Racket Assignment #2: Interactions, Definitions, Applications

Learning Abstract

In this assignment, we learned about the basic idea to generate code for computational problem-solving and draw some imaginative construction like tile and dots-permutations. It helps to create some of the basic functions like gaps, squares, rectangles, etc.

Task 1: Interactions - Scrap of Tin

Arithmetic Expressions

```
> 5
5
> 5.3
5.3
> ( * 3
10)
30
> ( + (*
3 10) 4)
34
> (* 9 9
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)
121576654590569288 01
```

Solve a Simple Problem (Area of Scrap)

```
> 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

```
>(require 2htdp/image)

>(define side 100)

>(define square-area(square side "solid" "silver"))

>square-area
```

>(define radius(/ side 2))

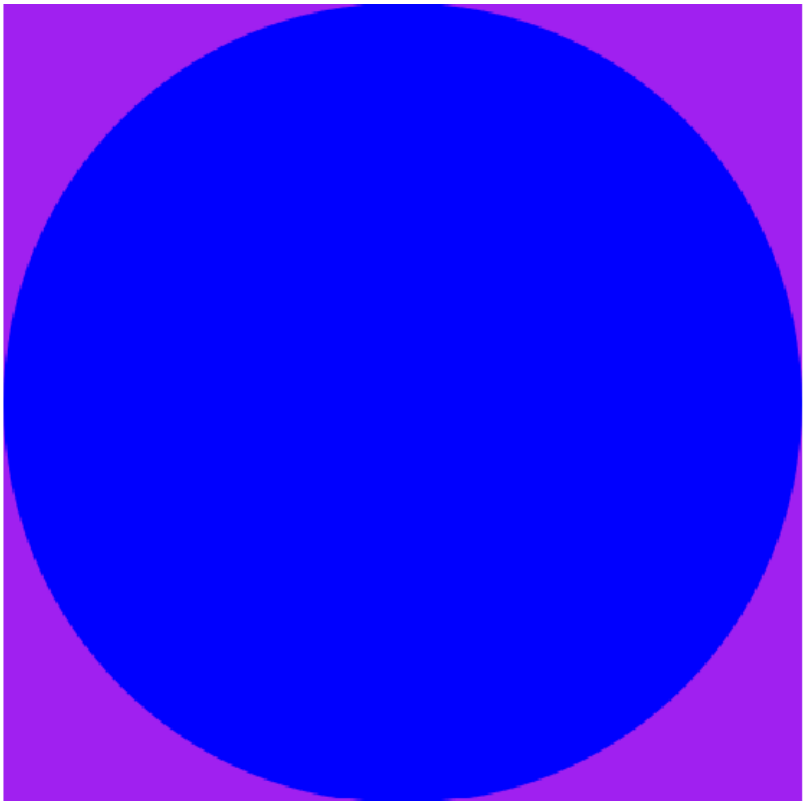
>(define circle-area(circle radius "solid" "white"))

>(define the-image(overlay circle-area square-area))

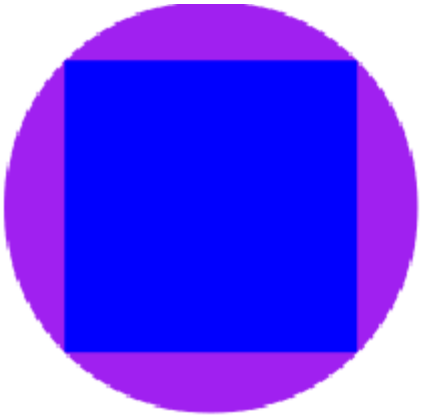
>the-image

Task 2: Definitions - Inscribing/Circumscribing Circles/Squares

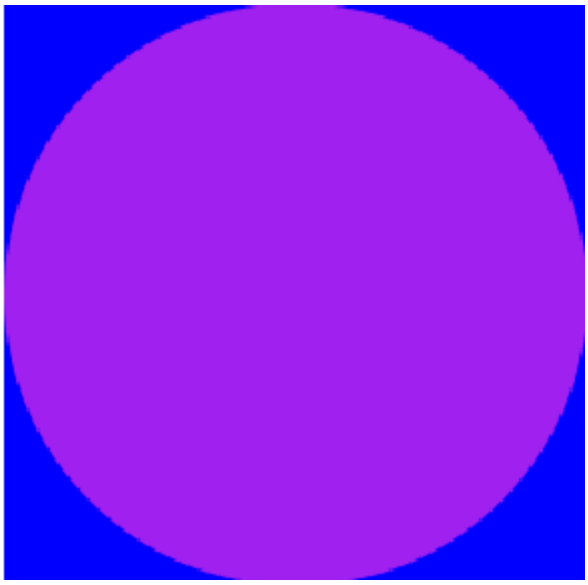
Cs-demo



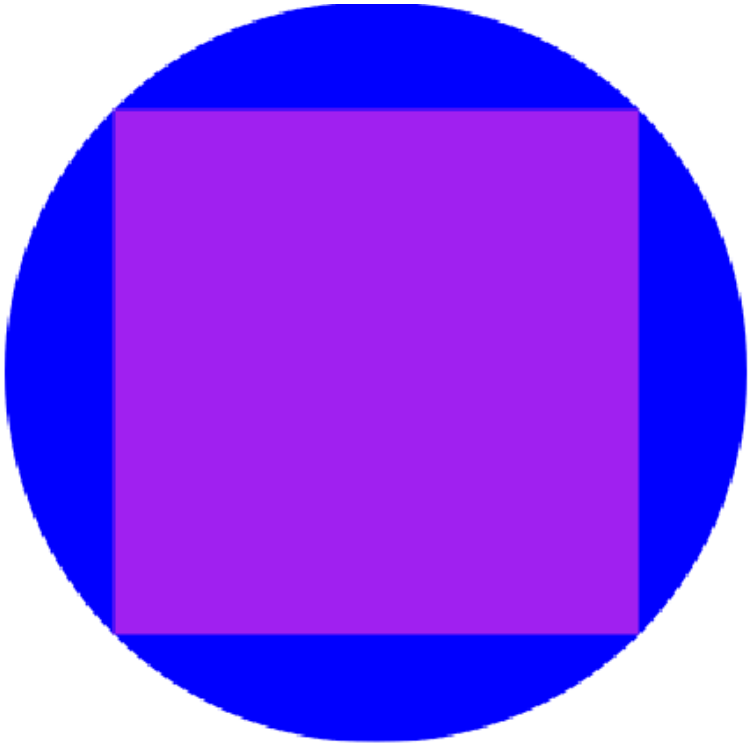
Cc-demo



Ic-demo



is-demo



The Code

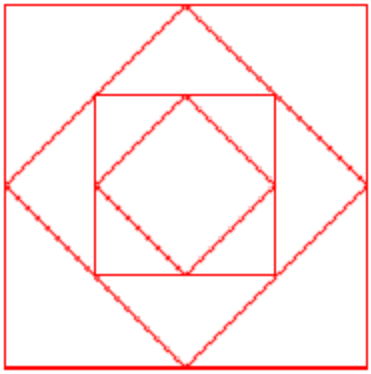
```
(require 2htdp/image)
(define (cs radius)
  (* radius 2)
)
(define (cc side)
  (/ (* (sqrt 2) side ) 2)
  )
(define (ic side)
  (/ side 2)
  )
(define (is radius)
  (* (sqrt 2) radius)
  )
(define (cs-demo r)
  (define s(* r 2))
 (define cir(circle r "solid" "blue"))
  (define squ(square s "solid" "purple"))
  (define ms(overlay cir squ  ) )
  ms
  )
```

