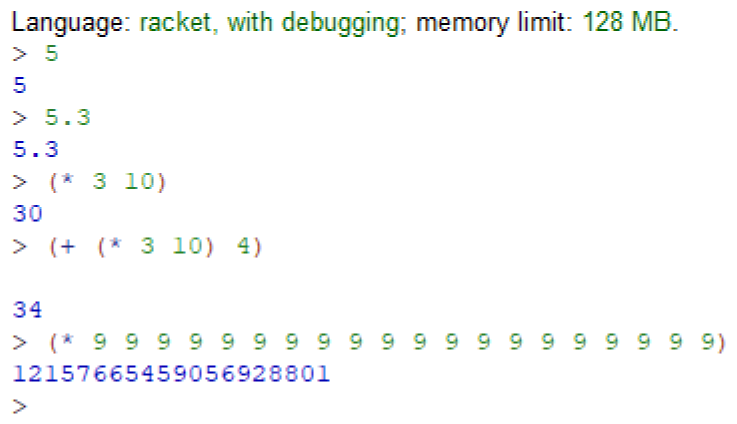
Title: Racket Assignment #2: Interactions, Definitions, Applications

Abstract:

In this assignment I wrote my first Racket programs for this semester.

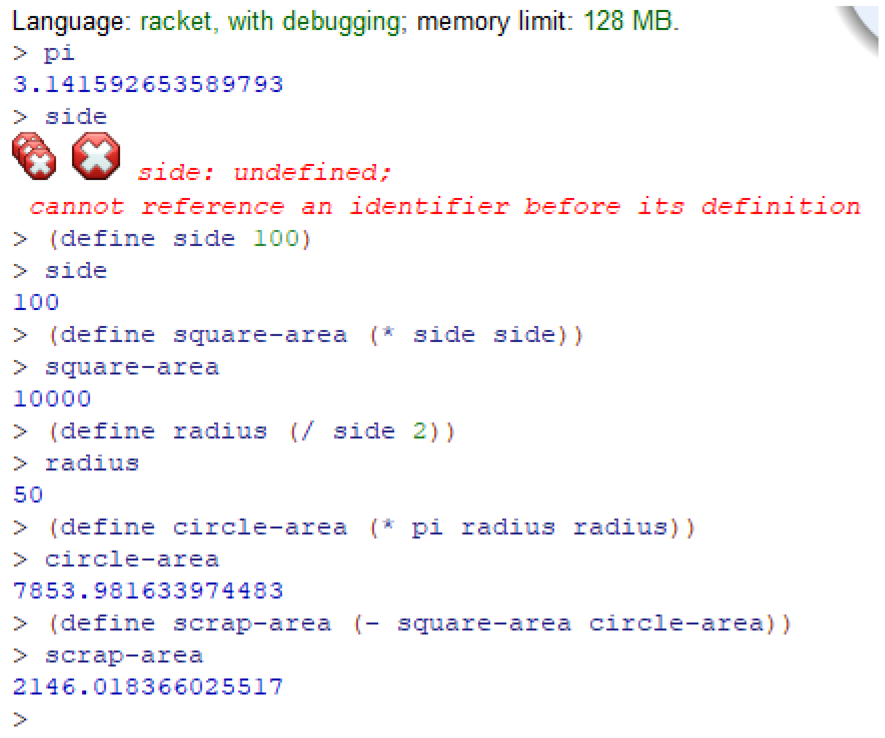
Task 1:

1:



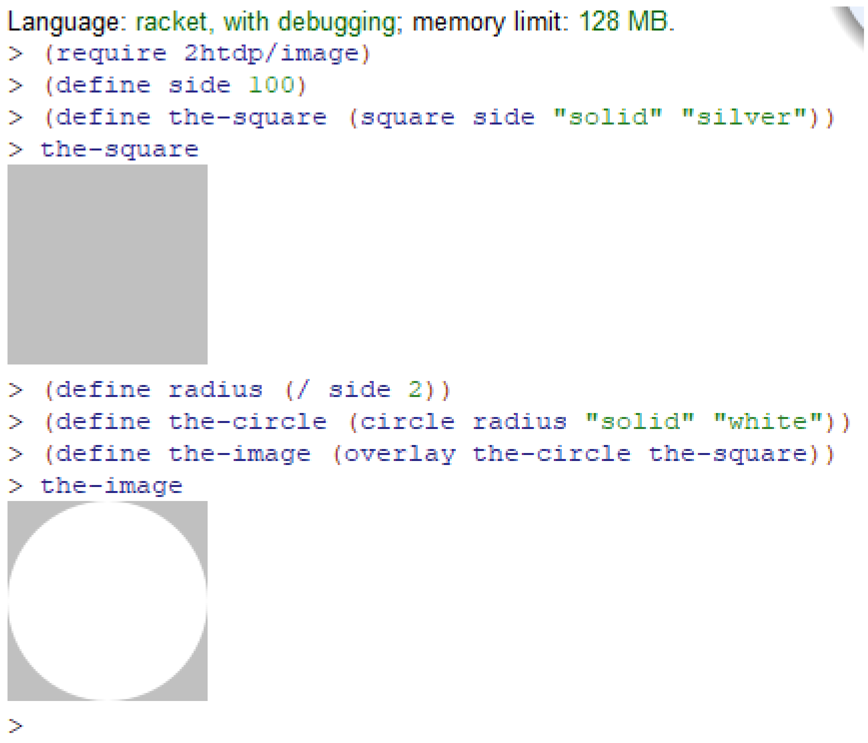
```
Language: racket, with debugging; memory limit: 128 MB.
> 5
5
> 5.3
5.3
> (* 3 10)
30
> (+ (* 3 10) 4)

34
> (* 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)
12157665459056928801
>
```

