Damian Randall

Racket Assignment #3: Recursion in Racket

Abstract: This assignment serves to further our engagement with racket along with our previous understanding of recursion from past coding experiences.

Task 1: Counting down, Counting up

Code:

```
(define (count-down x)

  (cond

    ((= x 1)
     (display x)
    )

    ((> x 0)
     (display x)
     (display "\n")
     (count-down (- x 1))
    )
  )
)

( define ( count-up n )
   ( cond
      ( ( = n 1 )

        ( display n )
        ( display "\n" )
        )
```

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

Demo:

```
> (count-down 5)
5
4
3
2
1
> (count-down 10)
10
9
8
7
6
5
4
3
2
1
```

Damian Randall

```
> (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 (row-of-stars n)
  (cond
    ((= n 0)
     (display "\n")
     )

    ((> n 0)
     (display "*")
     (row-of-stars (- n 1))
     )
    )
  )

(define (triangle-of-stars n)
  (cond
    ((= n 0)
     (display "")
     )

    ((> n 0)
     (triangle-of-stars (- n 1))
     (row-of-stars n)
     )
    )
  )
```

Damian Randall

Demo:

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

Task 3: Flipping a Coin

```
( define ( flip-coin )
  ( define outcome ( random 2 ) )
  ( cond
    ( ( = outcome 0 ) 't )
    ( ( = outcome 1 ) 'h )
    )
)
( define ( difference-check dif dif2 n )
  ( define m ( flip-coin ) )
  ( cond
    ( ( eq? ( abs ( - dif dif2 ) ) n )
      ( display "" )
      )
    ( ( eq? 'h m )
      ( difference-check ( + dif 1 ) dif2 n )
      ( display m )
      )
    ( ( eq? 't m )
      ( difference-check dif ( + dif2 1 ) n )
      ( display m )
      )
    )
  )

( define ( flip-for-difference n )
  ( difference-check 0 0 n )
)
```

Damian Randall

Demo:

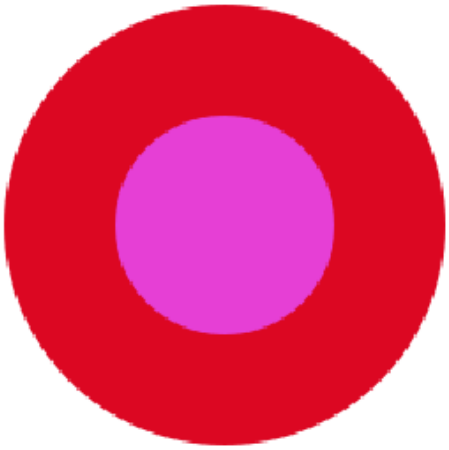
```
>  ( flip-for-difference 1 )
h
>  ( flip-for-difference 1 )
h
>  ( flip-for-difference 1 )
t
>  ( flip-for-difference 1 )
t
> ( flip-for-difference 2 )
tthtth
> ( flip-for-difference 2 )
hhth
> ( flip-for-difference 2 )
tt
> ( flip-for-difference 2 )
ttthht
> ( flip-for-difference 2 )
hhthth
> ( flip-for-difference 2 )
hh
> ( flip-for-difference 3 )
ttt
> ( flip-for-difference 3 )
hhthh
> ( flip-for-difference 3 )
hhhht
> ( flip-for-difference 3 )
ttthhhhtthhtthttthtthhhhhtthttthtthhthtthhtttthhtthtthhhhhttth
> ( flip-for-difference 3 )
ttht
> ( flip-for-difference 3 )
hhhht
> ( flip-for-difference 4 )
hhhhthhhhtthttttthhhthhhttttthhhhhtththt
> ( flip-for-difference 4 )
hhhtthhhthhhhhtthttttthhtthhthtthhhhhtththt
> ( flip-for-difference 4 )
ttttthhthhthhttthtthhtthttthth
> ( flip-for-difference 4 )
hhhtthhhht
> ( flip-for-difference 4 )
ttttth
> ( flip-for-difference 4 )
tttt
> ( flip-for-difference 4 )
hhhthhthhthhhhthht
> ( flip-for-difference 4 )
tttt
>
```

