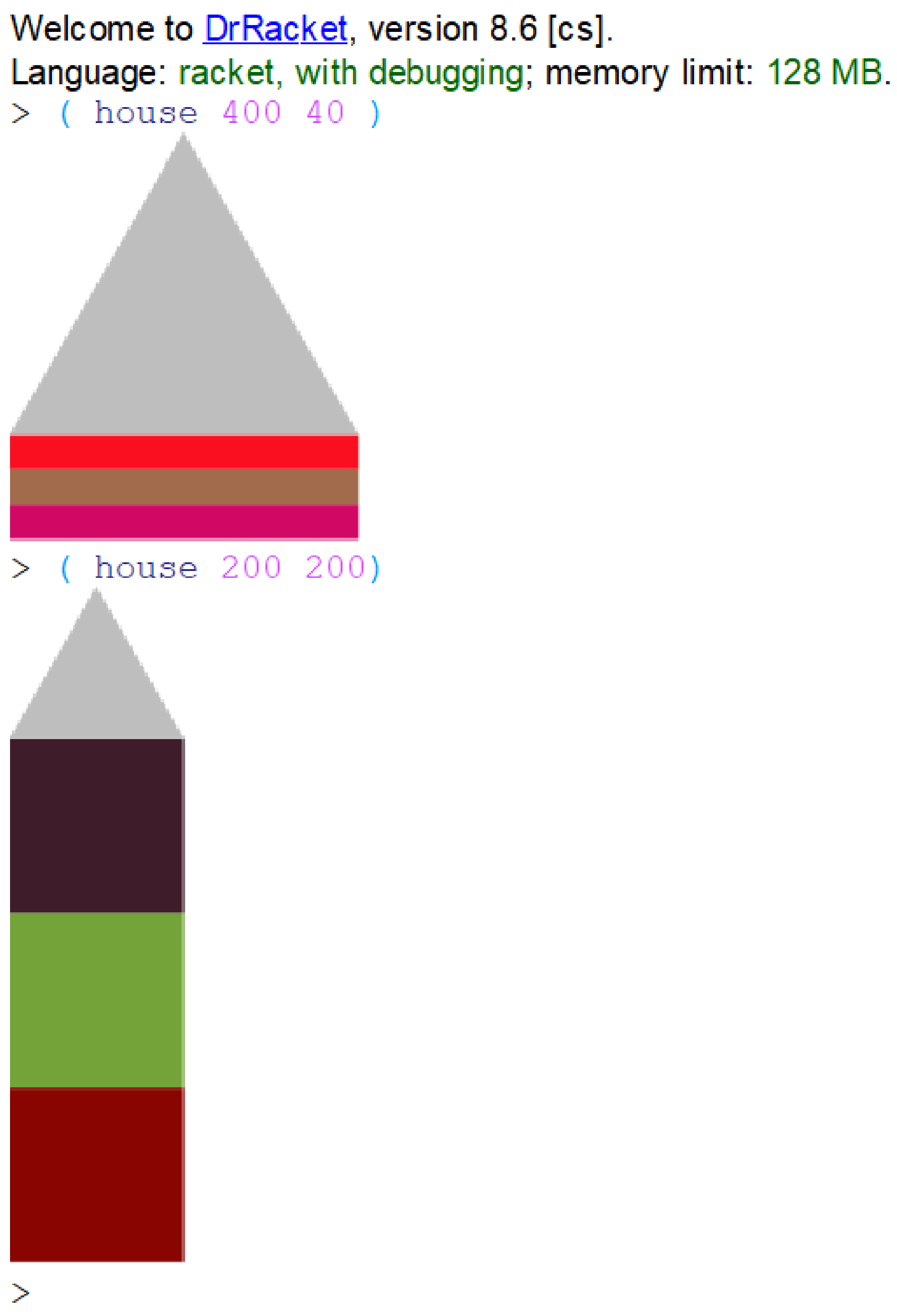
Racket Programming Assignment #2: Racket Functions and Recursion

Learning Abstract: We have seven tasks with commands and limitations in this racket assignment. For our first work, we had to design a home and a row of houses to assist me in doing so. I examined and applied the majority of the code we learnt in lesson 2 for producing a checkerboard. I used what we learned in lesson 3 with the flip coin to produce a roll dice for the second job. Task three was unquestionably the most difficult of them all. I used what we learned in lesson 3 with the tiles to assist produce the hirst dots for the fourth task. For the stella form, I took the stella square code as a guide but changed it to a circular shape. For the remainder of the work, we utilized the code provided to us to produce them and modified them. Finally, I utilized the function overlay/xy to make a Mario head for my creation.