2:



```
Language: racket, with debugging; 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
>
```

3



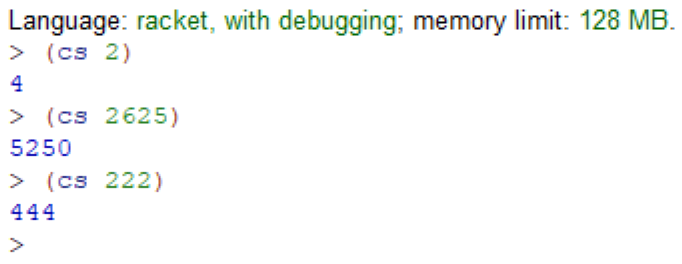
```
Language: racket, with debugging; memory limit: 128 MB.
> (require 2htdp/image)
> (define side 100)
> (define the-square (square side "solid" "silver"))
> the-square
```

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

```
>
```

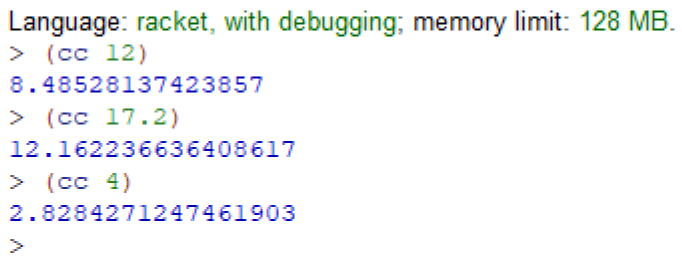
Task 2:

2:



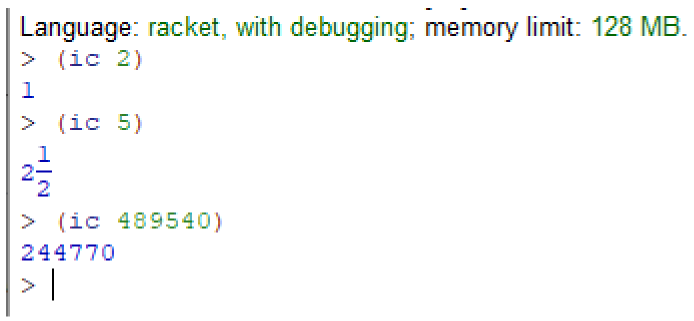
```
Language: racket, with debugging; memory limit: 128 MB.
> (cs 2)
4
> (cs 2625)
5250
> (cs 222)
444
>
```

3:



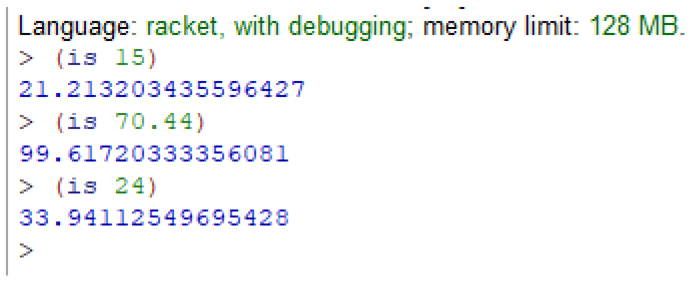
```
Language: racket, with debugging; memory limit: 128 MB.
> (cc 12)
8.48528137423857
> (cc 17.2)
12.162236636408617
> (cc 4)
2.8284271247461903
>
```

4:



```
Language: racket, with debugging; memory limit: 128 MB.
> (ic 2)
1
> (ic 5)
2 1/2
> (ic 489540)
244770
> 
```

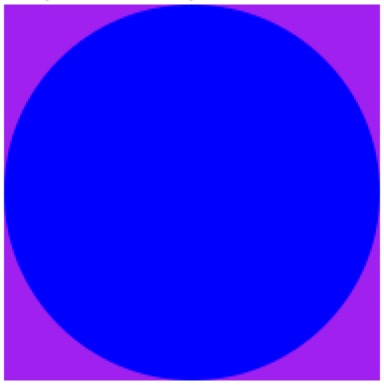
5:



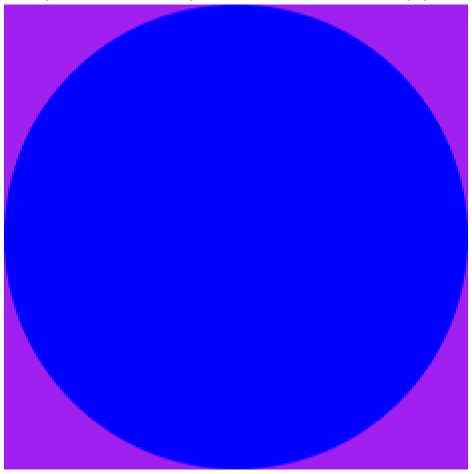
```
Language: racket, with debugging; memory limit: 128 MB.
> (is 15)
21.213203435596427
> (is 70.44)
99.61720333356081
> (is 24)
33.94112549695428
>
```

6:

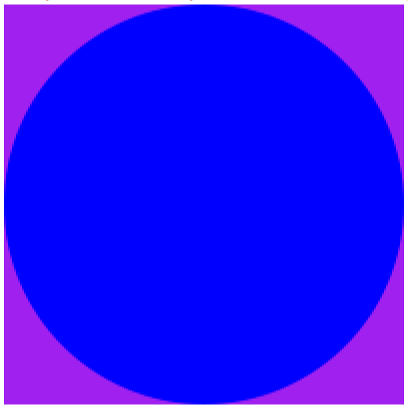
```
> (cs-demo (random 50 150))
```



```
> (cs-demo (random 50 150))
```



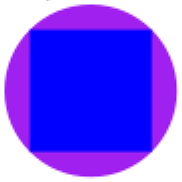
```
> (cs-demo (random 50 150))
```



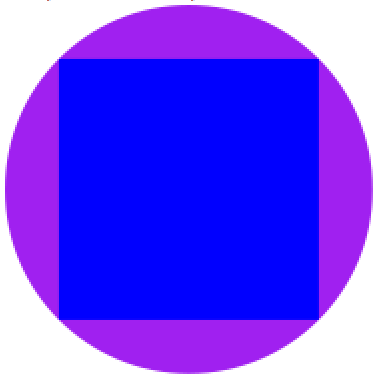
```
>
```

7:

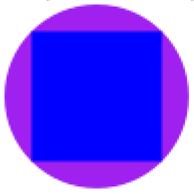
```
Language: racket, with debugging; memory limit: 128 MB.
> (cc-demo (random 50 150))
```



```
> (cc-demo (random 50 150))
```

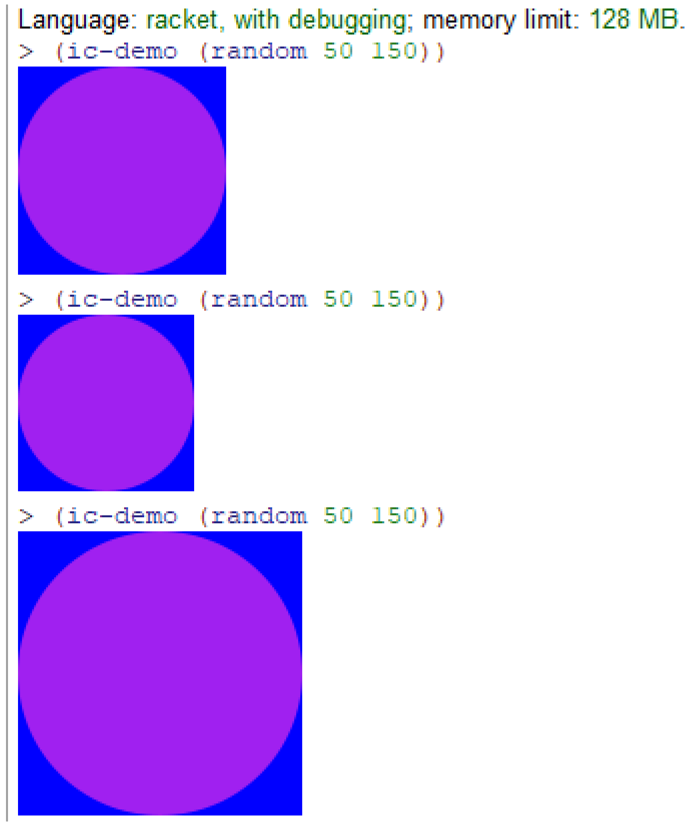


```
> (cc-demo (random 50 150))
```



```
>
```

8:



```
Language: racket, with debugging; memory limit: 128 MB.
> (ic-demo (random 50 150))
```

