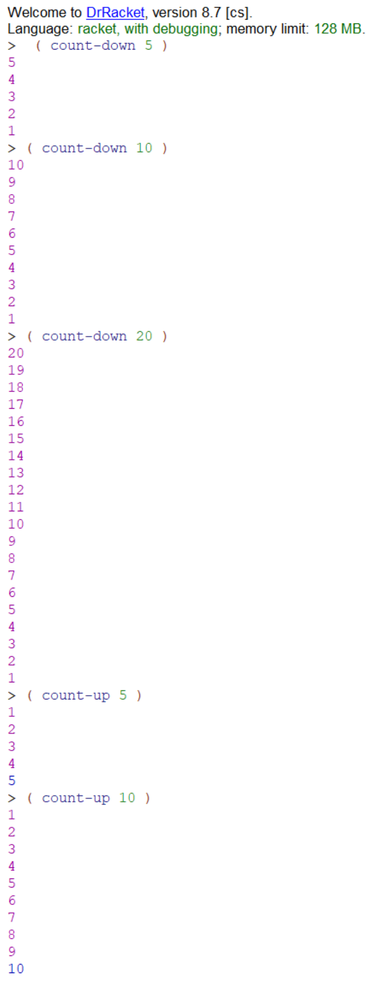
Learning Abstract: This assignment will introduce recursion in Racket and provide 5 tasks to practice it.

Task 1:

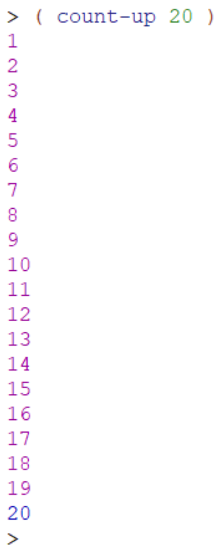
Code:

```
( define ( count-down integer)
   ( cond
      ( ( = integer 1 )
        ( display integer )
      )
      ( ( > integer 0)
        ( display  integer  )
        ( display  "\n" )
        ( count-down ( - integer 1 ) )
      )
    )
 )


( define ( count-up integer)
   ( define ( count-up-further integer final-num)
   ( cond
      ( ( <  integer  final-num )
        ( display  integer )
           ( display  "\n" )
           ( count-up-further (+ integer 1)  final-num )
      )
        ( = integer final-num )
           ( display "" )
        )
    )
   ( define final-num integer )
   ( cond
      ( ( = integer 0 )
        ( display "" )
      )
      ( ( > integer 0)
        ( display 1 )
        ( display  "\n" )
        ( count-up-further 2  final-num  )
        )
      )
   )
```

Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
>  ( count-down 5 )
5
4
3
2
1
> ( count-down 10 )
10
9
8
7
6
5
4
3
2
1
> ( count-down 20 )
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
> ( count-up 5 )
1
2
3
4
5
> ( count-up 10 )
1
2
3
4
5
6
7
8
9
10
```



```
> ( count-up 20 )
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
>
```

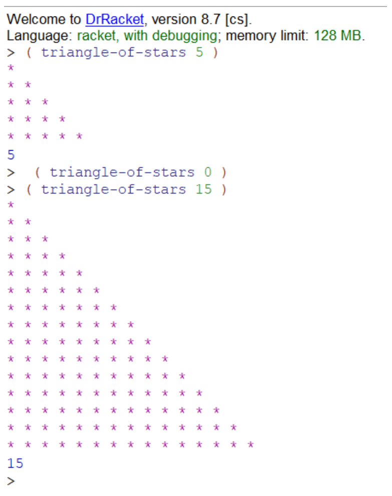
Task 2:

Code:

```
( define ( triangle-of-stars integer )
  ( define current-num 1 )

  ( define ( row-of-stars n )
    ( cond
      ( ( = n 0 )
        ( display "\n" )
        )
       ( ( > n 0 )
        ( display "* " )
        ( row-of-stars ( - n 1 ) )
        )
      )
   )

  ( define ( triangle-of-stars-further integer final-num)
  ( cond
    (
     ( < integer  ( + final-num 1 ) )
        ( row-of-stars integer )
        ( triangle-of-stars-further ( + integer 1) final-num )
    )
       ( = integer final-num )
```