Task 1: Colorful Permutations of Tract Houses

Demo for house:

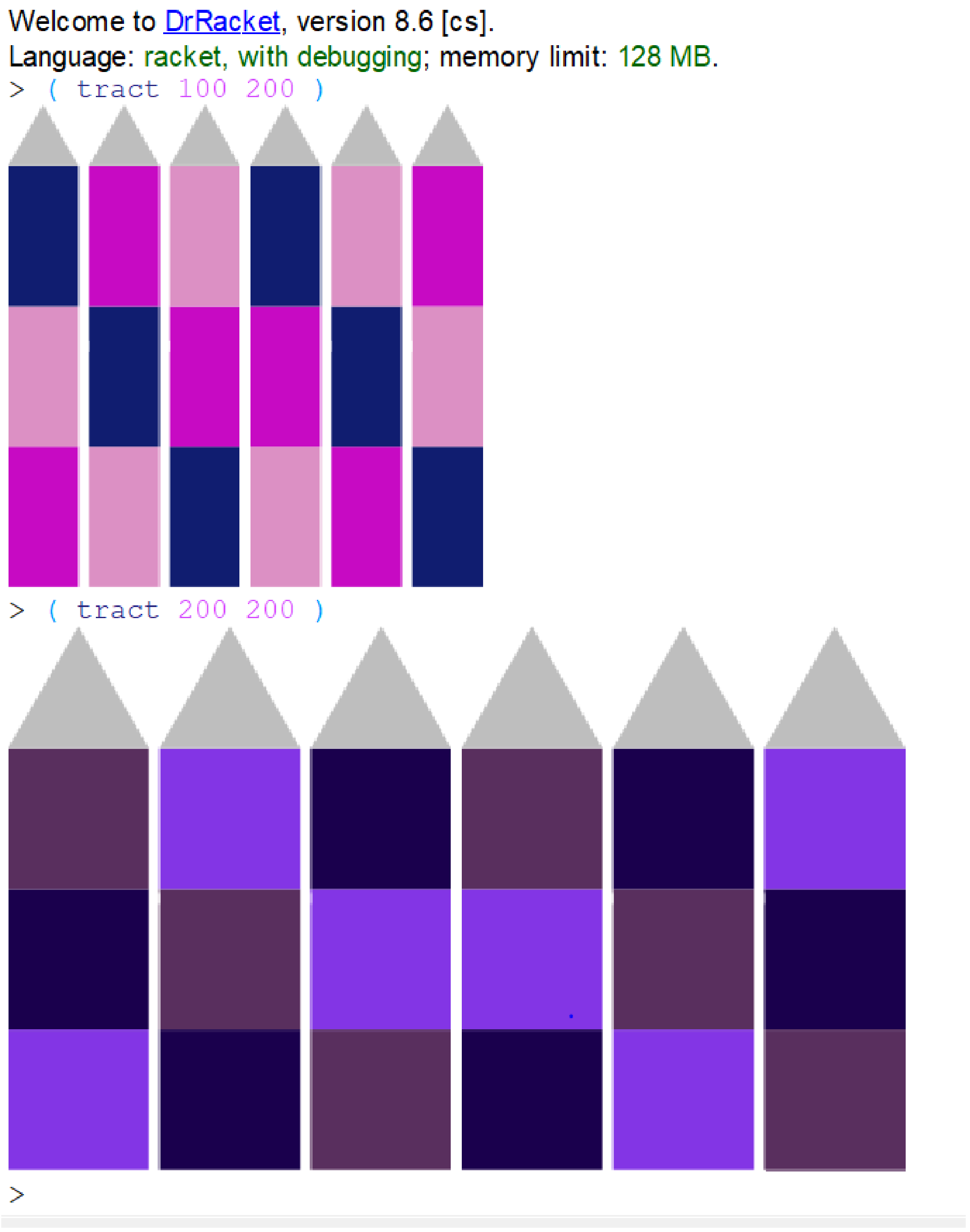


```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( house 400 40 )
```

```
> ( house 200 200)
```

```
>
```

Demo for tract:



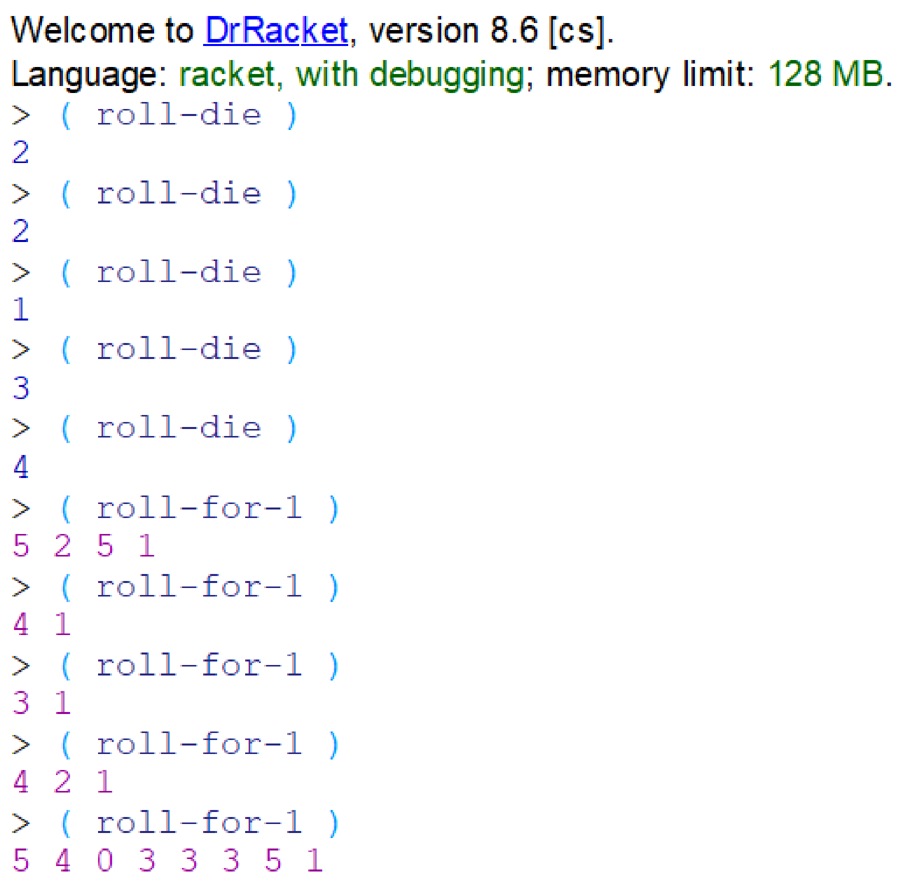
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( tract 100 200 )
> ( tract 200 200 )
>
```

The code for house and tract:

```
#lang racket

; Assignment #2
; 1. Required to make basic images
; 2. Generating random colors
; 3. Defining house floor
; 4. Defining width,and height of house. Defining all three floors and roof. Then putting it all to
; 5. Defining new six houses to make tract


; 1.
( require 2htdp/image )

; 2.
( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )
( define ( rgb-value ) ( random 256 ) )

; 3.
( define ( floor side1 side2 )
( rectangle side1 side2 "solid" ( random-color ) )
)

; 4.
( define ( house side1 side2 )
( define side1-of-house ( / side1 3 ) )
( define side2-of-house ( / side2 3 ) )
( define floor1 ( floor side1-of-house side2-of-house ) )
( define floor2 ( floor side1-of-house side2-of-house ) )
( define floor3 ( floor side1-of-house side2-of-house ) )
( define roof ( triangle side1-of-house "solid" "grey" ) )
( define house1 ( above roof floor1 floor2 floor3 ) )
house1
)

; 5.
( define ( tract side1 side2 )
( define side1-of-house ( / side1 3 ) )
( define side2-of-house ( / side2 3 ) )
( define floor1 ( floor side1-of-house side2-of-house ) )
( define floor2 ( floor side1-of-house side2-of-house ) )
( define floor3 ( floor side1-of-house side2-of-house ) )
( define roof ( triangle side1-of-house "solid" "grey" ) )
( define houseA ( above roof floor1 floor2 floor3 ) )
( define houseB ( above roof floor3 floor1 floor2 ) )
( define houseC ( above roof floor2 floor3 floor1 ) )
( define houseD ( above roof floor1 floor3 floor2 ) )
( define houseE ( above roof floor2 floor1 floor3 ) )
( define houseF ( above roof floor3 floor2 floor1 ) )
( define block ( square 5 "solid" "white" ) )
( define tract1 ( beside houseA block houseB block houseC block houseD block houseE block houseF ) )
tract1
)
```

Task 2: Dice

Demo for Dice:



```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( roll-die )
2
> ( roll-die )
2
> ( roll-die )
1
> ( roll-die )
3
> ( roll-die )
4
> ( roll-for-1 )
5 2 5 1
> ( roll-for-1 )
4 1
> ( roll-for-1 )
3 1
> ( roll-for-1 )
4 2 1
> ( roll-for-1 )
5 4 0 3 3 3 5 1
```

```
> ( roll-for-11 )
5 2 3 5 2 2 4 1 2 3 3 0 1 1
> ( roll-for-11 )
0 1 0 4 5 0 3 0 4 3 2 2 4 4 5 2 2 4 1 1
> ( roll-for-11 )
2 4 2 2 2 5 2 2 5 2 4 3 5 2 0 1 5 3 3 3 4 1 1
> ( roll-for-11 )
5 2 5 4 4 2 5 1 0 1 0 3 3 0 4 0 2 3 5 4 1 0 3 3 4 4 0 5 1 4
0 4 3 3 4 5 4 4 5 5 5 0 4 5 2 2 2 2 0 4 4 2 5 1 4 3 2 1 1
> ( roll-for-11 )
2 1 1
> ( roll-for-odd-even-odd )
5 4 4 3 5 2 3
> ( roll-for-odd-even-odd )
2 0 1 3 3 5 5 2 2 2 4 3 3 3 4 1
> ( roll-for-odd-even-odd )
3 3 1 1 1 0 0 0 3 4 3
> ( roll-for-odd-even-odd )
0 5 5 1 1 0 1
> ( roll-for-odd-even-odd )
5 0 2 4 4 0 1 3 1 2 4 3 5 5 4 5
> ( roll-two-dice-for-a-lucky-pair )
( 3 0 ) ( 0 4 ) ( 0 4 ) ( 1 3 ) ( 0 1 ) ( 0 2 ) ( 2 3 ) ( 0
2 ) ( 2 1 ) ( 0 3 ) ( 0 0 ) #t
> ( roll-two-dice-for-a-lucky-pair )
( 4 4 ) #t
> ( roll-two-dice-for-a-lucky-pair )
( 2 5 ) #t
> ( roll-two-dice-for-a-lucky-pair )
( 3 5 ) ( 5 1 ) ( 3 1 ) ( 0 3 ) ( 5 0 ) ( 3 3 ) #t
> ( roll-two-dice-for-a-lucky-pair )
( 3 5 ) ( 3 5 ) ( 1 3 ) ( 4 4 ) #t
>
```

The code for dice:

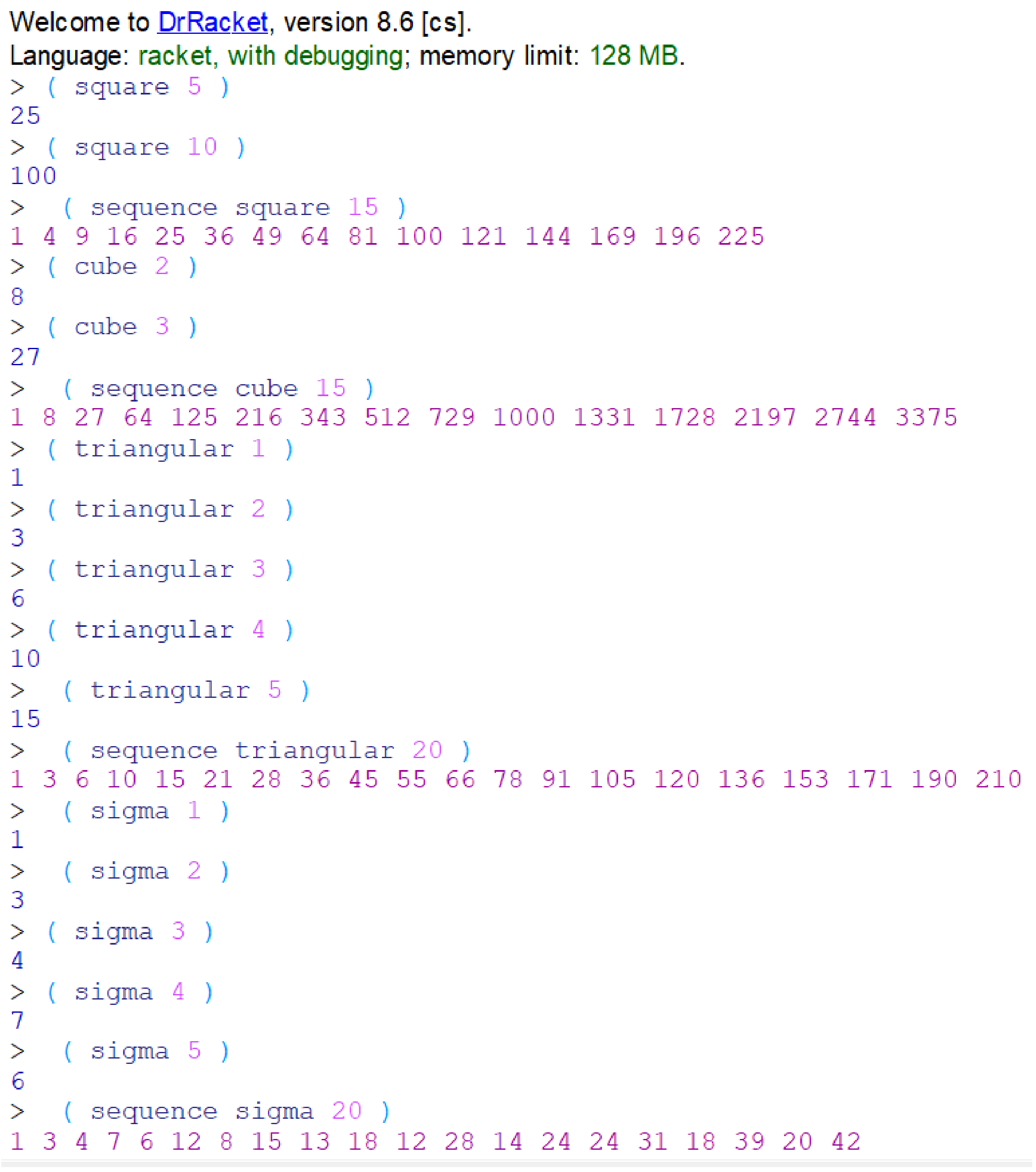
```
#lang racket

