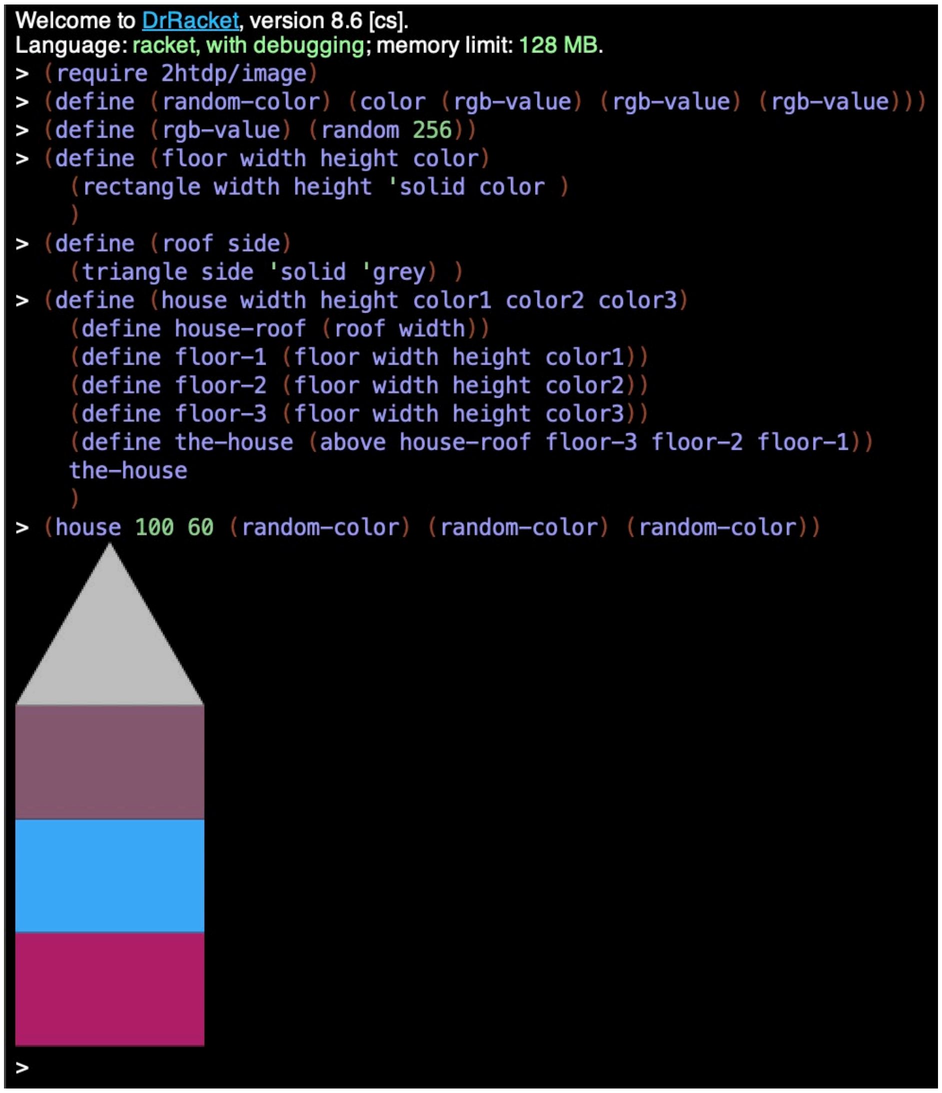
Racket Programming Assignment# 2: Racket Functions and Recursion

Learning Abstract

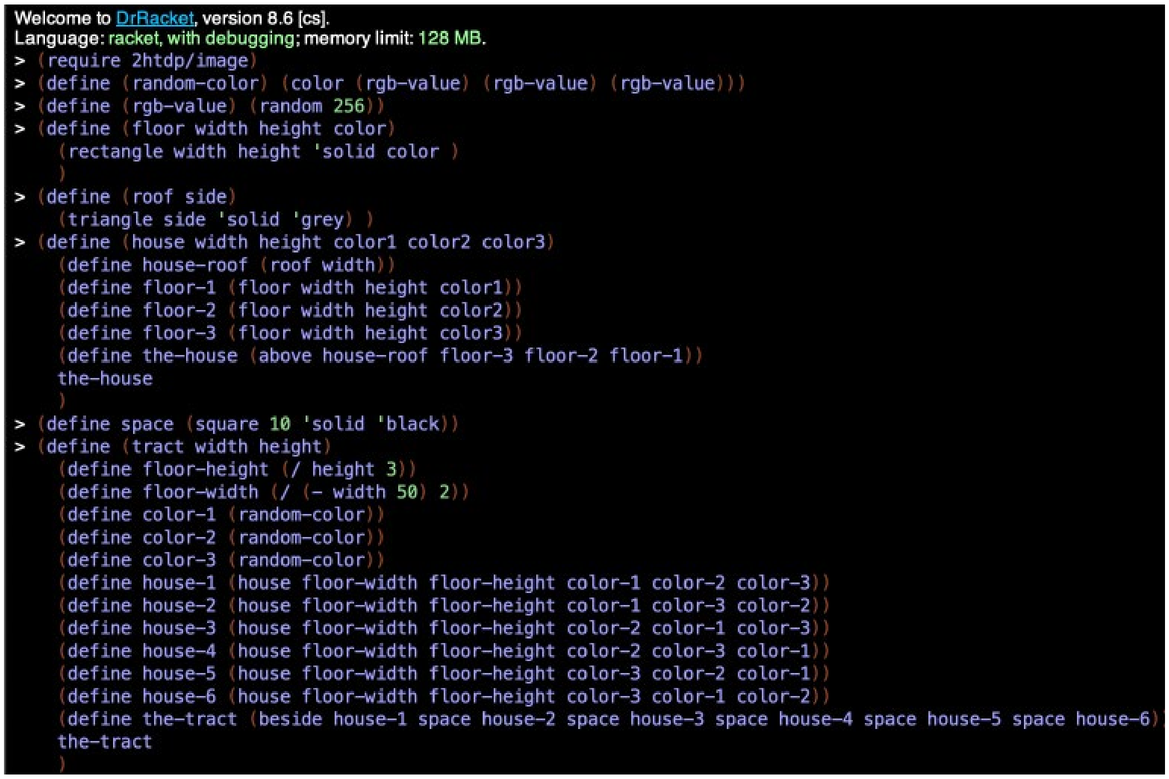
This assignment features programs that generate images in the context of the 2htdp/image library, most of which are recursive in nature.