```
        ( row-of-stars integer )
      )
    )

    ( cond
       ( ( = integer 0 )
         ( display "" )
         )
        ( ( > integer 0 )
           ( row-of-stars current-num )
           ( triangle-of-stars-further (+ current-num 1) integer )
         )
      )
)
```

Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( triangle-of-stars 5 )
*
* *
* * *
* * * *
* * * * *
5
>  ( triangle-of-stars 0 )
> ( triangle-of-stars 15 )
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * * *
* * * * * * * * * * * * *
* * * * * * * * * * * * * *
* * * * * * * * * * * * * * *
15
>
```

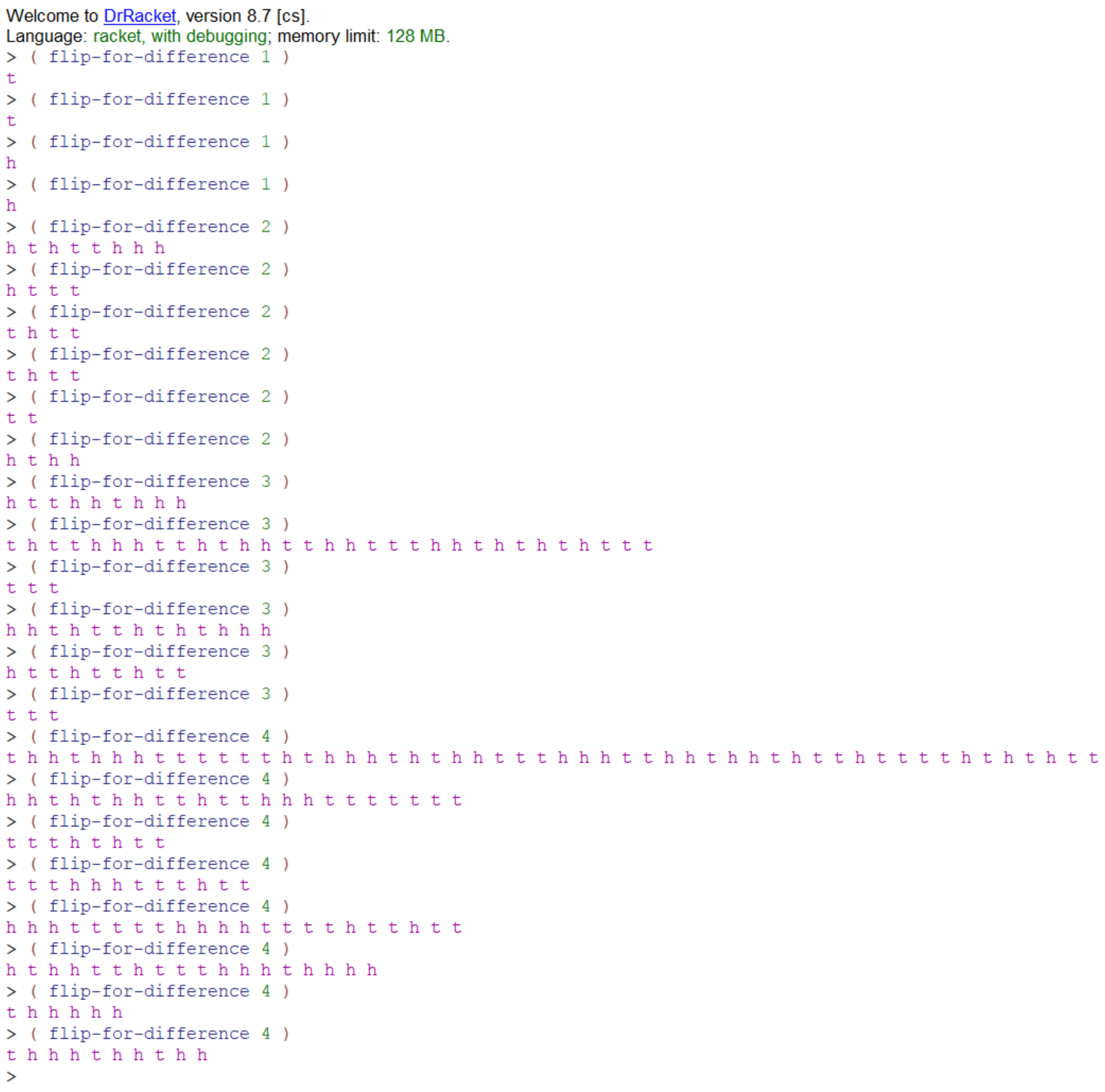
Task 3:

Code:

```
( define ( flip-for-difference difference-allowed )
  ( define ( flip-coin )
  (define outcome ( random 2 ) )
     ( cond
        ( ( = outcome 0 ) 't )
        ( ( = outcome 1 ) 'h ))
     outcome
   )


  ( define  ( flip-for-difference-further difference-allowed current-difference )
     ( define negative-value (* difference-allowed -1) )
     ( define value ( flip-coin ) )
     (cond
       [(and (< current-difference difference-allowed) (> current-difference negative-value ) )
           ( cond
        [(eq? value 0)
             (display "t ")
             ( flip-for-difference-further difference-allowed ( - current-difference 1 ) )
         ]
         [(eq? value 1 )
              ( display "h " )
              ( flip-for-difference-further difference-allowed ( + current-difference 1 ) ) ]
        )]
```

```
      [else (display "")] )
    )
  ( flip-for-difference-further difference-allowed 0 )
)
```

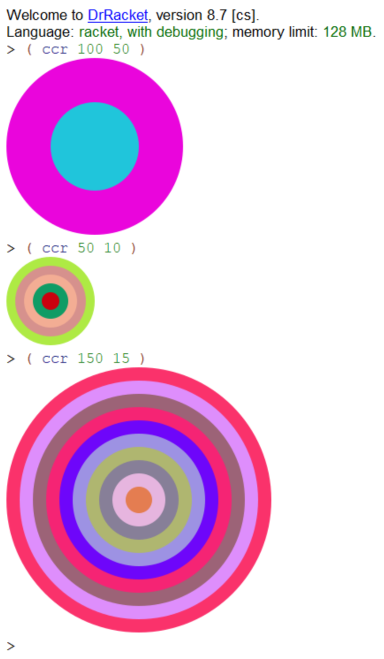
Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( flip-for-difference 1 )
t
> ( flip-for-difference 1 )
t
> ( flip-for-difference 1 )
h
> ( flip-for-difference 1 )
h
> ( flip-for-difference 2 )
h t h t t h h h
> ( flip-for-difference 2 )
h t t t
> ( flip-for-difference 2 )
t h t t
> ( flip-for-difference 2 )
t h t t
> ( flip-for-difference 2 )
t t
> ( flip-for-difference 2 )
h t h h
> ( flip-for-difference 3 )
h t t h h t h h h
> ( flip-for-difference 3 )
t h t t h h h t t h t h h t t h h t t t h h t h t h t h t t t
> ( flip-for-difference 3 )
t t t
> ( flip-for-difference 3 )
h h t h t t h t h t h h h
> ( flip-for-difference 3 )
h t t h t t h t t
> ( flip-for-difference 3 )
t t t
> ( flip-for-difference 4 )
t h h t h h h t t t t t t h t h h h t h t h h t t t h h h t t h h t h h t h t t h t t t t h t h t h t t
> ( flip-for-difference 4 )
h h t h t h h t t h t t h h h t t t t t t t
> ( flip-for-difference 4 )
t t t h t h t t
> ( flip-for-difference 4 )
t t t h h h t t t h t t
> ( flip-for-difference 4 )
h h h t t t t t h h h h t t t t h t t h t t
> ( flip-for-difference 4 )
h t h h t t h t t t h h h t h h h h
> ( flip-for-difference 4 )
t h h h h h
> ( flip-for-difference 4 )
t h h h t h h t h h
>
```

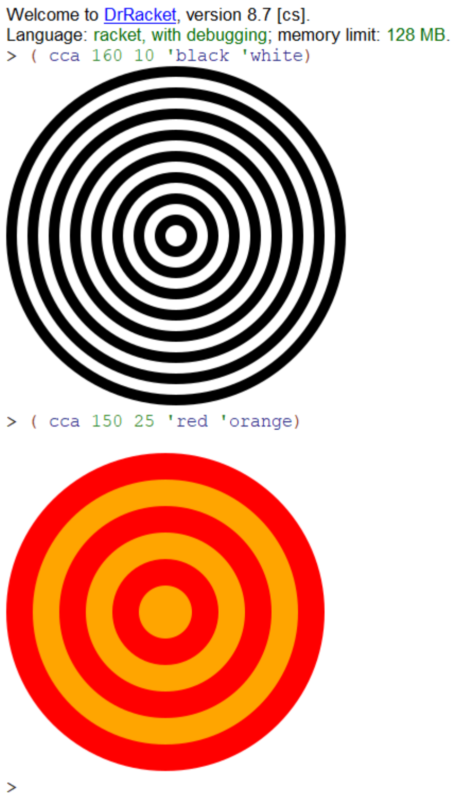
Task 4:

CCR Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( ccr 100 50 )
> ( ccr 50 10 )
> ( ccr 150 15 )
>
```

