( ellipse ( / star-size 4 ) ( / star-size 2 ) 'solid star-color )
             ( ellipse ( / star-size 5 ) star-size 'solid star-color ) )
)

( define ( place-star i total)
   ( cond
      ( ( = i total ) black-background )
      ( else
        ( define loc ( random 4 ) )
        ( cond
           ( ( = loc 0 ) ( overlay/offset ( star ) ( random ( / width 2 ) ) ( random ( / height 2 ) ) ( place-star ( + 1 i ) total ) ) )
           ( ( = loc 1 ) ( overlay/offset ( star ) ( * -1 ( random ( / width 2 ) ) ) ( random ( / height 2 ) ) ( place-star ( + 1 i ) total ) ) )
           ( ( = loc 2 ) ( overlay/offset ( star ) ( random ( / width 2 ) ) ( * -1 ( random ( / height 2 ) ) ) ( place-star ( + 1 i ) total ) ) )
           ( ( = loc 3 ) ( overlay/offset ( star ) ( * -1 ( random ( / width 2 ) ) ) ( * -1 ( random ( / height 2 ) ) ) ( place-star ( + 1 i ) total ) ) )
        )
      )
   )
)

( define ( my-image ) ( place-star 0 ( * ( random 1000 ) 10 ) ) )
```