Task 1: Colorful Permutations of Tract Houses

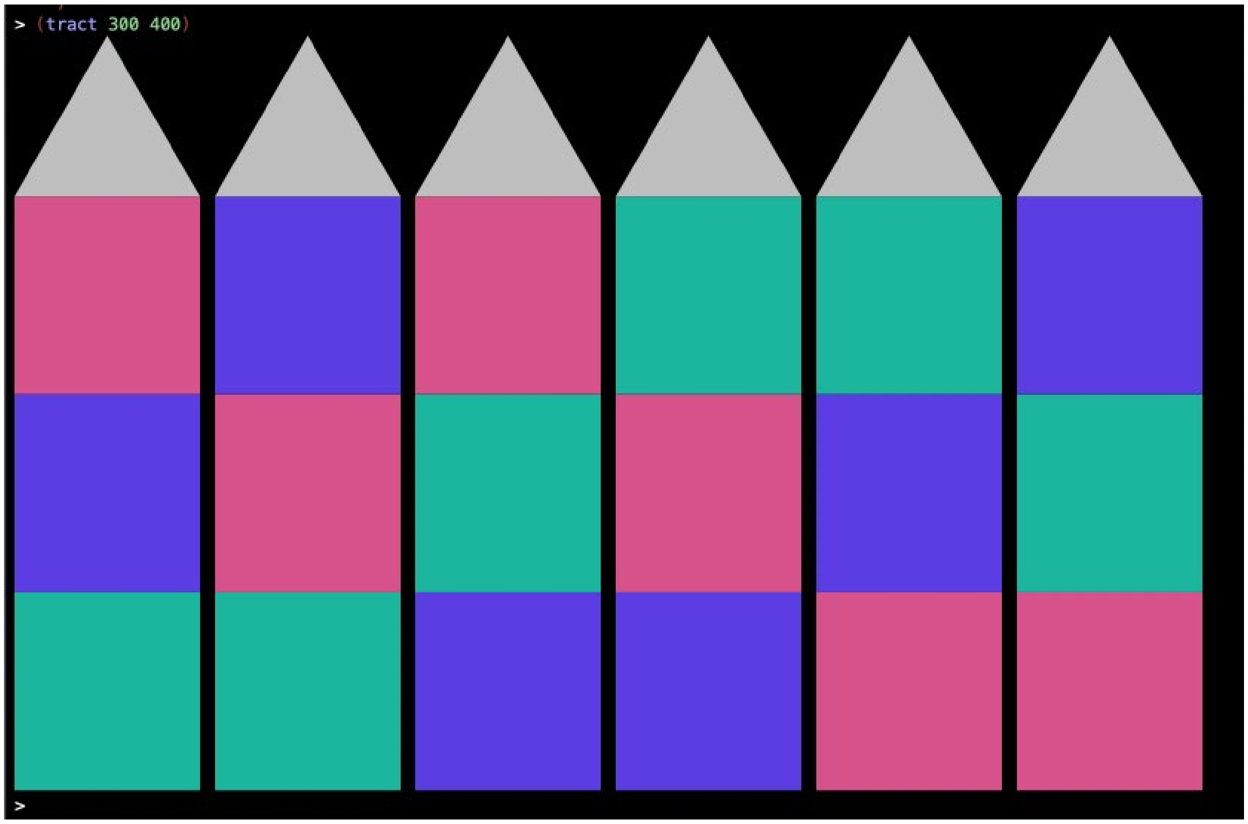


```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (require 2htdp/image)
> (define (random-color) (color (rgb-value) (rgb-value) (rgb-value)))
> (define (rgb-value) (random 256))
> (define (floor width height color)
    (rectangle width height 'solid color )
    )
> (define (roof side)
    (triangle side 'solid 'grey) )
> (define (house width height color1 color2 color3)
    (define house-roof (roof width))
    (define floor-1 (floor width height color1))
    (define floor-2 (floor width height color2))
    (define floor-3 (floor width height color3))
    (define the-house (above house-roof floor-3 floor-2 floor-1))
    the-house
    )
> (house 100 60 (random-color) (random-color) (random-color))
```

```
>
```



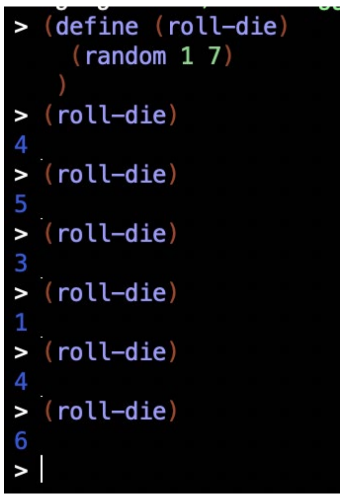
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (require 2htdp/image)
> (define (random-color) (color (rgb-value) (rgb-value) (rgb-value)))
> (define (rgb-value) (random 256))
> (define (floor width height color)
    (rectangle width height 'solid color )
    )
> (define (roof side)
    (triangle side 'solid 'grey) )
> (define (house width height color1 color2 color3)
    (define house-roof (roof width))
    (define floor-1 (floor width height color1))
    (define floor-2 (floor width height color2))
    (define floor-3 (floor width height color3))
    (define the-house (above house-roof floor-3 floor-2 floor-1))
    the-house
    )
> (define space (square 10 'solid 'black))
> (define (tract width height)
    (define floor-height (/ height 3))
    (define floor-width (/ (- width 50) 2))
    (define color-1 (random-color))
    (define color-2 (random-color))
    (define color-3 (random-color))
    (define house-1 (house floor-width floor-height color-1 color-2 color-3))
    (define house-2 (house floor-width floor-height color-1 color-3 color-2))
    (define house-3 (house floor-width floor-height color-2 color-1 color-3))
    (define house-4 (house floor-width floor-height color-2 color-3 color-1))
    (define house-5 (house floor-width floor-height color-3 color-2 color-1))
    (define house-6 (house floor-width floor-height color-3 color-1 color-2))
    (define the-tract (beside house-1 space house-2 space house-3 space house-4 space house-5 space house-6)
    the-tract
    )
```



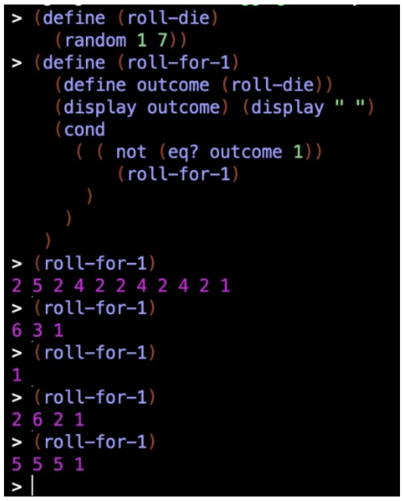
```
> (tract 300 400)
```