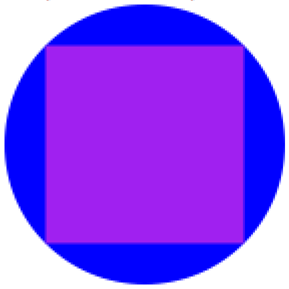
```
> (ic-demo (random 50 150))
```

```
> (ic-demo (random 50 150))
```

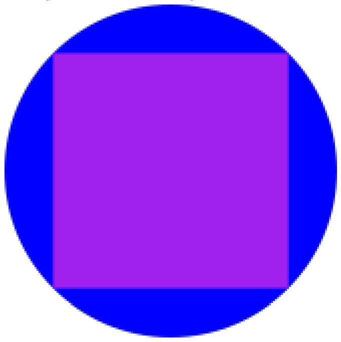
9:

Language: racket, with debugging; memory limit: 128 MB.

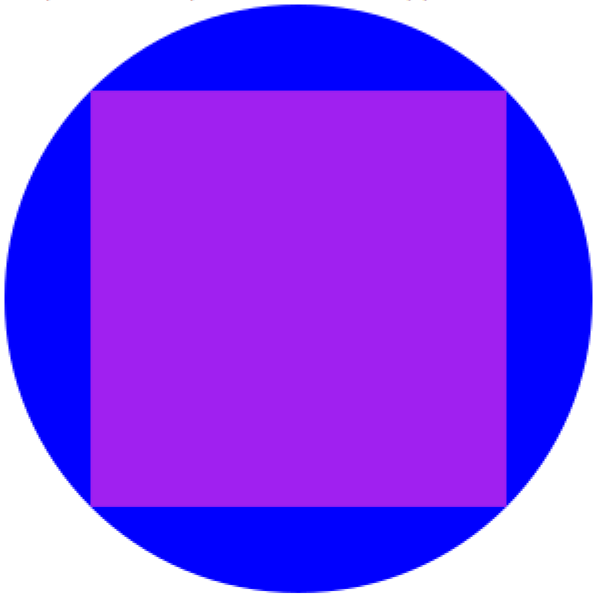
```
> (is-demo (random 50 150))
```



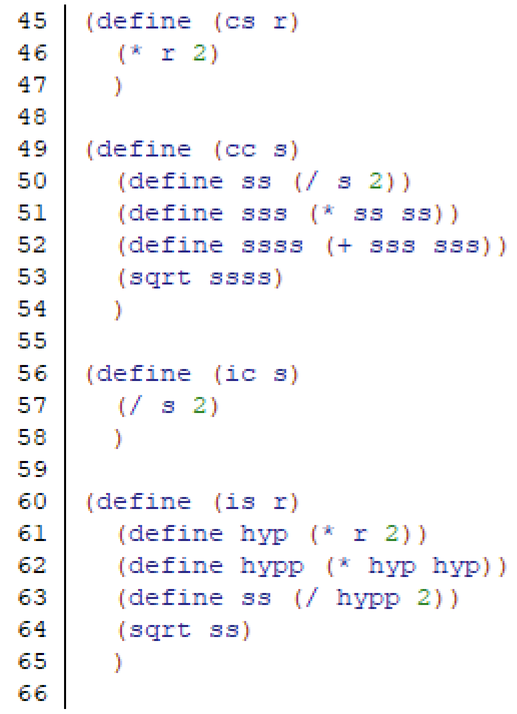
```
> (is-demo (random 50 150))
```



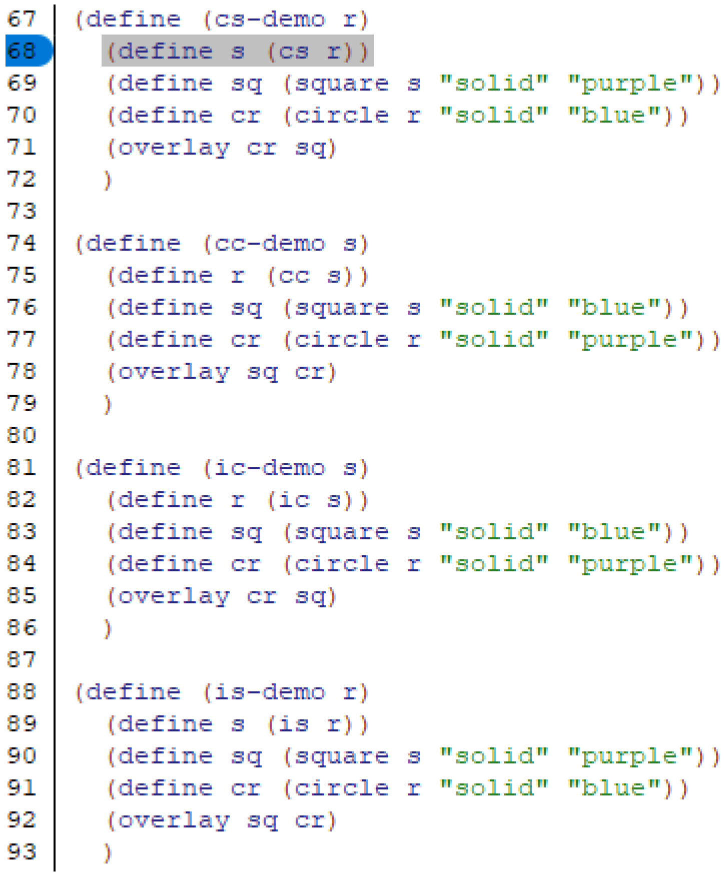
```
> (is-demo (random 50 150))
```



10:



```
(define (cs r)
  (* r 2)
  )

(define (cc s)
  (define ss (/ s 2))
  (define sss (* ss ss))
  (define ssss (+ sss sss))
  (sqrt ssss)
  )

(define (ic s)
  (/ s 2)
  )

(define (is r)
  (define hyp (* r 2))
  (define hypp (* hyp hyp))
  (define ss (/ hypp 2))
  (sqrt ss)
  )

```

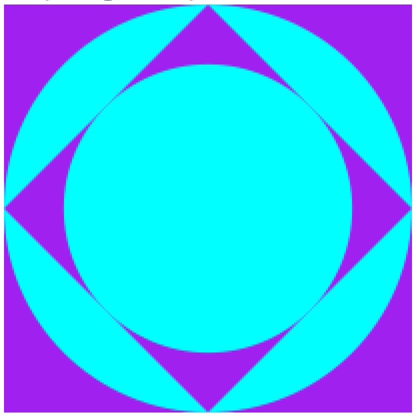


```
(define (cs-demo r)
  (define s (cs r))