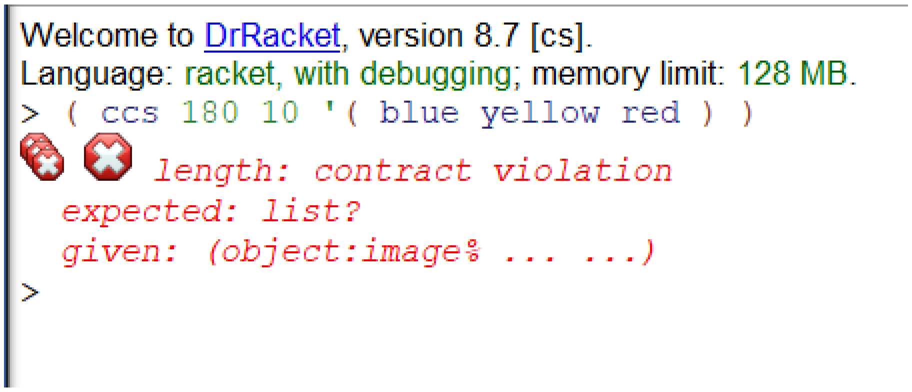
CCA Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( cca 160 10 'black 'white)
> ( cca 150 25 'red 'orange)
>
```

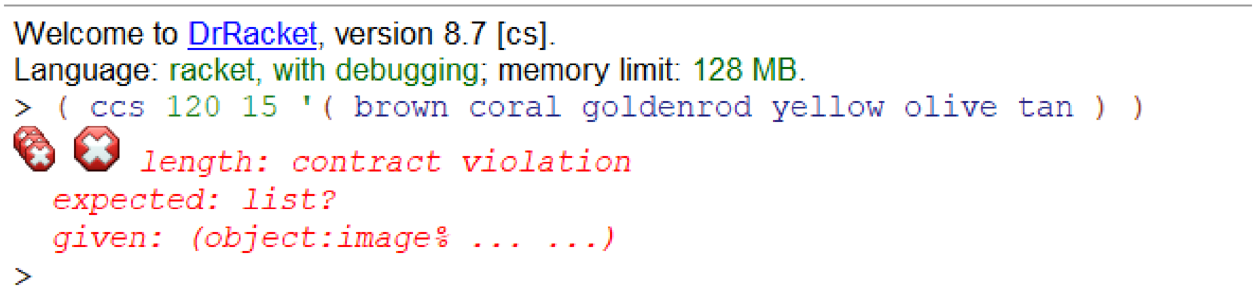
CCS Demo 1:

For this one, I tried for a very long time to get the colors from the list but for the life of me could not figure out why Racket was seeing the list as "(object:image% … …)". Because of this I was unable to establish a random color from the list through each recursion. However, I think besides this all the code should be correct. Unfortunately, I couldn't get successful demoes with this task.



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( ccs 180 10 '( blue yellow red ) )
 length: contract violation
  expected: list?
  given: (object:image% ... ...)
>
```