```
>
```

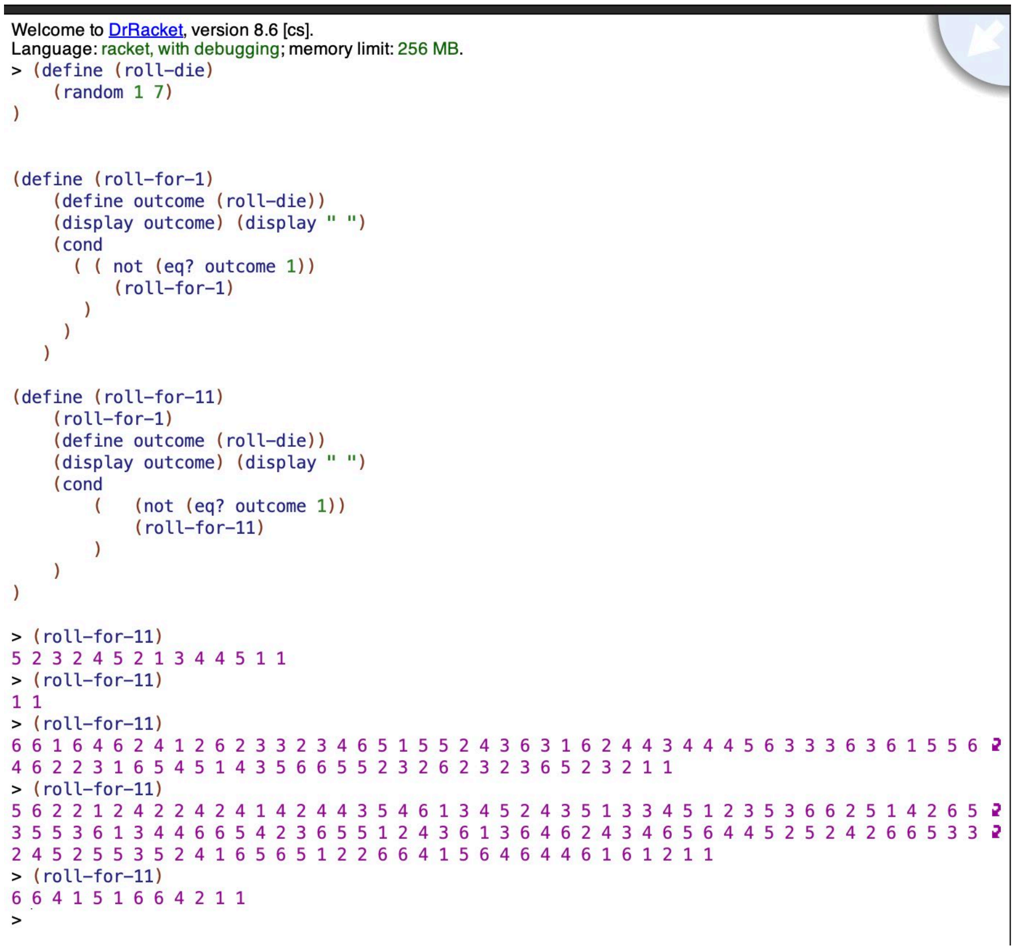
Task 2: Dice



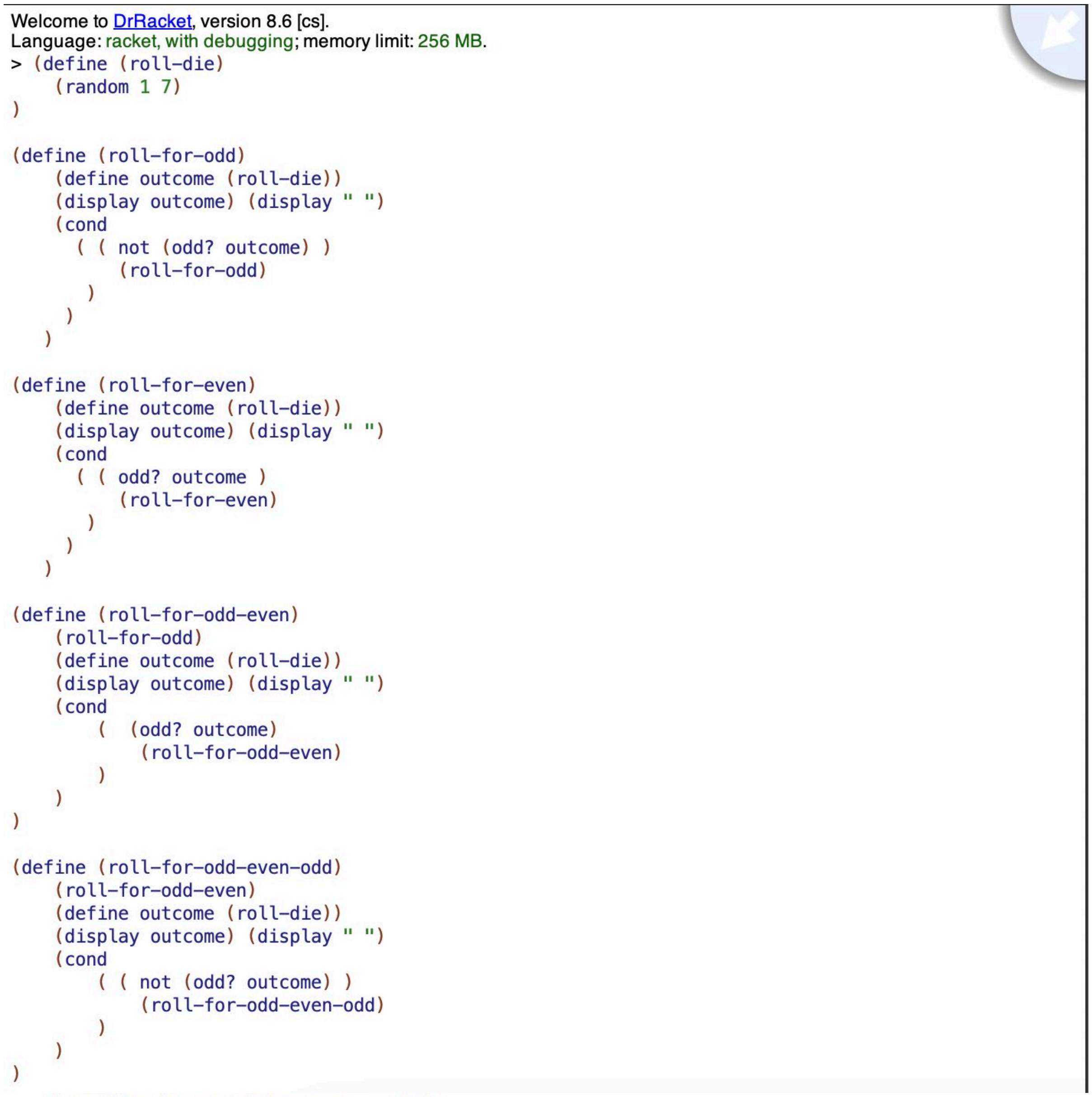
```
> (define (roll-die)
    (random 1 7)
   )
> (roll-die)
4
> (roll-die)
5
> (roll-die)
3
> (roll-die)
1
> (roll-die)
4
> (roll-die)
6
> |
```



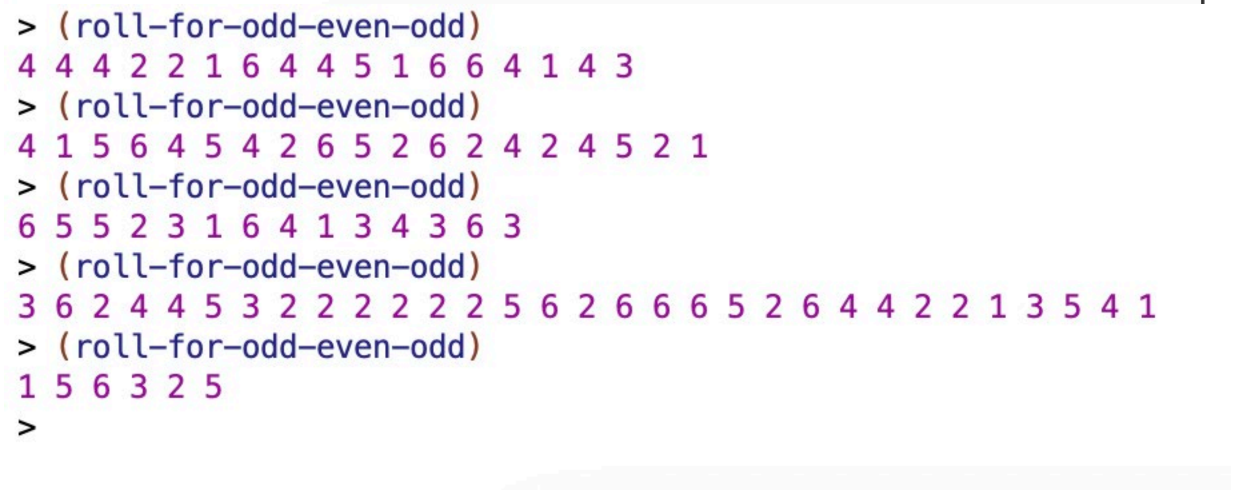
```
> (define (roll-die)
    (random 1 7))
> (define (roll-for-1)
    (define outcome (roll-die))
    (display outcome) (display " ")
    (cond
      ( ( not (eq? outcome 1))
          (roll-for-1)
        )
      )
    )
> (roll-for-1)
2 5 2 4 2 2 4 2 4 2 1
> (roll-for-1)
6 3 1
> (roll-for-1)
1
> (roll-for-1)
2 6 2 1
> (roll-for-1)
5 5 1
> |
```



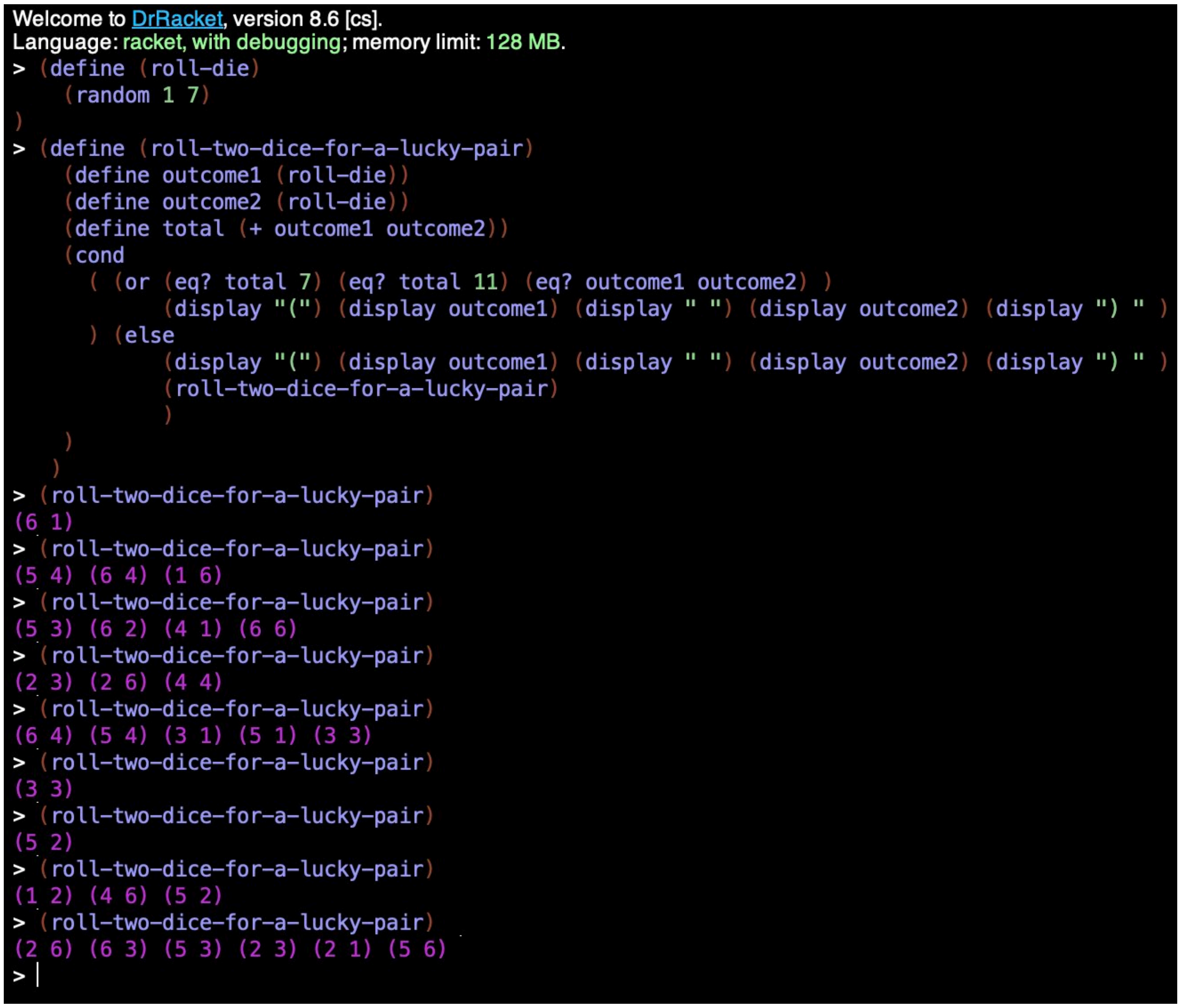
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> (define (roll-die)
    (random 1 7)
)


(define (roll-for-1)
    (define outcome (roll-die))
    (display outcome) (display " ")
    (cond
      ( ( not (eq? outcome 1))
          (roll-for-1)
        )
      )
   )

(define (roll-for-11)
    (roll-for-1)
    (define outcome (roll-die))
    (display outcome) (display " ")
    (cond
        (    (not (eq? outcome 1))
             (roll-for-11)
        )
    )
)

> (roll-for-11)
5 2 3 2 4 5 2 1 3 4 4 5 1 1
> (roll-for-11)
1 1
> (roll-for-11)
6 6 1 6 4 6 2 4 1 2 6 2 3 3 2 3 4 6 5 1 5 5 2 4 3 6 3 1 6 2 4 4 3 4 4 4 5 6 3 3 3 6 3 6 1 5 5 6
4 6 2 2 3 1 6 5 4 5 1 4 3 5 6 6 5 5 2 3 2 6 2 3 2 3 6 5 2 3 2 1 1
> (roll-for-11)
5 6 2 2 1 2 4 2 2 4 2 4 1 4 2 4 4 3 5 4 6 1 3 4 5 2 4 3 5 1 3 3 4 5 1 2 3 5 3 6 6 2 5 1 4 2 6 5
3 5 5 3 6 1 3 4 4 6 6 5 4 2 3 6 5 5 1 2 4 3 6 1 3 6 4 6 2 4 3 4 6 5 6 4 4 5 2 5 2 4 2 6 6 5 3 3
2 4 5 2 5 5 3 5 2 4 1 6 5 6 5 1 2 2 6 6 4 1 5 6 4 6 4 4 6 1 6 1 2 1 1
> (roll-for-11)
6 6 4 1 5 1 6 6 4 2 1 1
>
```



```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> (define (roll-die)
    (random 1 7)
)

(define (roll-for-odd)
    (define outcome (roll-die))
    (display outcome) (display " ")
    (cond
      ( ( not (odd? outcome) )
          (roll-for-odd)
       )
     )
   )

(define (roll-for-even)
    (define outcome (roll-die))
    (display outcome) (display " ")
    (cond
      ( ( odd? outcome )
          (roll-for-even)
       )
     )
   )

(define (roll-for-odd-even)
    (roll-for-odd)
    (define outcome (roll-die))
    (display outcome) (display " ")
    (cond
        (  (odd? outcome)
            (roll-for-odd-even)
        )
    )
)

(define (roll-for-odd-even-odd)
    (roll-for-odd-even)
    (define outcome (roll-die))
    (display outcome) (display " ")
    (cond
        ( ( not (odd? outcome) )
            (roll-for-odd-even-odd)
        )
    )
)
```

