Temitope Emokpae 2/28/2023

CSC 344

Racket Assignment #3: Recursions in Racket

Learning Abstract

This programming assignment is about doing recursive programming in Racket.

Task 1: Counting Down, Counting Up

Code:

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

( define ( count-up n )
   ( cond
      ( ( > n 0 )
        ( count-up ( - n 1 ) )
        ( display n )
        ( display "\n" )
        )
      )
   )
```

Demo:

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

```
( define ( star-lines n )
   ( cond
      ( ( > n 0 )
        ( display "* " )
        ( star-lines ( - n 1 ) )
       )
      )
   )

( define ( triangle-of-stars n )
   ( cond
      ( ( > n 0 )
        ( triangle-of-stars ( - n 1 ) )
        ( star-lines n )
        ( display "\n" )
       )
      )
   )
```

Demo:

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( triangle-of-stars 5 )
*
* *
* * *
* * * *
* * * * *
> ( triangle-of-stars 0 )
> ( triangle-of-stars 15 )
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * * *
* * * * * * * * * * * * *
* * * * * * * * * * * * * *
* * * * * * * * * * * * * * *
>
```

Task 3: Flipping a Coin

Code:

```
#lang racket
( define ( flip-for-difference n )
   ( define ( coin-flip )
      ( define outcome ( random 2 ) )
      ( cond
         ( ( = outcome 0 ) 't )
         ( ( = outcome 1 ) 'h ) ) outcome )
   ( define ( helper n count )
      ( define negative ( * n -1 ) )
      ( define result ( coin-flip ) )
      ( cond
         ( ( and ( < count n ) ( > count negative ) )
          ( cond
             ( ( eq? result 0 )
               ( display "t " )
               ( helper n ( - count 1 ) ) )
             ( ( eq? result 1 )
              ( display "h " )
              ( helper n ( + count 1 ) ) )
             )
          )
         ( else ( display "" ) ) )
      )
   ( helper n 0 )
   )
```

Demo:

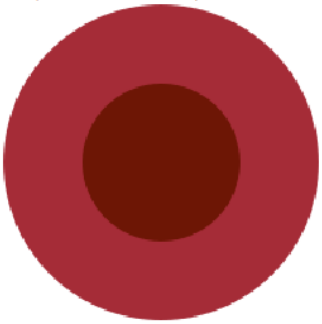
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( flip-for-difference 1 )
t
> ( flip-for-difference 1 )
h
> ( flip-for-difference 1 )
t
> ( flip-for-difference 1 )
h
> ( flip-for-difference 2 )
h t t t
> ( flip-for-difference 2 )
t t
> ( flip-for-difference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

Task 4: Laying Down Colorful Concentric Disks

CCR Demo:

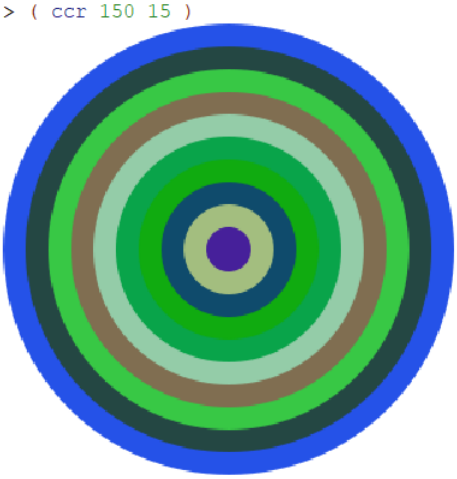
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
> ( ccr 100 50 )
```



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

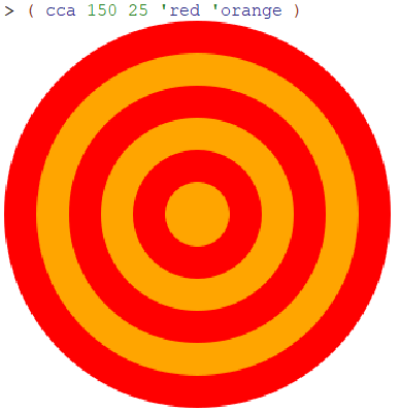
```
> ( ccr 150 15 )
```



CCA Demo:

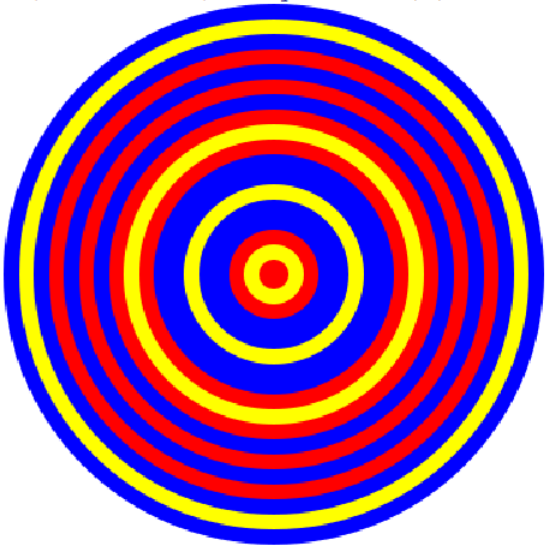
```
> ( cca 160 10 'black 'white )
```

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

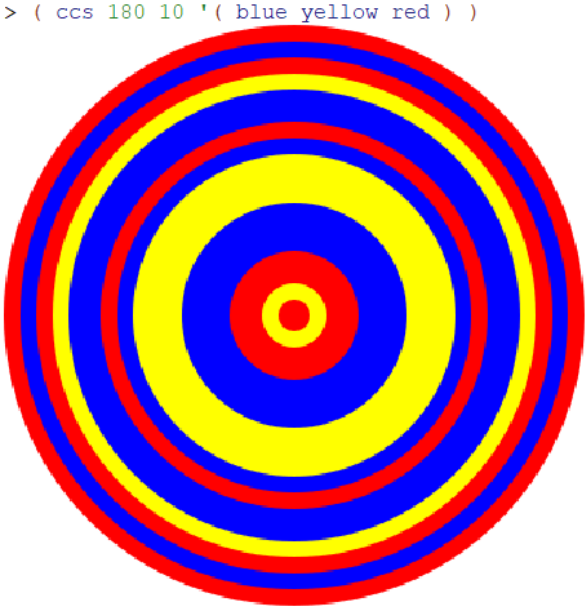


Demo 1:

```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> ( ccs 180 10 '( blue yellow red ) )
```

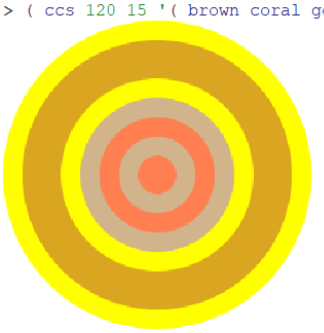


```
> ( ccs 180 10 '( blue yellow red ) )
```

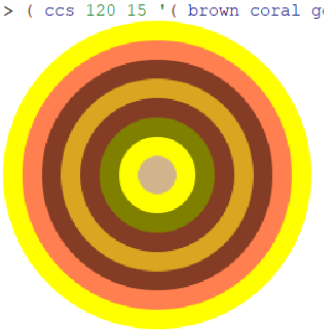


Demo 2:

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



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