Damian Randall

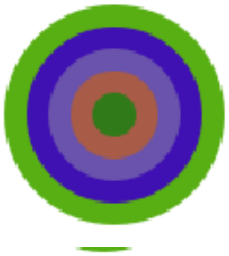
Task 4: Laying Down Colorful Concentric Disks

CCR Demo:

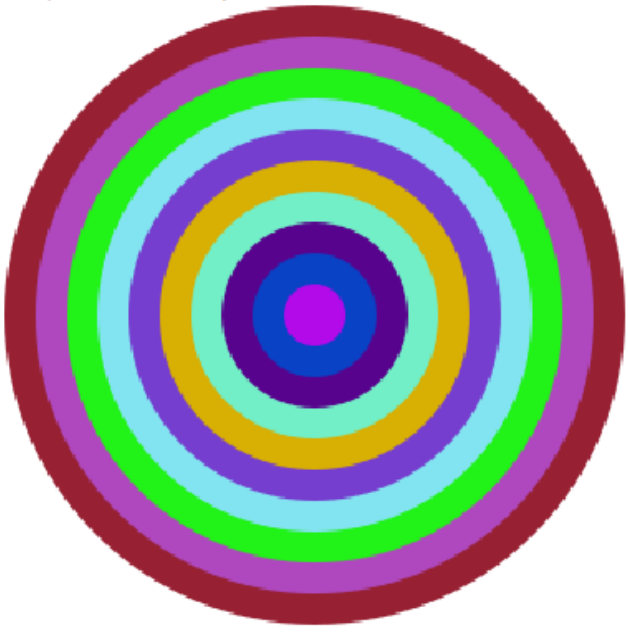
```
> (ccr 100 50)
```



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



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



```
>
```

Damian Randall

CCA Demo:

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

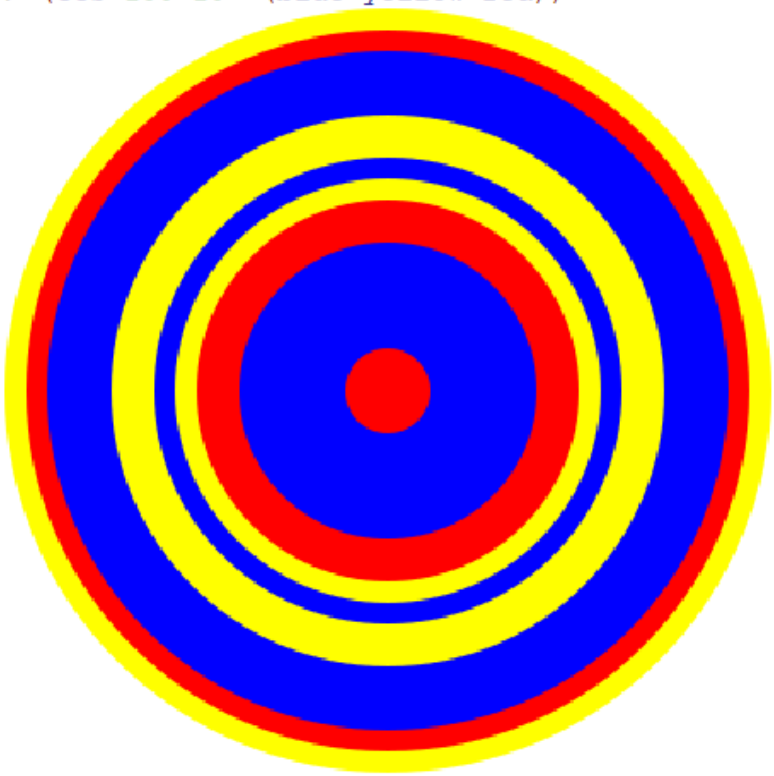
```
>
```

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

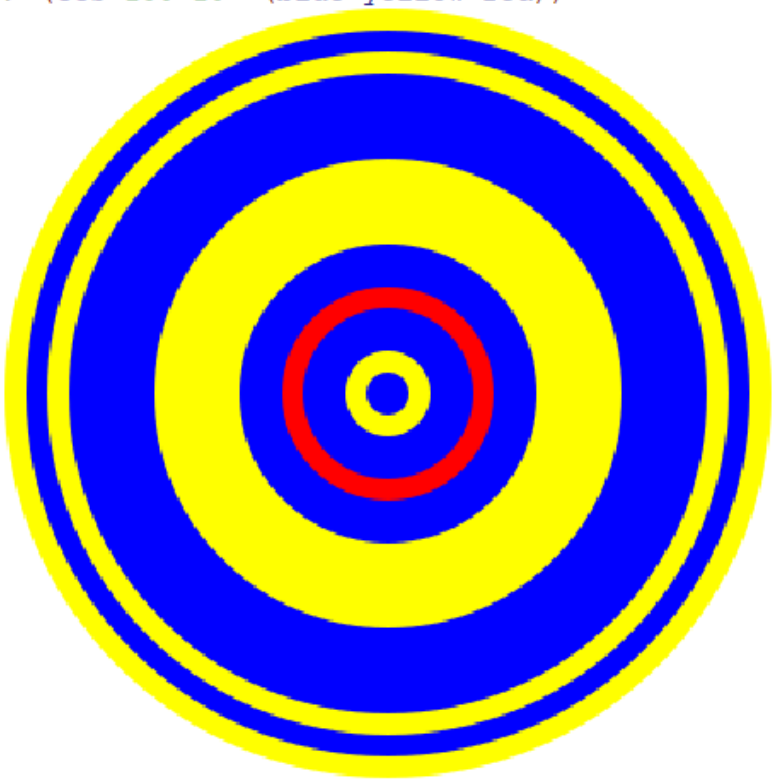
Damian Randall

CCS Demo:

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

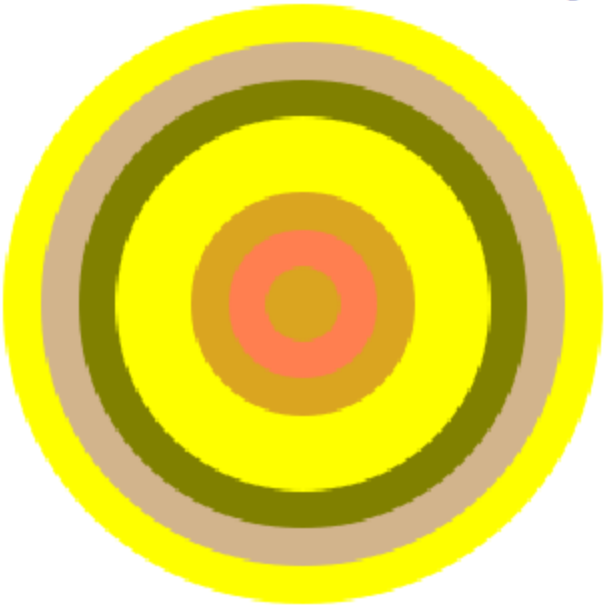


```
> (ccs 180 10 '(blue yellow red))
```



Damian Randall

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



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

Code:

```
(define (ccr currentRadius diff)
  (define (rc) (color(rgb)(rgb)(rgb)))
  (define (rgb) (random 0 256))

  (cond
    ((<= currentRadius 0)
     empty-image
     )

    ((> currentRadius 0)
     (define currentCircle (circle currentRadius "solid" (rc)))
     (overlay (ccr (- currentRadius diff) diff) currentCircle)
     )
    )
  )

(define (cca currentRadius diff color1 color2)

  (cond
    ((<= currentRadius 0)
     empty-image
     )

    ((> currentRadius 0)
     (define currentCircle (circle currentRadius "solid" color1))
     (overlay (cca (- currentRadius diff) diff color2 color1) currentCircle)
     )
    )
  )

(define (ccs currentRadius diff colorList)
  (define currentColor(list-ref colorList (random 0 (length colorList))))

  (cond
```

