Declan ONUNKWO 02/29/2023

CSC344 – Programming Languages

Assignment: **Racket Assignment 3 – Recursions in Racket**

LEARNING ABSTRACT

The project is aimed to explore the concept of recursions and its applications in the Racket programming language. I was able to gain an in-depth understanding of the theory and practices of recursion, as well as develop practical skills in racket programming. This project helped me to develop a deeper appreciation for the power and elegance of recursive programming, as well as to improve my problem-solving abilities and programming skills.

Task 1: Counting Down, Counting Up

Code

```
#lang racket
( define ( count-down int )
   ( cond ( ( > int 0 )
            ( display int )
            ( display "\n" )
            ( count-down ( - int 1 )
            )
          )
   )
)

( define ( count-up int )
   ( cond ( ( = int 1 )
            ( display int )
            ( display "\n" )
            )
          (else ( count-up ( - int 1 ) )
                ( display int )
                ( display "\n" )
                )
          )
   )
```

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: Triangle of Stars

Code

```
24  ( define ( triangle-of-stars int )
25     ( cond ( ( = int 0 )
26              ( display "" )
27              )
28            (else ( triangle-of-stars ( - int 1 ) )
29                  (displayln ( string-join ( make-list int "*" ) " " ) )
30                  )
31            )
32     )
```

Demo

Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( triangle-of-stars 5 )
*
* *
* * *
* * * *
* * * * *
> ( triangle-of-stars 0 )
> ( triangle-of-stars 15 )
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * * *
* * * * * * * * * * * * *
* * * * * * * * * * * * * *
* * * * * * * * * * * * * * *
>
```

Task 3: Flipping a Coin

Code

```
34 ( define ( flip-for-difference int )
35    ( define ( flip )
36      ( if ( = ( random 2 ) 0 )
37           'h 't )
38    )
39    ( define ( flip-helper int amount )
40       ( define negate ( * int -1 ) )
41       ( cond (
42               ( and ( < amount int ) ( > amount negate ) )
43               ( let (
44                      ( value ( flip ) ) )
45                  ( display ( if ( eq? value 't ) "t" "h" ) )
46                  ( flip-helper int ( if ( eq? value 't ) ( - amount 1 ) ( + amount 1 )
47                                      )
48                                  )
49                  )
50               )
51              ( else ( display "" ) )
52              )
53       )
54    ( flip-helper int 0 )
55    )
```

Demo

Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.

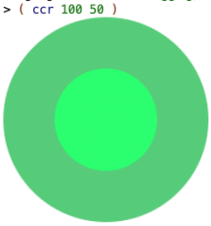
```
> ( flip-for-difference 1 )
t
> ( flip-for-difference 1 )
t
> ( flip-for-difference 1 )
t
> ( flip-for-difference 1 )
h
> ( flip-for-difference 2 )
ththhtthhh
> ( flip-for-difference 2 )
tt
> ( flip-for-difference 2 )
hh
> ( flip-for-difference 2 )
hthtthtt
> ( flip-for-difference 2 )
hthhthh
> ( flip-for-difference 2 )
hh
```

```
> ( flip-for-difference 3 )
htthttt
> ( flip-for-difference 3 )
hhh
> ( flip-for-difference 3 )
ttt
> ( flip-for-difference 3 )
ttt
> ( flip-for-difference 3 )
ththhtttt
> ( flip-for-difference 3 )
hhttthhhttttthhhhtthhthh
> ( flip-for-difference 4 )
thhhthhh
> ( flip-for-difference 4 )
ththhhttttthhhhh
> ( flip-for-difference 4 )
tthhhhttttthttt
> ( flip-for-difference 4 )
hhththhthh
> ( flip-for-difference 4 )
hhtthhhhh
> ( flip-for-difference 4 )
hhthtthtthtttt
> ( flip-for-difference 4 )
httttt
> ( flip-for-difference 4 )
ttthhhhhthhthtthttthhthttttttt
>
```

Task 4: Laying Down Colorful Concentric Disks

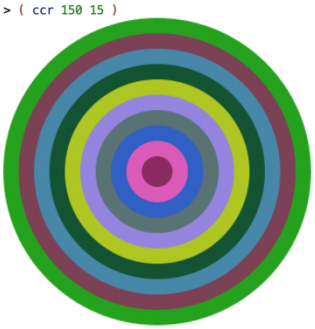
CCR Demo

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



```
> ( ccr 50 10 )
```

```
> ( ccr 150 15 )
```

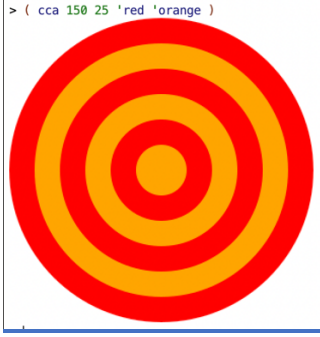


CCA Demo

Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( cca 160 10 'black 'white )
```

