Racket Assignment #2: Interactions, Definitions, Applications

Abstract

This assignment will help me hone my foundational abilities in the Racket programming language. I will be performing a range of interactions, coming up with function definitions, and advance my ability to solve problems. In addition to these, it will also help me in practicing how to reuse code efficiently.

Task 1: Interactions – Scrap of Tin

Arithmetic Expressions

Solving a Simple Problem (Area of Scrap)

```
Welcome to DrRacket, version 8.7 [cs].
Language: Determine language from source; memory limit: 128 MB.
> pi
3.141592653589793
> side
side: undefined;
 cannot reference an identifier before its definition
> ( define side 100 )
> side
100
> ( define square-area ( * side side ) )
> square-area
10000
> ( define radius ( / side 2 ) )
> radius
50
> ( define circle-area ( * pi radius radius ) )
> circle-area
7853.981633974483
> ( define scrap-area ( - square-area circle-area ) )
> scrap-area
2146.018366025517
>
```

Rendering an Image of the Problem Situation

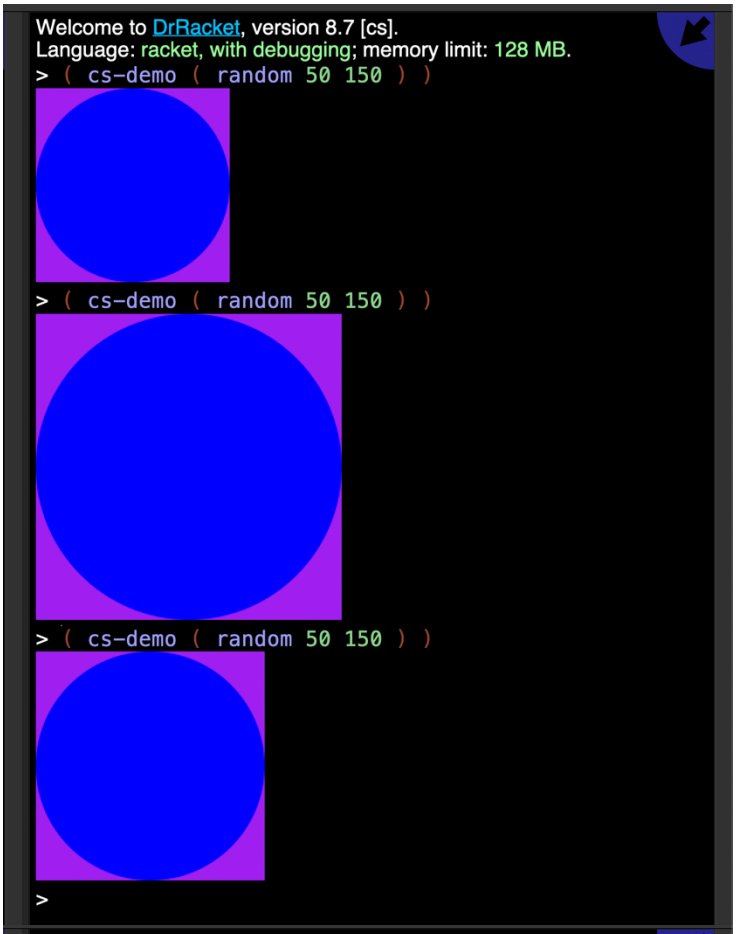
```
Welcome to DrRacket, version 8.7 [cs].
Language: Determine language from source; memory limit: 128 MB.
> ( require 2htdp/image )
> ( define side 100 )
> ( define the-square ( square side "solid" "silver" ) )
> the-square
```

```
> ( define radius ( / side 2 ) )
> ( define the-circle ( circle radius "solid" "white" ) )
> ( define the-image ( overlay the-circle the-square ) )
> the-image
```

```
>
```

Task 2: Definitions – Inscribing/Circumscribing Circles/Squares

Demos




Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (ic-demo (random 50 150))
> (ic-demo (random 50 150))
> (ic-demo (random 50 150))
>