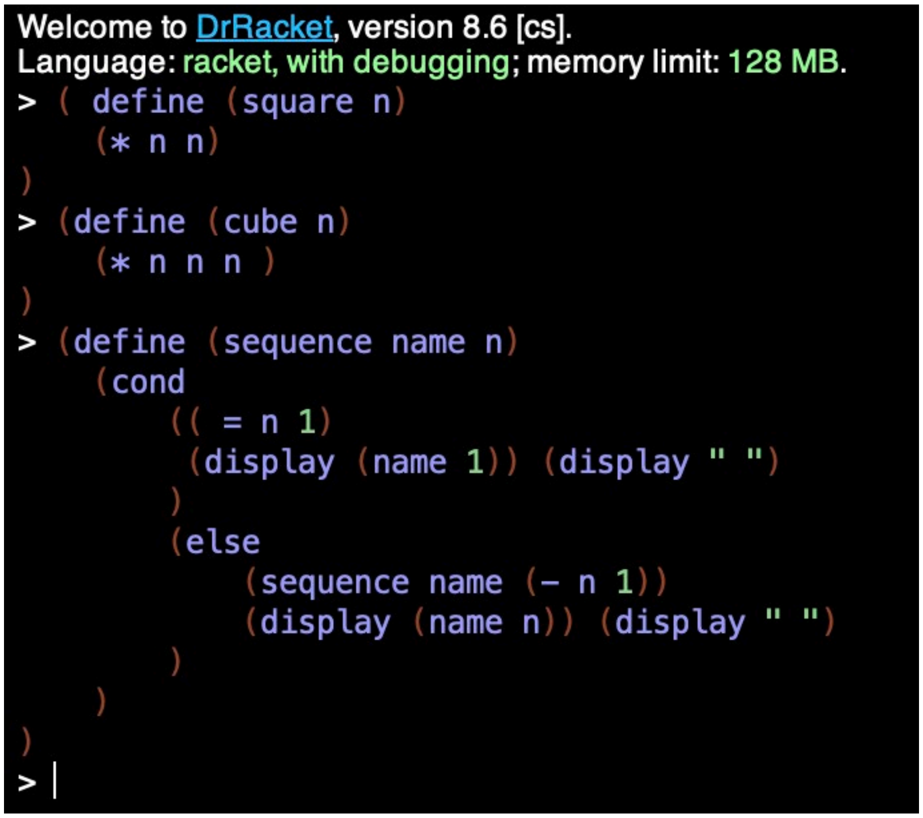


```
> (roll-for-odd-even-odd)
4 4 4 2 2 1 6 4 4 5 1 6 6 4 1 4 3
> (roll-for-odd-even-odd)
4 1 5 6 4 5 4 2 6 5 2 6 2 4 2 4 5 2 1
> (roll-for-odd-even-odd)
6 5 5 2 3 1 6 4 1 3 4 3 6 3
> (roll-for-odd-even-odd)
3 6 2 4 4 5 3 2 2 2 2 2 2 5 6 2 6 6 6 5 2 6 4 4 2 2 1 3 5 4 1
> (roll-for-odd-even-odd)
1 5 6 3 2 5
>
```