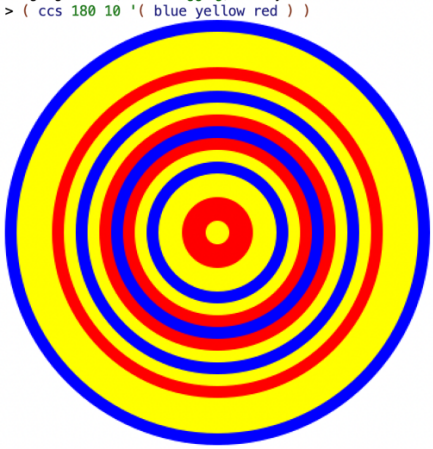
```
> ( cca 150 25 'red 'orange )
```



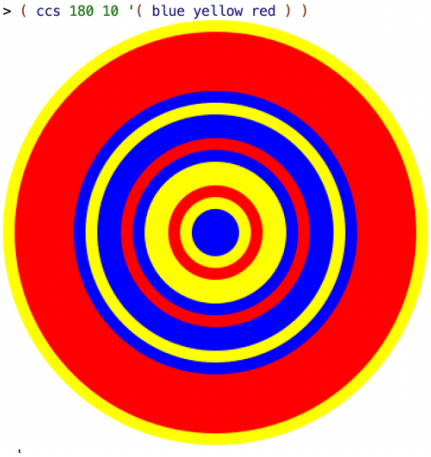
CCS Demo 1

Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( ccs 180 10 '( blue yellow red ) )
```



```
> ( ccs 180 10 '( blue yellow red ) )
```



CCS Demo 2

```
> ( ccs 120 15 '( brown coral goldenrod yellow olive tan ) )
```

```
> ( ccs 120 15 '( brown coral goldenrod yellow olive tan ) )
```

Code

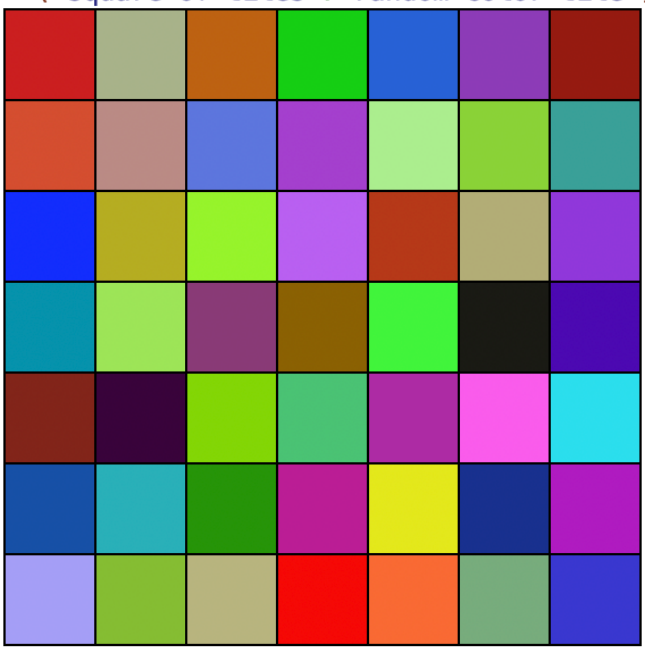
```
( define ( rndColor )
   ( color ( random 256 ) ( random 256 ) ( random 256 ) )
   )

( define ( ccr radius radiusDifference )
   ( cond ( ( <= radius 0 ) empty-image )
          ( else ( overlay ( ccr ( - radius radiusDifference ) radiusDifference )
                           ( circle radius "solid" ( rndColor ) )
                           )
                 )
          )
   )

( define ( cca radius difference c1 c2 )
   ( cond ( ( <= radius 0 ) empty-image )
          ( else ( overlay ( cca ( - radius difference ) difference c1 c2 )
                           ( circle radius "solid"
                                    ( if ( even? ( round ( / radius difference ) ) )
                                         c1 c2 )
                           )
                 )
          )
   )
)