```
(define (cc-demo s)
  (define r(/ s (sqrt 2)))
  (define cir(circle r "solid" "purple"))
  (define squ(square s "solid" "blue"))
  (define ms(overlay squ cir  ) )
  ms
  )

(define (ic-demo s)
  (define r(/ s 2))
   (define cir(circle r "solid" "purple"))
  (define squ(square s "solid" "blue"))
  (define ms(overlay cir squ) )
  ms
  )

(define (is-demo r)
  (define s(* (sqrt 2) r))
  (define cir(circle r "solid" "blue"))
  (define squ(square s "solid" "purple"))
  (define ms(overlay squ cir) )
  ms
  )
```

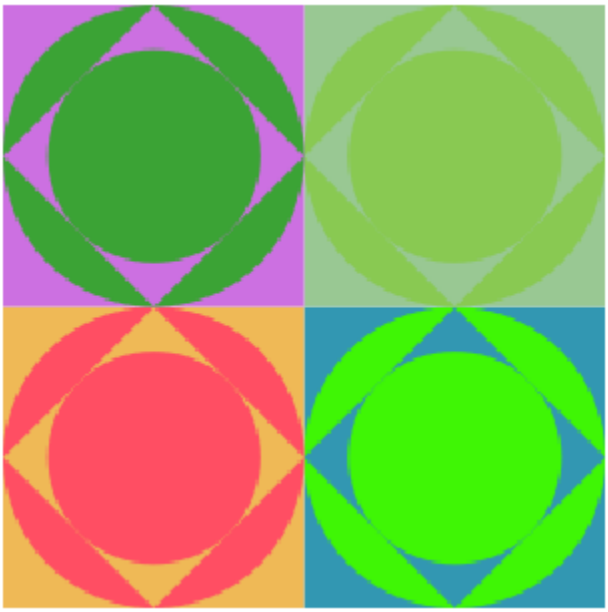
Task 3: Inscribing/Circumscribing Images

Image 1 Demo

Image 2 Demo



Warholesque Image



The Code

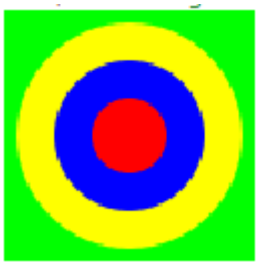
```
(require 2htdp/image)
(define (image-1 s)
  (define square1(square s "solid" "purple"))
  (define r1(/ s 2))
  (define circle1(circle r1 "solid" "cyan"))
  (define s2(* (sqrt 2) r1))
  (define square2(square s2 "solid" "purple"))
  (define rot(rotate 45 square2))
```

```
  (define r2(/ s2 2))
  (define circle2(circle r2 "solid" "cyan"))
  (define ms(overlay circle2 rot circle1 square1))
  ms
  )

(define (image-2 s1)
  (define square1(square s1 "outline" "red"))
  (define s2(* (sqrt 2) (/ s1 2)))
  (define sq2(square s2 "outline" "red"))
  (define square2(rotate 45 sq2))
  (define s3(* (sqrt 2) (/ s2 2)))
  (define square3(square s3 "outline" "red"))
  (define s4(* (sqrt 2) (/ s3 2)))
  (define sq4(square s4 "outline" "red"))
  (define square4(rotate 45 sq4))
  (define ms(overlay square4 square3 square2 square1))
  ms
  )

(define (Warholesque-image side)
  (define s1(/ side 2))
  (above
   (beside(image-3 s1)(image-3 s1))
   (beside(image-3 s1)(image-3 s1))
   ))
```

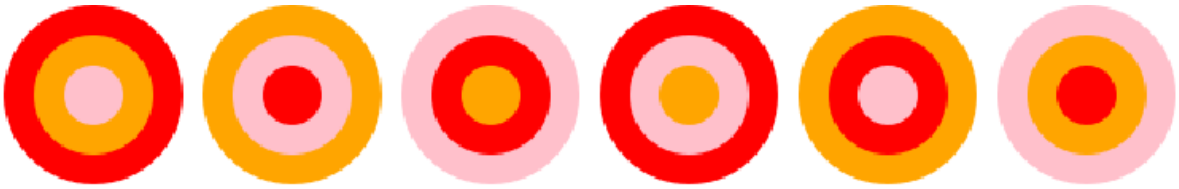
Task 4: Permutations of Randomly Colored Stacked Dots

.Demo

This is the demo for tile function



This is the demo for dot-permutations function



Code

```
(require 2htdp/image)
(define (gap)
  (define r2(rectangle 10 90 "solid" "white"))
  r2
  )

 (define(dots c1 c2 c3)
  (define diameter 90)
  (define r1(/ diameter 2))
  (define circle1(circle r1 "solid" c1))
  (define r2(- r1 15))
   (define circle2(circle r2 "solid" c2))
   (define r3(- r2 15))
   (define circle3(circle r3 "solid" c3))
   (define ms(overlay circle3 circle2 circle1))
   ms
   )

(define (dots-permutations c1 c2 c3)
  (beside
   (dots c1 c2 c3)
   (gap)
   (dots c2 c3 c1)
   (gap)
   (dots c3 c1 c2)
   (gap)
   (dots c1 c3 c2)
   (gap)
   (dots c2 c1 c3)
   (gap)
   (dots c3 c2 c1)
   ))
```