CCS Demo 2:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( ccs 120 15 '( brown coral goldenrod yellow olive tan ) )
 length: contract violation
  expected: list?
  given: (object:image% ... ...)
>
```

Code:

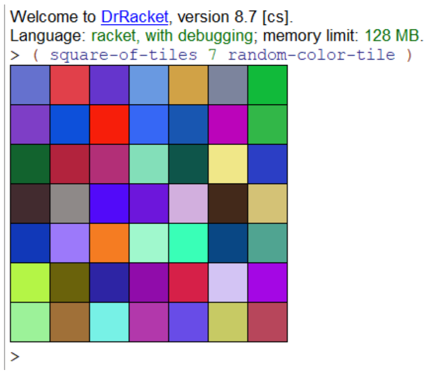
```
( define ( ccr radius radius-diff )
  ( define ( ccr-further radius radius-diff curr-circle )
    ( cond
      [ ( > ( - radius radius-diff ) 0 )
```

```
          ( define new-radius (- radius radius-diff) )
          ( define new-circle ( circle new-radius  "solid" ( rc ) ) )
          ( define result-circle ( overlay new-circle curr-circle) )
          ( ccr-further new-radius radius-diff result-circle )
          result-circle
          ]
          [ else
            ( display curr-circle )
          ]
      )
    )
  ( define ( rbg ) ( random 0 256 ) )
  ( define ( rc ) ( color ( rbg ) ( rbg ) ( rbg ) ) )
  ( define curr-circle (circle radius "solid" ( rc ) ) )
  ( ccr-further radius radius-diff curr-circle )
  ( display "" )
)

( define ( cca radius radius-diff color1 color2 )
  ( define ( cca-further radius radius-diff curr-circle curr-num )
    (define new-curr-num ( + curr-num 1 ) )
    ( cond
      [ ( > ( - radius radius-diff ) 0 )
        ( cond [(eq? (modulo curr-num 2 ) 0 )
            ( define new-radius (- radius radius-diff) )
```

```
              ( define new-circle ( circle new-radius  "solid" color1 ) )
              ( define result-circle ( overlay new-circle curr-circle) )
              ( cca-further new-radius radius-diff result-circle new-curr-num )
              result-circle
            ]
            [ else
              ( define new-radius (- radius radius-diff) )
              ( define new-circle ( circle new-radius  "solid" color2 ) )
              ( define result-circle ( overlay new-circle curr-circle) )
              ( cca-further new-radius radius-diff result-circle new-curr-num )
              result-circle
            ]
           )
          ]
          [ else
            ( display curr-circle )
          ]
      )
    )
  ( define curr-circle (circle radius "solid" color1 ) )
  ( cca-further radius radius-diff curr-circle 1 )
  ( display "" )
 )

( define ( ccs radius radius-diff color-list )
 ( define ( random-color color-list )
    (list-ref  color-list  (random (length color-list ) ) )
    )


 ( define ( ccs-further radius radius-diff color-list curr-circle )
    ( define curr-color ( random-color color-list ) )
    ( define new-radius (- radius radius-diff) )
    ( define new-circle ( circle new-radius "solid" curr-color ) )
    ( define result-circle ( overlay new-circle curr-circle) )
    ( ccs-further new-radius radius-diff color-list result-circle )
    result-circle
)


    ( define curr-circle (circle radius "solid" ( random-color color-list ) ) )
    ( ccs-further radius radius-diff curr-circle 1 )
    ( display "" )
)
```

Task 5:

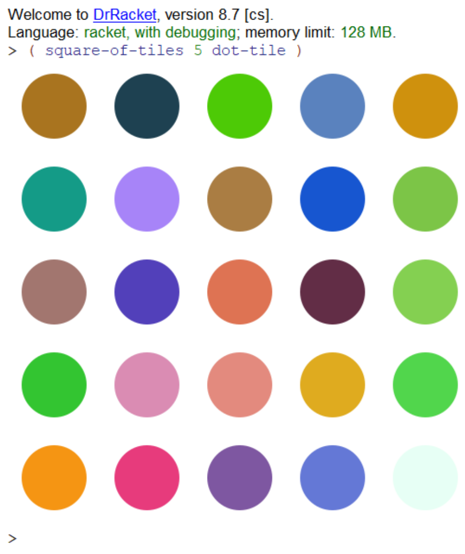
Random Colored Tile Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( square-of-tiles 7 random-color-tile )
```

```
>
```