  (define sq (square s "solid" "purple"))
  (define cr (circle r "solid" "blue"))
  (overlay cr sq)
  )

(define (cc-demo s)
  (define r (cc s))
  (define sq (square s "solid" "blue"))
  (define cr (circle r "solid" "purple"))
  (overlay sq cr)
  )

(define (ic-demo s)
  (define r (ic s))
  (define sq (square s "solid" "blue"))
  (define cr (circle r "solid" "purple"))
  (overlay cr sq)
  )

(define (is-demo r)
  (define s (is r))
  (define sq (square s "solid" "purple"))
  (define cr (circle r "solid" "blue"))
  (overlay sq cr)
  )
```

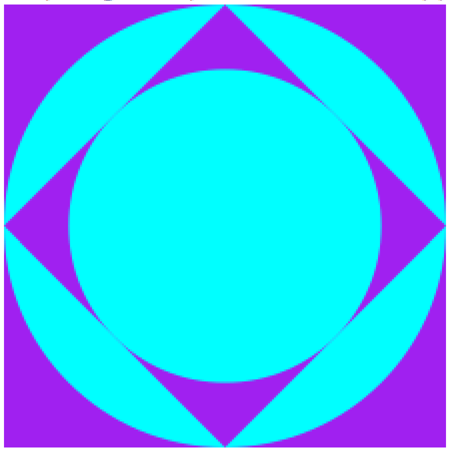
Task 3:

1:

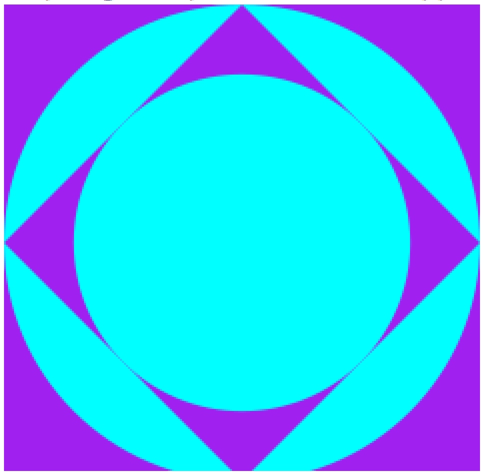
```
> (image-1 (random 200 300))
```



```
> (image-1 (random 200 300))
```

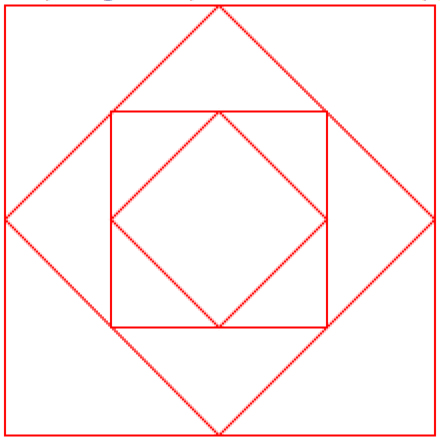


```
> (image-1 (random 200 300))
```

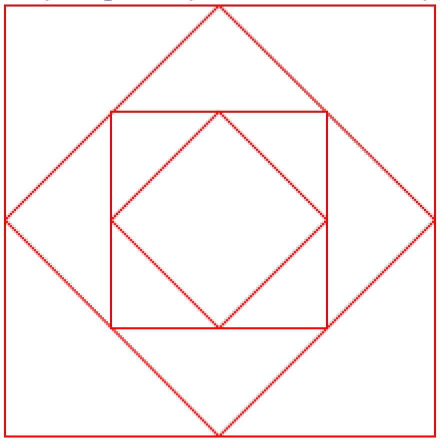


2:

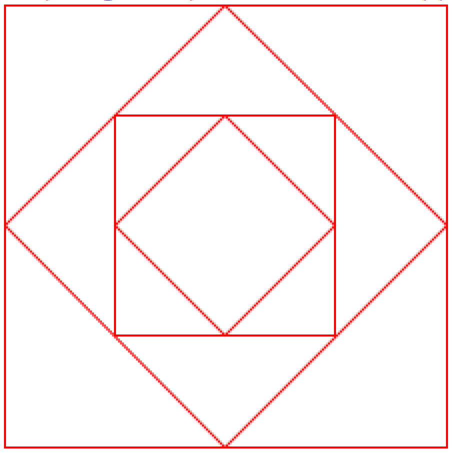
```
> (image-2 (random 200 300))
```



```
> (image-2 (random 200 300))
```



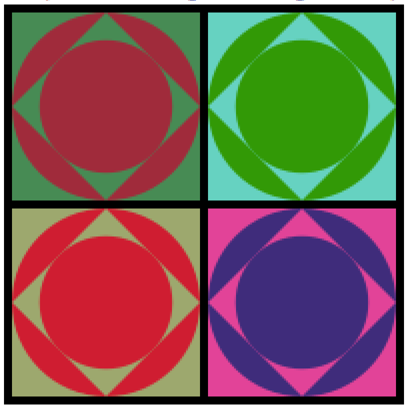
```
> (image-2 (random 200 300))
```



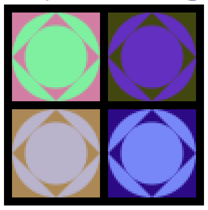
3:

Language: racket, with debugging; memory limit: 128 MB.