```
>
```

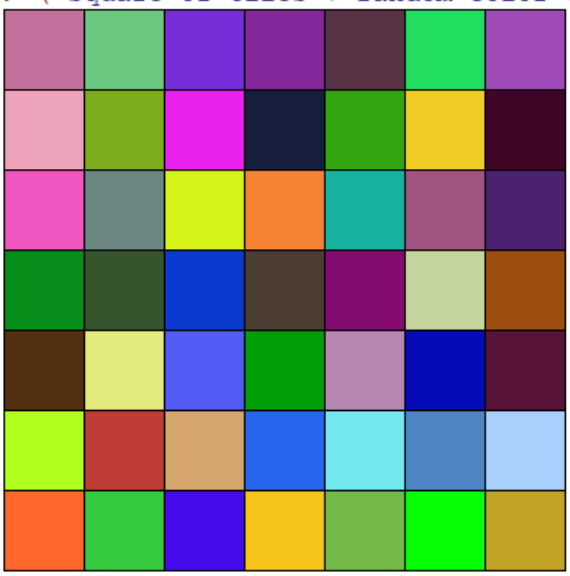
Code:

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

( define ( ccr radius diff )
   ( cond
      ( ( = radius 0 ) empty-image )
      ( ( > radius 0 )
        ( overlay ( ccr ( - radius diff ) diff ) ( circle radius 'solid ( random-color ) ) )
        )
      )
   )

( define ( cca radius diff c1 c2 )
   ( helper radius diff c1 c2 1 )
   )

( define ( helper radius diff c1 c2 numOfColor )
   ( cond ( ( > radius 0 )
            ( cond ( ( = numOfColor 1 )
                     ( overlay ( helper ( - radius diff ) diff c1 c2 2 )
                               ( circle radius 'solid c1 ) ) )
                   ( ( = numOfColor 2 )
                     ( overlay ( helper ( - radius diff ) diff c1 c2 1 )
                               ( circle radius 'solid c2 ) ) ) ) )
          ( ( = radius 0 ) empty-image ) ) )

( define ( ccs radius diff color )
   ( define numForColors ( length color ) )
   ( helper-2 radius diff color numForColors )
   )

( define ( helper-2 radius diff colors numForColors )
   ( cond ( ( > radius 0 )
            ( define ( numForColor ) ( random numForColors ) )
            ( define color ( list-ref colors ( numForColor ) ) )
            ( overlay ( helper-2 ( - radius diff ) diff colors numForColors )
                      ( circle radius 'solid color ) ) )
          ( ( = radius 0 ) empty-image ) ) )
```

Task 5: Variations on Hirst Dots

Random Colored Tile Demo:

```
> ( square-of-tiles 7 random-color-tile )
```



```
>
```

Hirst Dots Demo:

```
> ( square-of-tiles 5 dot-tile )
```

```
>
```

CCS Dots Demo:

```
> ( square-of-tiles 7 ccs-tile )
```

```
>
```

Nested Diamonds Demo:

```
> ( square-of-tiles 6 diamond-tile )
```

```
>
```

Unruly Squares Demo:

```
> ( square-of-tiles 6 wild-square-tile )
```

```
>
```

Code:

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

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

( define ( random-color )
   ( define ( rgb ) ( random 0 256 ) )
   ( color ( rgb ) ( rgb ) ( rgb ) )
   )

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

```
( define ( ccs-tile )
   ( define color ( random-colors 3 ) )
   ( overlay
     ( ccs 35 5 color )
     ( square 100 "solid" "white" ) )
   )

( define ( random-colors n )
   ( cond
      ( ( > n 0 )
        ( cons ( random-color ) ( random-colors ( - n 1 ) ) )
       )
      ( ( = n 0 ) empty ) ) )

( define ( diamond-tile )
   ( define diamond ( random-color ) )
   ( overlay ( rotate 45 ( square 30 "solid" "white" ) )
             ( rotate 45 ( square 40 "solid" diamond ) )
             ( rotate 45 ( square 50 "solid" "white" ) )
             ( rotate 45 ( square 60 "solid" diamond ) )
             ( square 100 "solid" "white" ) )
   )

( define ( wild-square-tile )
   ( define squares ( random-color ) )
   ( define angle ( random 0 90 ) )
   ( overlay
     ( rotate angle ( square 30 "solid" "white" ) )
     ( rotate angle ( square 40 "solid" squares ) )
     ( rotate angle ( square 50 "solid" "white" ) )
     ( rotate angle ( square 60 "solid" squares ) )
     ( square 100 "solid" "white" ) )
   )

( define ( ccs radius diff colors )
   ( define numOfColors ( length colors ) )
   ( helper radius diff colors numOfColors )
   )

( define ( helper radius diff colors numOfColors )
   ( cond
      ( ( > radius 0 )
           ( define ( colorNum ) ( random numOfColors ) )
           ( define color ( list-ref colors ( colorNum ) ) )
           ( overlay ( helper ( - radius diff ) diff colors numOfColors ) ( circle radius 'solid color ) )
           )
         ( ( = radius 0 ) empty-image ) ) )
```