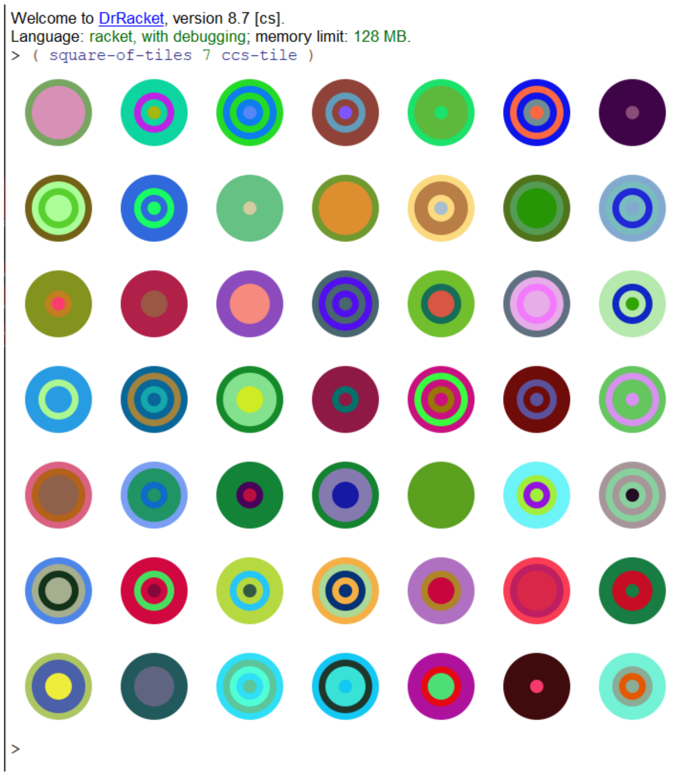
Hirst Dots Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( square-of-tiles 5 dot-tile )
```

```
>
```

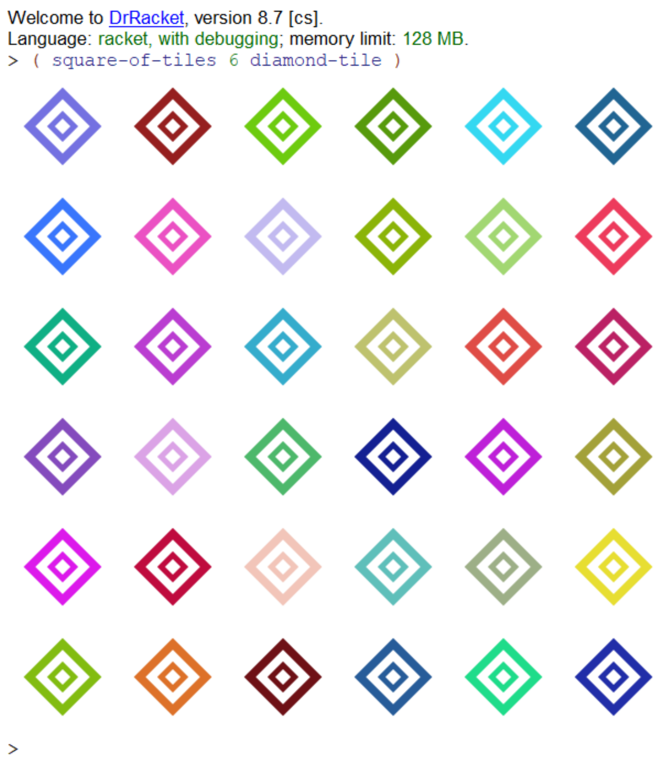
CCS Dots Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( square-of-tiles 7 ccs-tile )
```

```
>
```