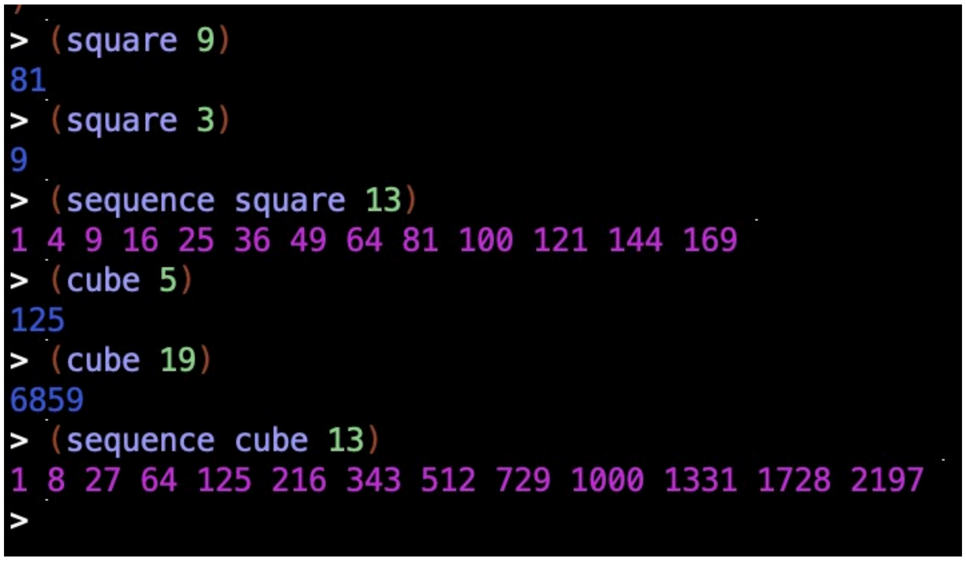


```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (define (roll-die)
    (random 1 7)
)
> (define (roll-two-dice-for-a-lucky-pair)
    (define outcome1 (roll-die))
    (define outcome2 (roll-die))
    (define total (+ outcome1 outcome2))
    (cond
      ( (or (eq? total 7) (eq? total 11) (eq? outcome1 outcome2) )
            (display "(") (display outcome1) (display " ") (display outcome2) (display ") " )
      ) (else
            (display "(") (display outcome1) (display " ") (display outcome2) (display ") " )
            (roll-two-dice-for-a-lucky-pair)
            )
    )
   )
> (roll-two-dice-for-a-lucky-pair)
(6 1)
> (roll-two-dice-for-a-lucky-pair)
(5 4) (6 4) (1 6)
> (roll-two-dice-for-a-lucky-pair)
(5 3) (6 2) (4 1) (6 6)
> (roll-two-dice-for-a-lucky-pair)
(2 3) (2 6) (4 4)
> (roll-two-dice-for-a-lucky-pair)
(6 4) (5 4) (3 1) (5 1) (3 3)
> (roll-two-dice-for-a-lucky-pair)
(3 3)
> (roll-two-dice-for-a-lucky-pair)
(5 2)
> (roll-two-dice-for-a-lucky-pair)
(1 2) (4 6) (5 2)
> (roll-two-dice-for-a-lucky-pair)
(2 6) (6 3) (5 3) (2 3) (2 1) (5 6)
> 
```