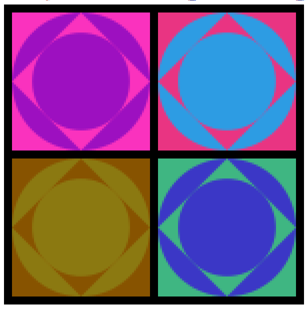
```
> (warholesque-image 200)
```

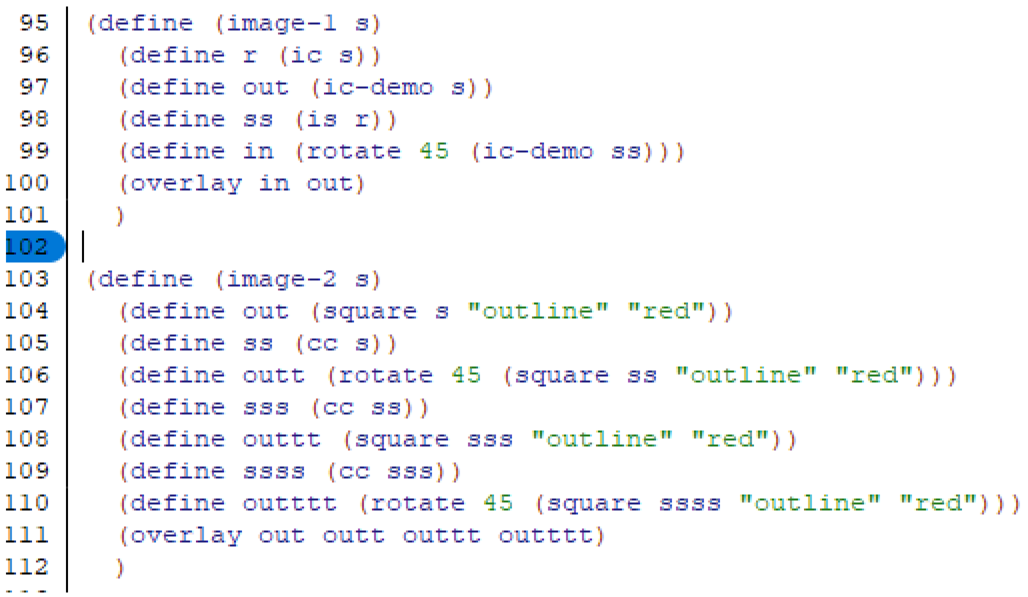


```
> (warholesque-image 100)
```

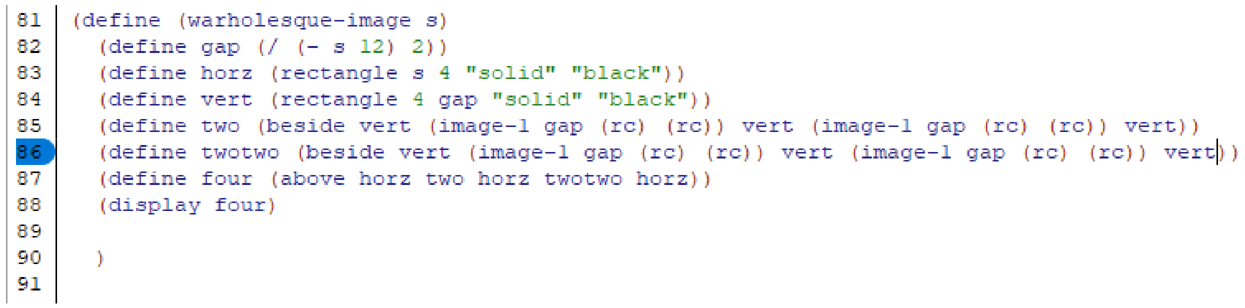


```
> (warholesque-image 150)
```

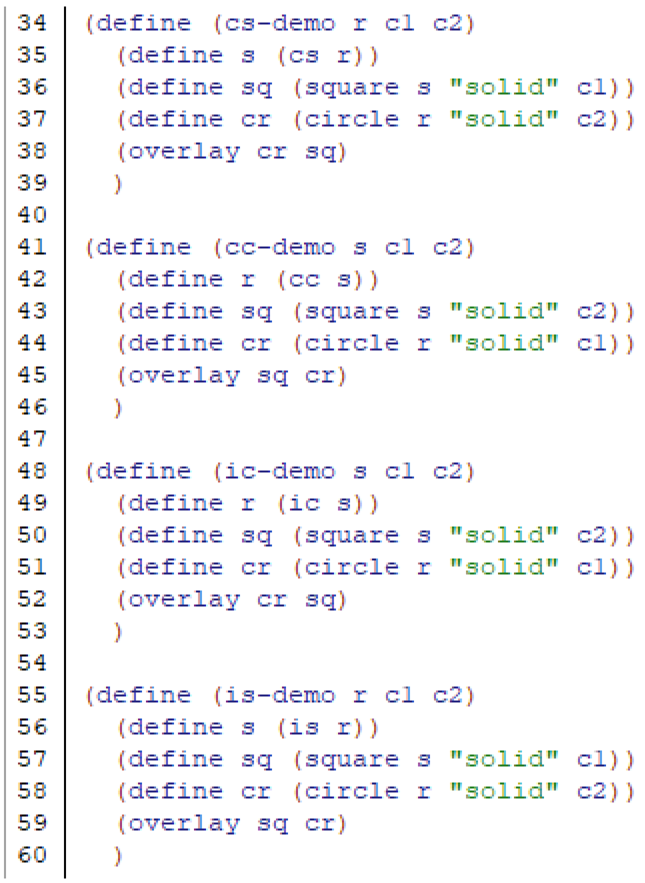




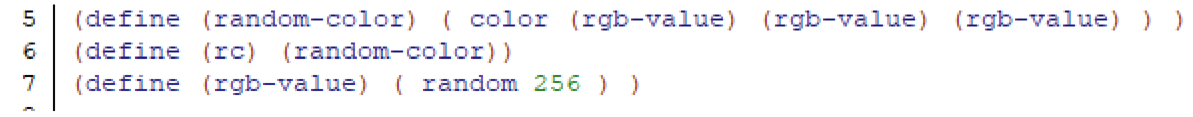
```
(define (image-1 s)
  (define r (ic s))
  (define out (ic-demo s))
  (define ss (is r))
  (define in (rotate 45 (ic-demo ss)))
  (overlay in out)
  )

(define (image-2 s)
  (define out (square s "outline" "red"))
  (define ss (cc s))
  (define outt (rotate 45 (square ss "outline" "red")))
  (define sss (cc ss))
  (define outtt (square sss "outline" "red"))
  (define ssss (cc sss))
  (define outttt (rotate 45 (square ssss "outline" "red")))
  (overlay out outt outtt outttt)
  )
```



```
(define (warholesque-image s)
  (define gap (/ (- s 12) 2))
  (define horz (rectangle s 4 "solid" "black"))
  (define vert (rectangle 4 gap "solid" "black"))
  (define two (beside vert (image-1 gap (rc) (rc)) vert (image-1 gap (rc) (rc)) vert))
  (define twotwo (beside vert (image-1 gap (rc) (rc)) vert (image-1 gap (rc) (rc)) vert))
  (define four (above horz two horz twotwo horz))
  (display four)

  )
```