( define ( ccs radius difference color-list )
   ( cond ( ( <= radius 0 ) empty-image )
          ( else (
                   let (
                         ( myColor ( list-ref color-list ( random ( length
                                                                    color-list )
                                                           )
                                   )
                         )
                    ( overlay ( ccs ( - radius difference ) difference color-list )
                              ( circle radius "solid" myColor)
                              )
                    )
                 )
          )
   )
```

Task 5: Variations on Hirst Dots

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

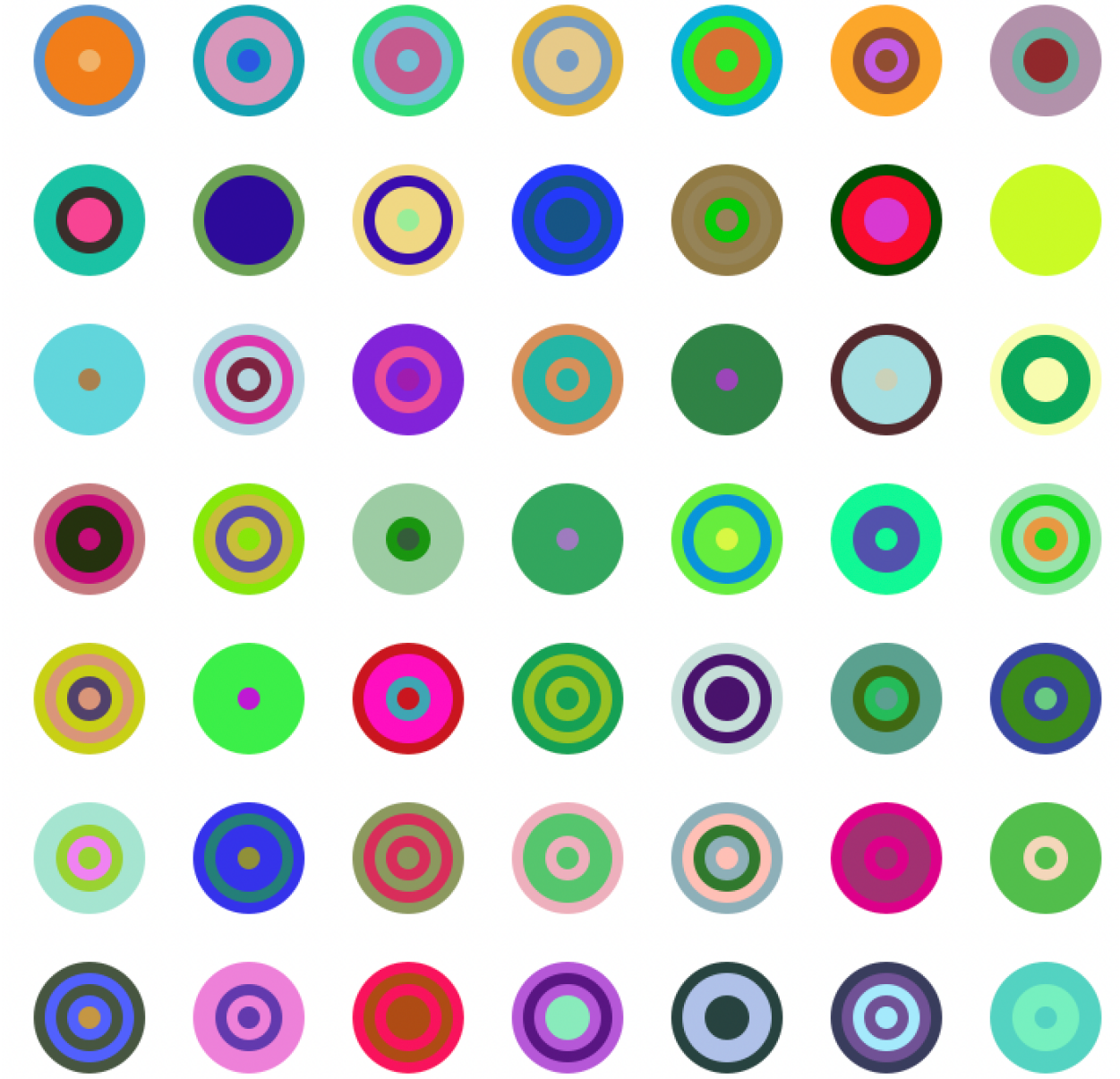
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.

```
> ( 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 Square 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 ( random-color-tile )
   ( overlay ( square 40 "outline" "black" )
             ( square 40 "solid" ( rndColor ) )
             )
   )

( define ( dot-tile )
   ( overlay ( circle 35 "solid" ( rndColor ) )
             ( square 100 0 "white" )
             )
   )

( 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 ( random-color-list n )
   ( cond ( ( = n 0 ) empty )
          ( else ( cons ( rndColor ) ( random-color-list ( - n 1 ) ) ) )
                 )
          )
   )

( define ( ccs-tile )
   ( overlay ( ccs 35 7 ( random-color-list 3 ) )
             ( square 100 0 "white" )
             )
   )

( define ( diamond-tile )
   ( define colorToUse ( rndColor ) )
   ( overlay ( rotate 45 ( square 20 "solid" "white" ) )
             ( rotate 45 ( square 30 "solid" colorToUse ) )
             ( rotate 45 ( square 40 "solid" "white" ) )
             ( rotate 45 ( square 50 "solid" colorToUse ) )
             ( square 100 0 "white" )
             )
   )

( define ( wild-square-tile )
   ( define colorToUse ( rndColor ) )
   ( define randomDegree ( random 180 ) )
   ( overlay ( rotate randomDegree ( square 20 "solid" "white" ) )
             ( rotate randomDegree ( square 30 "solid" colorToUse ) )
             ( rotate randomDegree ( square 40 "solid" "white" ) )
             ( rotate randomDegree ( square 50 "solid" colorToUse ) )
             ( square 100 0 "white" )
             )
   )
```