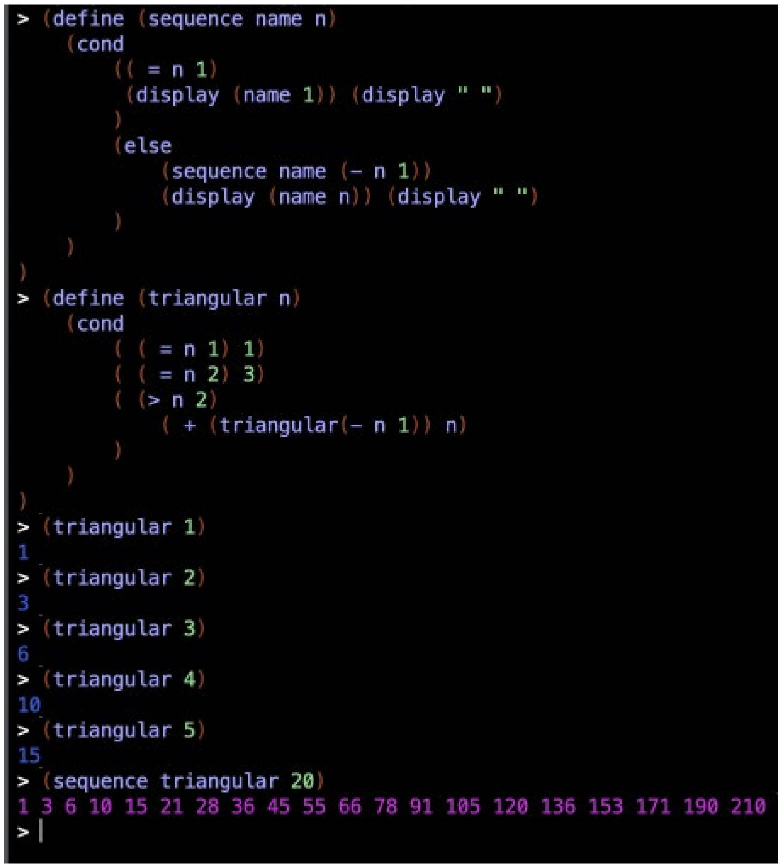
Task 3: Number Sequences



```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( define (square n)
    (* n n)
)
> (define (cube n)
    (* n n n )
)
> (define (sequence name n)
    (cond
        (( = n 1)
         (display (name 1)) (display " ")
        )
        (else
            (sequence name (- n 1))
            (display (name n)) (display " ")
        )
    )
)
> |
```

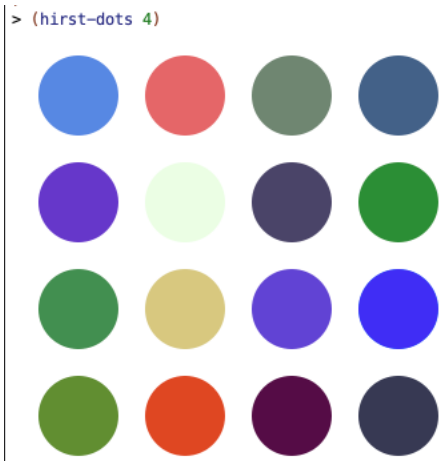


```
> (square 9)
81
> (square 3)
9
> (sequence square 13)
1 4 9 16 25 36 49 64 81 100 121 144 169
> (cube 5)
125
> (cube 19)
6859
> (sequence cube 13)
1 8 27 64 125 216 343 512 729 1000 1331 1728 2197
>
```

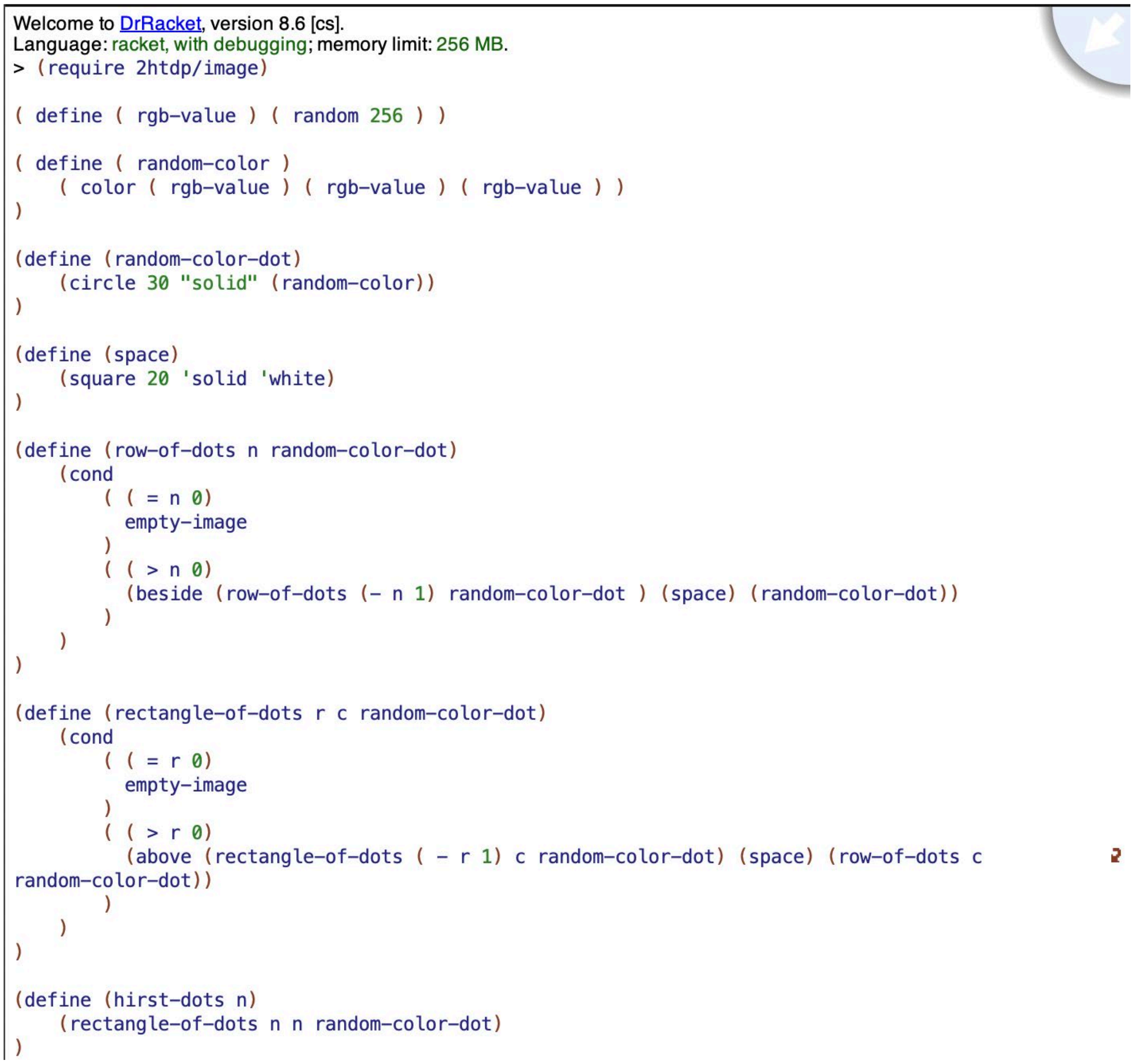


```
> (define (sequence name n)
    (cond
        (( = n 1)
         (display (name 1)) (display " ")
        )
        (else
            (sequence name (- n 1))
            (display (name n)) (display " ")
        )
    )
)
> (define (triangular n)
    (cond
        ( ( = n 1) 1)
        ( ( = n 2) 3)
        ( (> n 2)
            ( + (triangular(- n 1)) n)
        )
    )
)
> (triangular 1)
1
> (triangular 2)
3
> (triangular 3)
6
> (triangular 4)
10
> (triangular 5)
15
> (sequence triangular 20)
1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136 153 171 190 210
> |
```