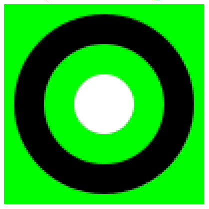
```
(define (cs-demo r c1 c2)
  (define s (cs r))
  (define sq (square s "solid" c1))
  (define cr (circle r "solid" c2))
  (overlay cr sq)
  )

(define (cc-demo s c1 c2)
  (define r (cc s))
  (define sq (square s "solid" c2))
  (define cr (circle r "solid" c1))
  (overlay sq cr)
  )

(define (ic-demo s c1 c2)
  (define r (ic s))
  (define sq (square s "solid" c2))
  (define cr (circle r "solid" c1))
  (overlay cr sq)
  )

(define (is-demo r c1 c2)
  (define s (is r))
  (define sq (square s "solid" c1))
  (define cr (circle r "solid" c2))
  (overlay sq cr)
  )
```



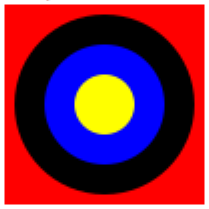
```
(define (random-color) ( color (rgb-value) (rgb-value) (rgb-value) ) )
(define (rc) (random-color))
(define (rgb-value) ( random 256 ) )
```

Task 4:

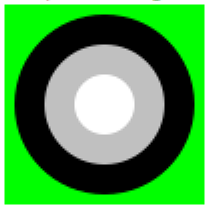
```
> (tile "green" "black" "green" "white")
```



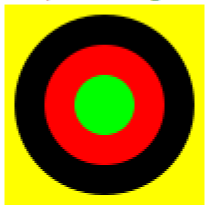
```
> (tile "red" "black" "blue" "yellow")
```



```
> (tile "green" "black" "silver" "white")
```

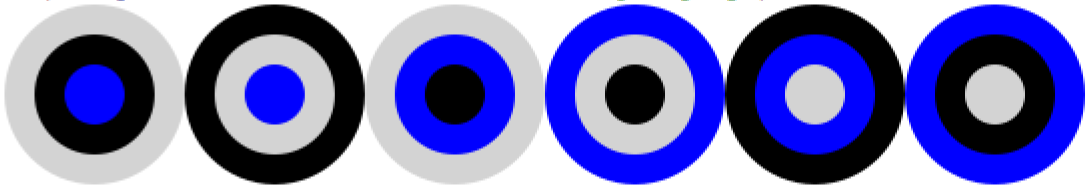


```
> (tile "yellow" "black" "red" "green")
```

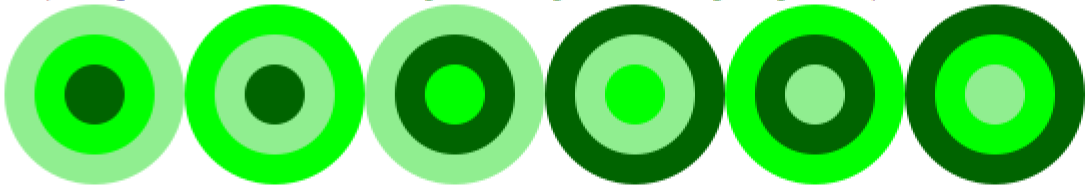


```
>
```

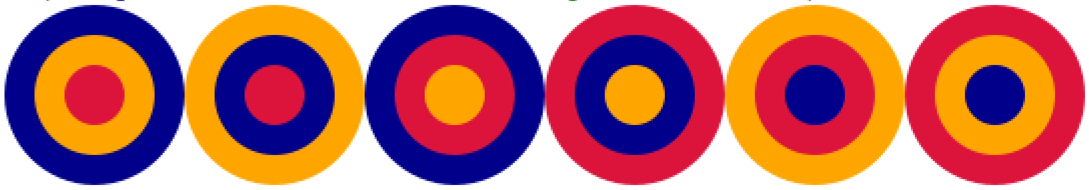
```
> (dot-permutations "blue" "black" "light gray")
```



```
> (dot-permutations "dark green" "green" "light green")
```



```
> (dot-permutations "crimson" "orange" "dark blue")
```



```
>
```

```
(define (tile a b c d)
  (define sq (square 100 "solid" a))
  (define out (circle 45 "solid" b))
  (define mid (circle 30 "solid" c))
  (define in (circle 15 "solid" d))
  (overlay in mid out sq)
  )


(define (dot-permutations a b c)
  (define c1 (overlay (circle 15 "solid" a) (circle 30 "solid" b) (circle 45 "solid" c)))
  (define c2 (overlay (circle 15 "solid" a) (circle 30 "solid" c) (circle 45 "solid" b)))
  (define c3 (overlay (circle 15 "solid" b) (circle 30 "solid" a) (circle 45 "solid" c)))
  (define c4 (overlay (circle 15 "solid" b) (circle 30 "solid" c) (circle 45 "solid" a)))
  (define c5 (overlay (circle 15 "solid" c) (circle 30 "solid" a) (circle 45 "solid" b)))
  (define c6 (overlay (circle 15 "solid" c) (circle 30 "solid" b) (circle 45 "solid" a)))
  (beside c1 c2 c3 c4 c5 c6)
  )
```