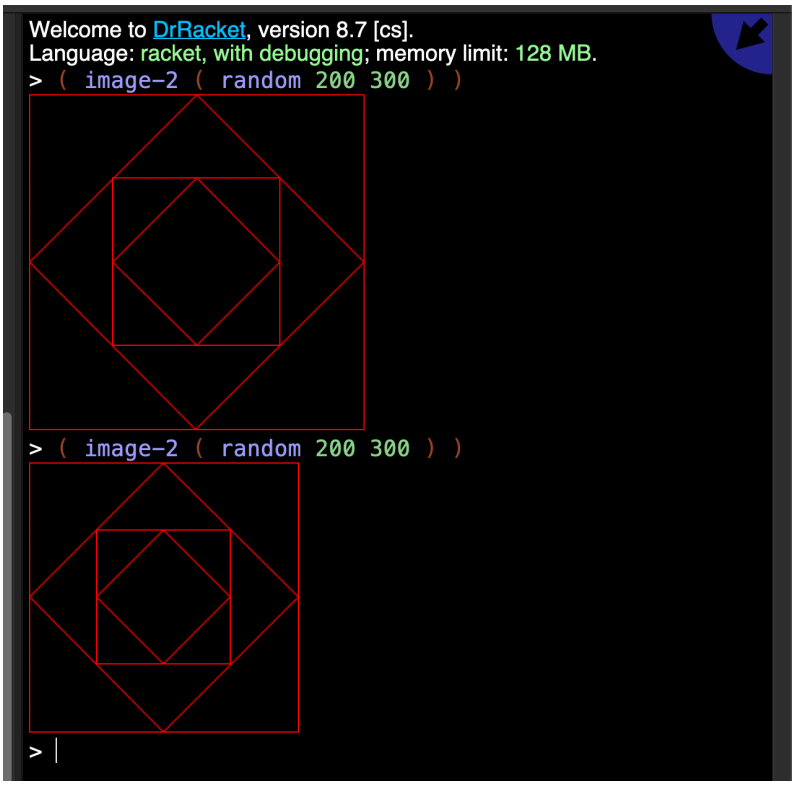

Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (is-demo (random 50 150))
> (is-demo (random 50 150))
> (is-demo (random 50 150))
>

The Code

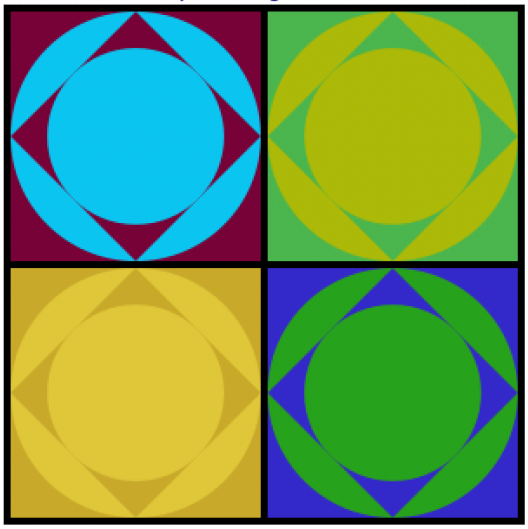
```
#lang racket
( require 2htdp/image )

( define ( cs radius )
   ( * radius 2 )
)

( define ( cc side-length )
   ( / ( * side-length ( sqrt 2 ) ) 2 )
)

( define ( ic side-length )
   ( / side-length 2 )
)

( define ( is radius )
   ( * radius ( sqrt 2 ) )
)

( define ( cs-demo radius )
   ( define the-square ( square (cs radius) "solid" "purple" ) )
   ( define the-circle ( circle radius "solid" "blue" ) )
   ( overlay the-circle the-square )
)

( define ( cc-demo side-length )
   ( define the-circle ( circle (cc side-length) "solid" "purple" ) )
   ( define the-square ( square side-length "solid" "blue" ) )
   ( overlay the-square the-circle )
)

( define ( ic-demo side-length )
   ( define the-circle ( circle (ic side-length) "solid" "purple" ) )
   ( define the-square ( square side-length "solid" "blue" ) )
   ( overlay the-circle the-square )
)

( define ( is-demo radius )
   ( define the-square ( square (is radius) "solid" "purple" ) )
   ( define the-circle ( circle radius "solid" "blue" ) )
   ( overlay the-square the-circle )
)
```

Task 3: Inscribing/Circumscribing Images

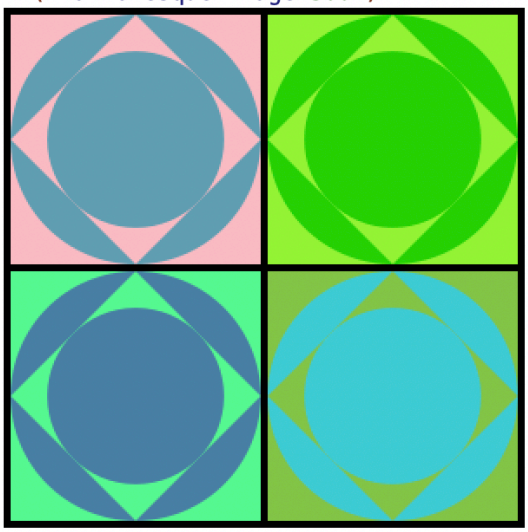
Demos



```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( warholesque-image 300 )
```



```
> ( warholesque-image 300 )
```



```
>
```