Task 4: Hirst Dots



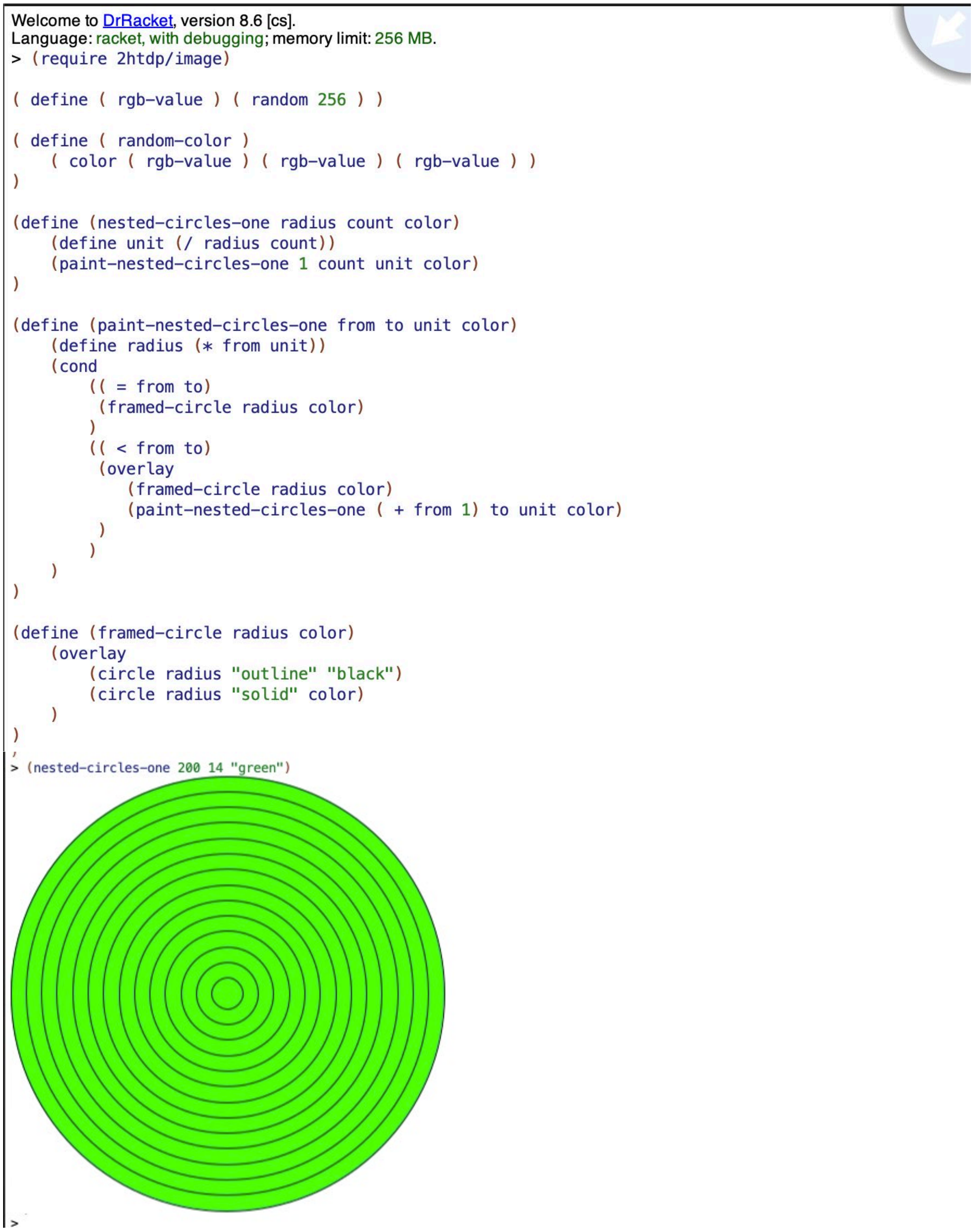
```
> (hirst-dots 4)
> (hirst-dots 10)
```



Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 256 MB.

```
> (require 2htdp/image)

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

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

(define (random-color-dot)
    (circle 30 "solid" (random-color))
)

(define (space)
    (square 20 'solid 'white)
)

(define (row-of-dots n random-color-dot)
    (cond
        ( ( = n 0)
          empty-image
        )
        ( ( > n 0)
          (beside (row-of-dots (- n 1) random-color-dot ) (space) (random-color-dot))
        )
    )
)

(define (rectangle-of-dots r c random-color-dot)
    (cond
        ( ( = r 0)
          empty-image
        )
        ( ( > r 0)
          (above (rectangle-of-dots ( - r 1) c random-color-dot) (space) (row-of-dots c
random-color-dot))
        )
    )
)

(define (hirst-dots n)
    (rectangle-of-dots n n random-color-dot)
)
```