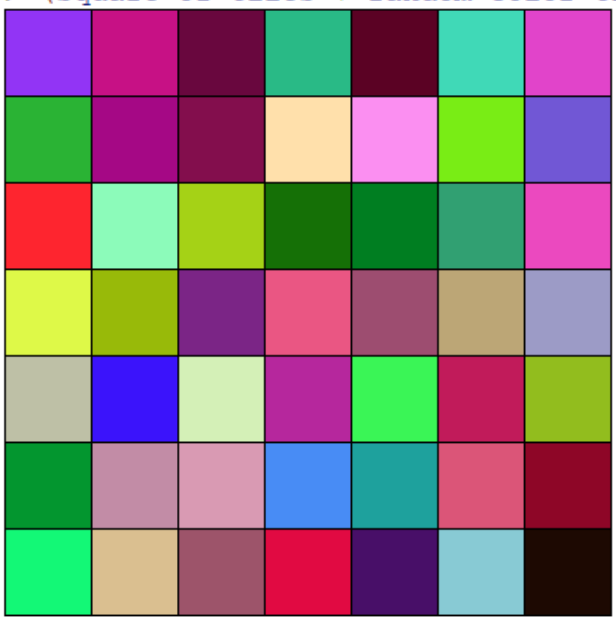
```
    ((<= currentRadius 0)
     empty-image
     )

    ((> currentRadius 0)
     (define currentCircle (circle currentRadius "solid" currentColor))
     (overlay (ccs (- currentRadius diff) diff colorList) currentCircle)
     )
    )
  )
```

Task 5: Variations on Hirst Dots

Demo 1:

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



Demo 2:

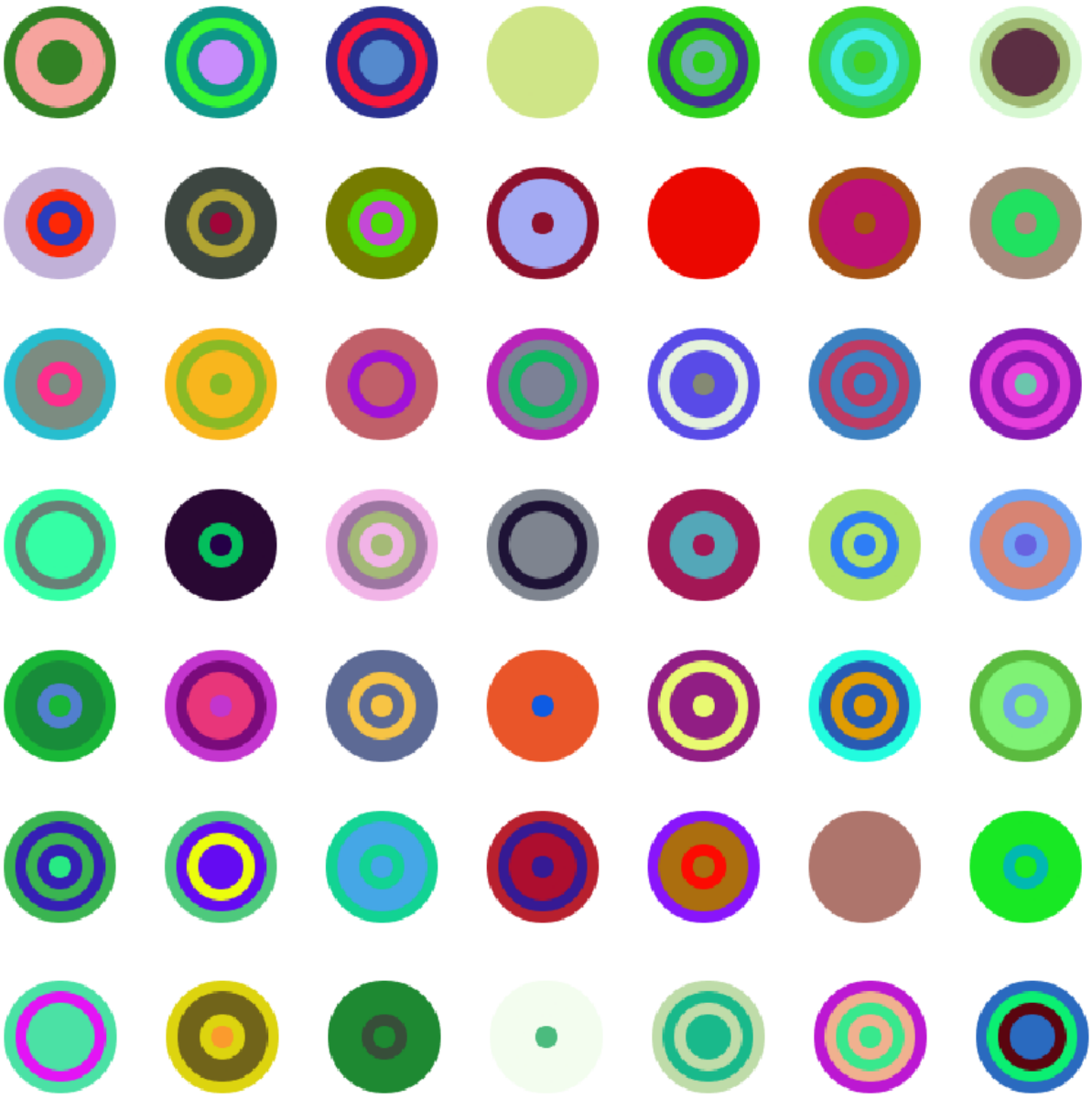
Damian Randall

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

Demo 3:

Damian Randall

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



Demo3:

Damian Randall

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

Demo 4:

Damian Randall

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

Code:

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

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

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

(define (dot-tile)
  (define blankTile (square 100 "solid" "white"))
  (define dot (circle (/ 70 2) "solid" (random-color)))
  (overlay dot blankTile)
  )

(define (ccs-tile)
  (define color1(random-color))
  (define color2(random-color))
  (define color3(random-color))
  (define colorList(list color1 color2 color3))
  (define blankTile (square 100 "solid" "white"))
  (define currentTile(ccs 35 7 colorList))
  (overlay currentTile blankTile)
  )

(define (diamond-tile)
  (define currentColor(random-color))
  (define blankTile (square 100 "solid" "white"))
  (define outerSquare (square 65 "solid" currentColor))
```

```
  (define outerBlank (square 45 "solid" "white"))
  (define innerSquare(square 25 "solid" currentColor))
  (define innerBlank (square 10 "solid" "white"))
  (define outerPicture(overlay outerBlank outerSquare))
  (define innerPicture(overlay innerBlank innerSquare))
  (define totalDiamond(overlay innerPicture outerPicture))
  (define totalDiamondRotated(rotate 45 totalDiamond))
  (overlay totalDiamondRotated blankTile)
  )

(define (wild-diamond-tile)
  (define randomRotate(random 0 46))
  (define currentColor(random-color))
  (define blankTile (square 100 "solid" "white"))
  (define outerSquare (square 65 "solid" currentColor))
  (define outerBlank (square 45 "solid" "white"))
  (define innerSquare(square 25 "solid" currentColor))
  (define innerBlank (square 10 "solid" "white"))
  (define outerPicture(overlay outerBlank outerSquare))
  (define innerPicture(overlay innerBlank innerSquare))
  (define totalDiamond(overlay innerPicture outerPicture))
  (define totalDiamondRotated(rotate randomRotate totalDiamond))
  (overlay totalDiamondRotated blankTile)
  )



(define (row-of-tiles x tile)
  (cond
    ((= x 0)
     empty-image
     )
    ((> x 0)
     (beside (row-of-tiles (- x 1) tile) (tile))
     )
```

```
    )
  )

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

(define (square-of-tiles x tile)
  (rectangle-of-tiles x x tile)
  )
```