The Code

```
( define ( image-1 side-length )
   ( overlay ( rotate 45 ( ic-demo ( is ( ic side-length ) ) ) ) ( ic-
demo side-length ) )
)

( define ( image-2 side-length )
   ( define squareA ( square side-length "outline" "red" ) )
   ( define B ( is ( ic side-length ) ) )
   ( define squareB ( rotate 45 ( square B "outline" "red" ) ) )
   ( define C ( is ( ic B ) ) )
   ( define squareC ( square C "outline" "red" ) )
   ( define D ( is ( ic C ) ) )
   ( define squareD ( rotate 45 ( square D "outline" "red" ) ) )
   ( overlay squareA squareB squareC squareD )
)


( define ( warholesque-image side-length )

   ( define ( BG side-length )
      ( square side-length "solid" "black")
   )

   ( define sideLength2
      ( - side-length 4 )
   )

   ( define pieceSide
      ( / sideLength2 2 )
   )

   ( define pieceSide2
      ( - pieceSide 4 )
   )

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

   ( define squareC1 ( rc ) )
   ( define circleC1 ( rc ) )
   ( define squareC2 ( rc ) )
   ( define circleC2 ( rc ) )
   ( define squareC3 ( rc ) )
   ( define circleC3 ( rc ) )
   ( define squareC4 ( rc ) )
   ( define circleC4 ( rc ) )


   ( define ( subPiece1 pieceSide )
```

```
      ( define the-circle ( circle (ic pieceSide) "solid" squareC1 ) )
      ( define the-square ( square pieceSide "solid" circleC1 ) )
      ( overlay the-circle the-square )
   )

   ( define ( piece1 pieceSide pieceSide2)
      ( define the-border ( square pieceSide "solid" "black" ) )
      ( overlay ( rotate 45 ( subPiece1 ( is ( ic pieceSide2 ) ) ) ) (
subPiece1 pieceSide2 ) the-border )
   )


   ( define ( subPiece2 pieceSide )
      ( define the-circle ( circle (ic pieceSide) "solid" squareC2 ) )
      ( define the-square ( square pieceSide "solid" circleC2 ) )
      ( overlay the-circle the-square )
   )

   ( define ( piece2 pieceSide pieceSide2)
      ( define the-border ( square pieceSide "solid" "black" ) )
      ( overlay ( rotate 45 ( subPiece2 ( is ( ic pieceSide2 ) ) ) ) (
subPiece2 pieceSide2 ) the-border )
   )


   ( define ( subPiece3 pieceSide )
      ( define the-circle ( circle (ic pieceSide) "solid" squareC3 ) )
      ( define the-square ( square pieceSide "solid" circleC3 ) )
      ( overlay the-circle the-square )
   )

   ( define ( piece3 pieceSide pieceSide2 )
      ( define the-border ( square pieceSide "solid" "black" ) )
      ( overlay ( rotate 45 ( subPiece3 ( is ( ic pieceSide2 ) ) ) ) (
subPiece3 pieceSide2 ) the-border )
   )

   `z
   ( define ( subPiece4 pieceSide )
      ( define the-circle ( circle (ic pieceSide) "solid" squareC4 ) )
      ( define the-square ( square pieceSide "solid" circleC4 ) )
      ( overlay the-circle the-square )
   )

   ( define ( piece4 pieceSide pieceSide2 )
      ( define the-border ( square pieceSide "solid" "black" ) )
      ( overlay ( rotate 45 ( subPiece4 ( is ( ic pieceSide2 ) ) ) ) (
subPiece4 pieceSide2 ) the-border )
   )

   ( overlay ( above ( beside ( piece1 pieceSide pieceSide2 ) ( piece2
pieceSide pieceSide2 ) )
```

```
            ( beside ( piece3 pieceSide pieceSide2 ) ( piece4 pieceSide
pieceSide2 ) )
            ) (BG side-length )
              )
)
```

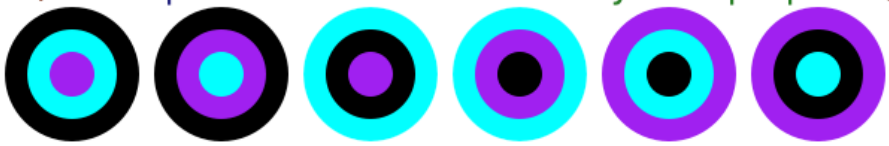
Task 4: Permutations of Randomly Colored Stacked Dots