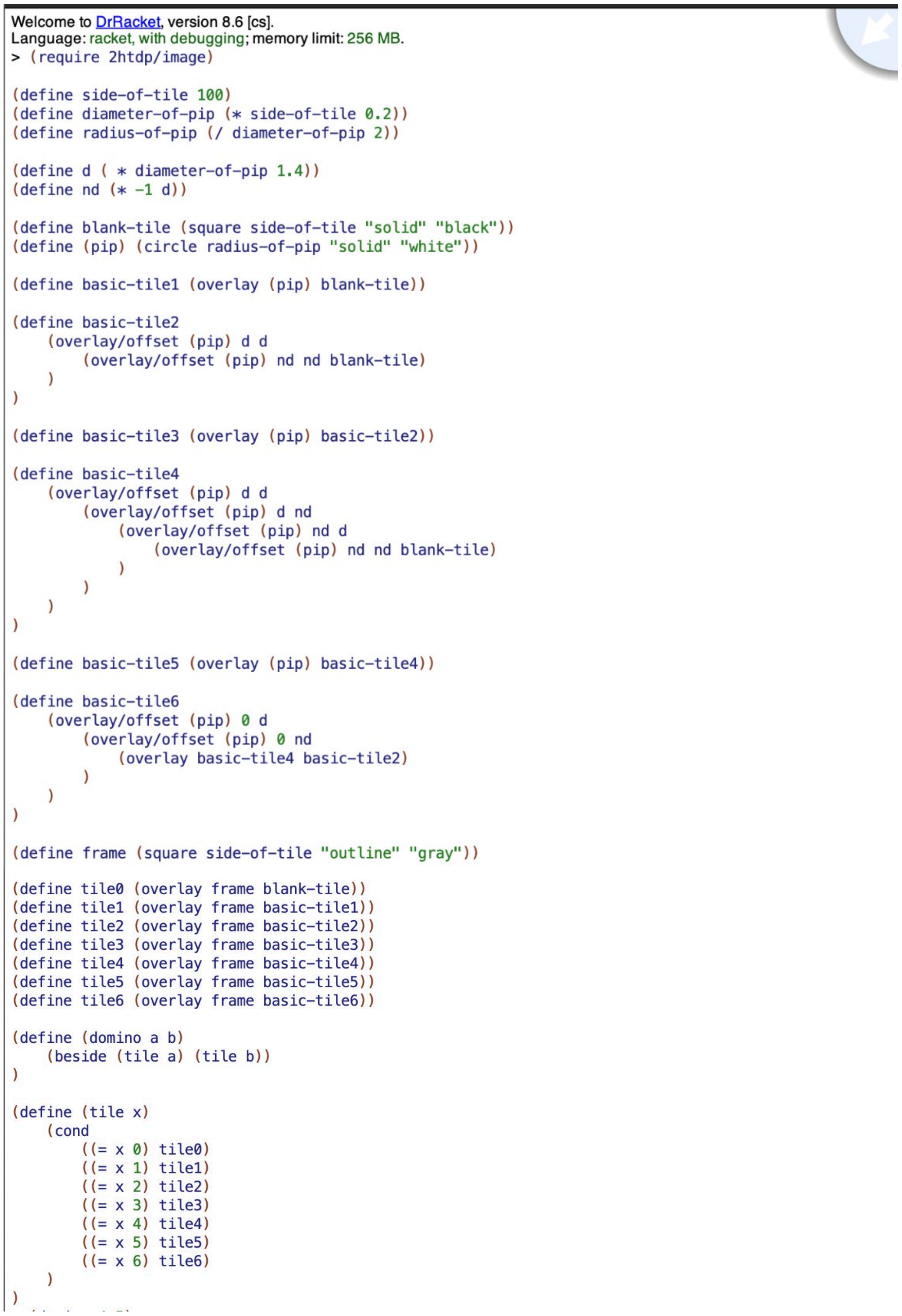
Task 5 : Channeling Frank Stella



```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> (require 2htdp/image)

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

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

(define (nested-circles-one radius count color)
    (define unit (/ radius count))
    (paint-nested-circles-one 1 count unit color)
)

(define (paint-nested-circles-one from to unit color)
    (define radius (* from unit))
    (cond
        (( = from to)
         (framed-circle radius color)
        )
        (( < from to)
         (overlay
            (framed-circle radius color)
            (paint-nested-circles-one ( + from 1) to unit color)
         )
        )
    )
)

(define (framed-circle radius color)
    (overlay
        (circle radius "outline" "black")
        (circle radius "solid" color)
    )
)
> (nested-circles-one 200 14 "green")
```

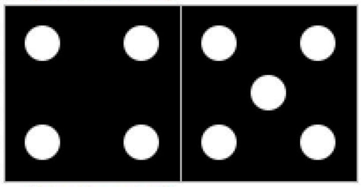
```
>
```

Task 6: Dominos

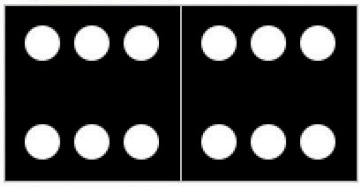


```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 256 MB.
> (require 2htdp/image)

(define side-of-tile 100)
(define diameter-of-pip (* side-of-tile 0.2))
(define radius-of-pip (/ diameter-of-pip 2))

(define d ( * diameter-of-pip 1.4))
(define nd (* -1 d))

(define blank-tile (square side-of-tile "solid" "black"))
(define (pip) (circle radius-of-pip "solid" "white"))

(define basic-tile1 (overlay (pip) blank-tile))

(define basic-tile2
    (overlay/offset (pip) d d
        (overlay/offset (pip) nd nd blank-tile)
    )
)

(define basic-tile3 (overlay (pip) basic-tile2))

(define basic-tile4
    (overlay/offset (pip) d d
        (overlay/offset (pip) d nd
            (overlay/offset (pip) nd d
                (overlay/offset (pip) nd nd blank-tile)
            )
        )
    )
)

(define basic-tile5 (overlay (pip) basic-tile4))

(define basic-tile6
    (overlay/offset (pip) 0 d
        (overlay/offset (pip) 0 nd
            (overlay basic-tile4 basic-tile2)
        )
    )
)

(define frame (square side-of-tile "outline" "gray"))

(define tile0 (overlay frame blank-tile))
(define tile1 (overlay frame basic-tile1))
(define tile2 (overlay frame basic-tile2))
(define tile3 (overlay frame basic-tile3))
(define tile4 (overlay frame basic-tile4))
(define tile5 (overlay frame basic-tile5))
(define tile6 (overlay frame basic-tile6))

(define (domino a b)
    (beside (tile a) (tile b))
)

(define (tile x)
    (cond
        ((= x 0) tile0)
        ((= x 1) tile1)
        ((= x 2) tile2)
        ((= x 3) tile3)
        ((= x 4) tile4)
        ((= x 5) tile5)
        ((= x 6) tile6)
    )
)
```

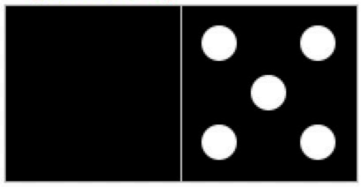
```
> (domino 4 5)
```



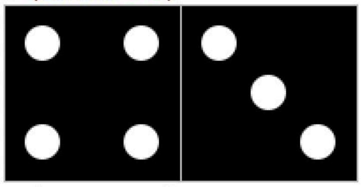
```
> (domino 6 6)
```



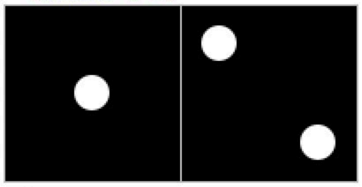
```
> (domino 0 5)
```



```
> (domino 4 3)
```



```
> (domino 1 2)
```



```
>
```

Task 7: Creation

```
> (require 2htdp/image)

(define (body)
    (overlay
   (overlay
    (overlay
     (rotate 75 (radial-star 8 8 64 "solid" "purple"))
    (rotate 20 (radial-star 8 8 64 "solid" "maroon")))
            (radial-star 8 8 64 "solid" "darkslategray"))
           (radial-star 32 30 64 "solid" "magenta"))
)



(define (my-creation)
    (above
    (body )
    (scale/xy 1 1/2 (flip-vertical (radial-star 32 30 64 "solid" "gray"))))
)

> (my-creation)
```