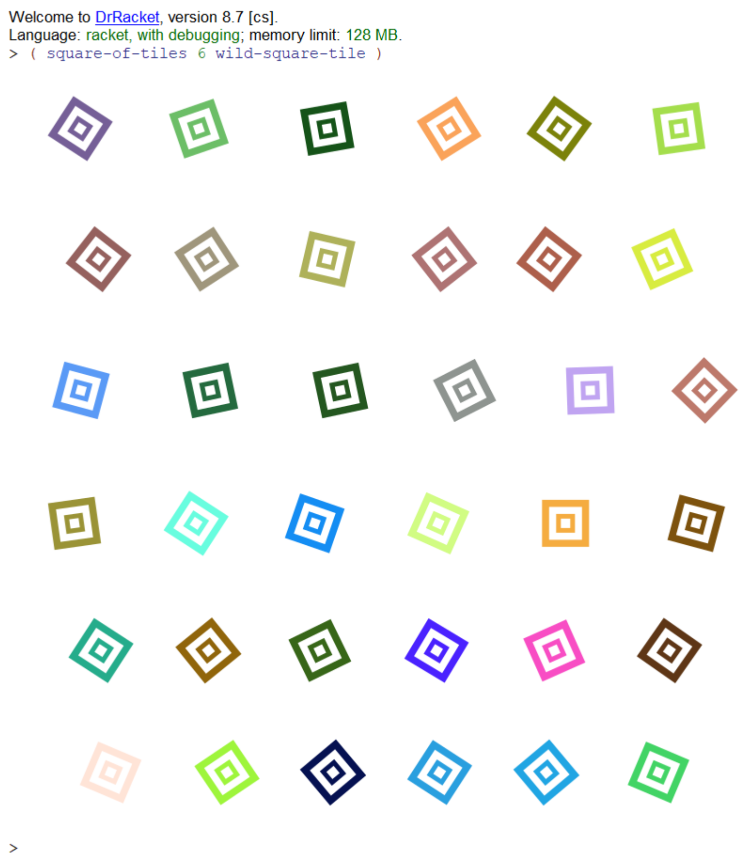
Nested Diamonds Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( square-of-tiles 6 diamond-tile )
```

```
>
```

Unruly Squares Demo:



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( square-of-tiles 6 wild-square-tile )
```

```
>
```

Code:

```
( define ( row-of-tiles n tile )
( cond
  ( ( = n 0)
    empty-image
  )

  ( ( > n 0)
    ( beside ( row-of-tiles ( - n 1 ) tile ) ( tile ) )
```

```
   )
  )
)

( define ( rectangle-of-tiles r c tile )
   ( cond
      ( ( = r 0 )
        empty-image
      )
      ( ( > r 0 )
        ( above( rectangle-of-tiles ( - r 1 ) c tile ) ( row-of-tiles c tile ) )
      )
   )
 )

( define ( square-of-tiles n tile )
   ( rectangle-of-tiles n n tile )
)

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

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

```
( define ( random-color-tile )
   ( overlay
     ( square 40 "outline" "black" )
     ( square 40 "solid" ( random-color ) )
   )
)

( define ( dot-tile )
   ( overlay
     ( circle 35 "solid" ( random-color ) )
     ( square 100 "solid" "white" )
   )
 )

( define ( ccs-tile )
   ( define color-1 ( random-color ) )
   ( define color-2 ( random-color ) )
   ( define color-3 ( random-color ) )
   ( define list-1 ( list color-1 color-2 color-3 ) )

   ( define ( random-color-from-list color-list )
     (list-ref  color-list  (random 3))
      )

( overlay
  (  circle 7 "solid" ( random-color-from-list list-1 ) )
    ( overlay
      (  circle 14 "solid" ( random-color-from-list list-1 ) )
      ( overlay
       ( circle 21 "solid" ( random-color-from-list list-1 ) )
       ( overlay
         ( circle 28 "solid" ( random-color-from-list list-1 ) )
         ( overlay
           ( circle 35 "solid" ( random-color-from-list list-1 ) )
           ( square 100 "solid" "white" )
          )
         )
       )
     )
   )
 )

( define ( diamond-tile )
   ( define color ( random-color ) )

   ( overlay
     ( rotate 45 ( square 10 "solid" "white" ) )
     ( overlay
       ( rotate 45 ( square 20 "solid" color ) )
       ( overlay
         ( overlay
           ( rotate 45 ( square 35 "solid" "white" ) )
           ( rotate 45 ( square 50 "solid" color ) )
          )
         ( square 100 "solid" "white" )
        )
      )
    )
 )

( define ( wild-square-tile )
    ( rotate ( random 360 ) ( diamond-tile ) )
 )
```