Demos

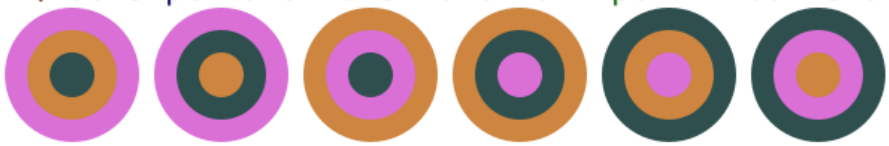
```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( tile "khaki" "steelblue" "plum" "seagreen" )
```

```
> ( tile "navy" "teal" "firebrick" "salmon" )
```

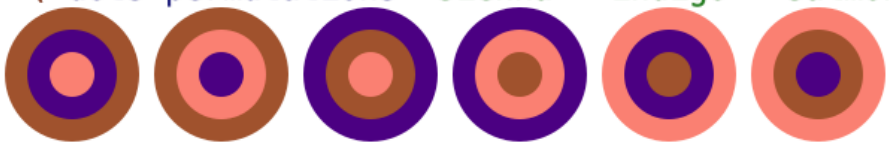
```
> ( dots-permutations "black" "cyan" "purple" )
```



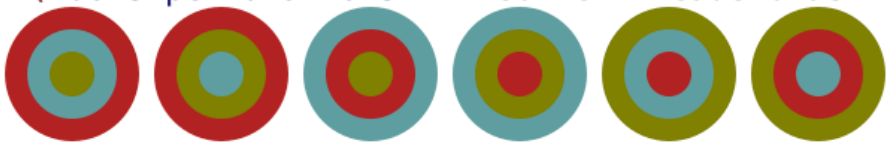
```
> ( dots-permutations "orchid" "peru" "darkslategray" )
```



```
> ( dots-permutations "sienna" "indigo" "salmon" )
```



```
> ( dots-permutations "firebrick" "cadetblue" "olive" )
```



```
>
```

The Code

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

( define side 100 )

( define d1 90 )
( define d2 60 )
( define d3 30 )

( define r1 ( / d1 2 ) )
( define r2 ( / d2 2 ) )
( define r3 ( / d3 2 ) )

( define ( tile color1 color2 color3 color4 )
   ( define squ1 ( square side "solid" color1 ) )
   ( define cir1 ( circle r1 "solid" color2 ) )
   ( define cir2 ( circle r2 "solid" color3 ) )
   ( define cir3 ( circle r3 "solid" color4 ) )
   ( overlay cir3 cir2 cir1 squ1 )
)



( define ( dots-permutations color1 color2 color3 )
   ( define ( target1 color1 color2 color3 )
      ( define cir1 ( circle r1 "solid" color1 ) )
      ( define cir2 ( circle r2 "solid" color2 ) )
      ( define cir3 ( circle r3 "solid" color3 ) )
      ( overlay cir3 cir2 cir1 )
   )
   ( define ( target2 color1 color2 color3 )
      ( define cir1 ( circle r1 "solid" color1 ) )
      ( define cir2 ( circle r2 "solid" color3 ) )
      ( define cir3 ( circle r3 "solid" color2 ) )
      ( overlay cir3 cir2 cir1 )
   )
   ( define ( target3 color1 color2 color3 )
      ( define cir1 ( circle r1 "solid" color2 ) )
      ( define cir2 ( circle r2 "solid" color1 ) )
      ( define cir3 ( circle r3 "solid" color3 ) )
      ( overlay cir3 cir2 cir1 )
   )
   ( define ( target4 color1 color2 color3 )
      ( define cir1 ( circle r1 "solid" color2 ) )
      ( define cir2 ( circle r2 "solid" color3 ) )
      ( define cir3 ( circle r3 "solid" color1 ) )
      ( overlay cir3 cir2 cir1 )
   )
   ( define ( target5 color1 color2 color3 )
```

```
      ( define cir1 ( circle r1 "solid" color3 ) )
      ( define cir2 ( circle r2 "solid" color2 ) )
      ( define cir3 ( circle r3 "solid" color1 ) )
      ( overlay cir3 cir2 cir1 )
   )
   ( define ( target6 color1 color2 color3 )
      ( define cir1 ( circle r1 "solid" color3 ) )
      ( define cir2 ( circle r2 "solid" color1 ) )
      ( define cir3 ( circle r3 "solid" color2 ) )
      ( overlay cir3 cir2 cir1 )
   )

   ( define gap ( square 10 "solid" "white" ) )

   ( beside gap ( target1 color1 color2 color3 )
            gap ( target2 color1 color2 color3 )
            gap ( target3 color1 color2 color3 )
            gap ( target4 color1 color2 color3 )
            gap ( target5 color1 color2 color3 )
            gap ( target6 color1 color2 color3 ) )
)
```