;1. Simulates the roll of a standard die
;2. Simulates the roll of a standard die until a 1 turns up
;3. Simulates the roll of a standard die until two consecutive 1s turn up
;4. simulates the roll of a standard die until consecutive values
;which are odd then even then odd turn up
;5.simulates the roll of two standard dice until
;either the sum of seven, the sum of eleven, or a double turns up


;1.
( define ( roll-die ) ( random 6 ) )


;2.
( define ( roll-for-1 )
   ( define outcome ( roll-die ) )
   ( display outcome ) ( display " " )
   ( cond
      ( ( not ( eq? outcome 1 ) )
        ( roll-for-1 )
        )
      )
   )
```

```
;3.
( define ( roll-for-11 )
   ( roll-for-1 )
   ( define outcome ( roll-die ) )
   ( display outcome ) ( display " " )
   ( cond
      ( ( not ( eq? outcome 1 ) )
        ( roll-for-11 )
        )
      )
   )

;4.
( define ( roll-for-roll )
   ( define outcome ( roll-die ) )
   ( display outcome ) ( display " " )
   ( cond
      ( ( odd? outcome )
        ( roll-for-roll )
        )
      )
   )



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

```
;5.
( define ( roll-two-dice-for-a-lucky-pair )
   ( define outcome1 ( roll-die ) )
   ( define outcome2 ( roll-die ) )
   ( display "( " ) ( display outcome1 ) ( display " " )
   ( display outcome2 ) ( display " ) " )
   ( cond
      ( ( eq? outcome1 outcome2 ) )
      ( else
        ( cond
           ( ( eq? ( + outcome1 outcome2 ) 7 ) )
           ( else
             ( cond
                ( ( eq? ( + outcome1 outcome2 ) 11 ) )
                  ( else
                    ( cond
                     (( roll-two-dice-for-a-lucky-pair )))

                  ) ) )
           )
        )
      )
   )
```

Task 3: Number Sequences:

Demo for Number Sequences:



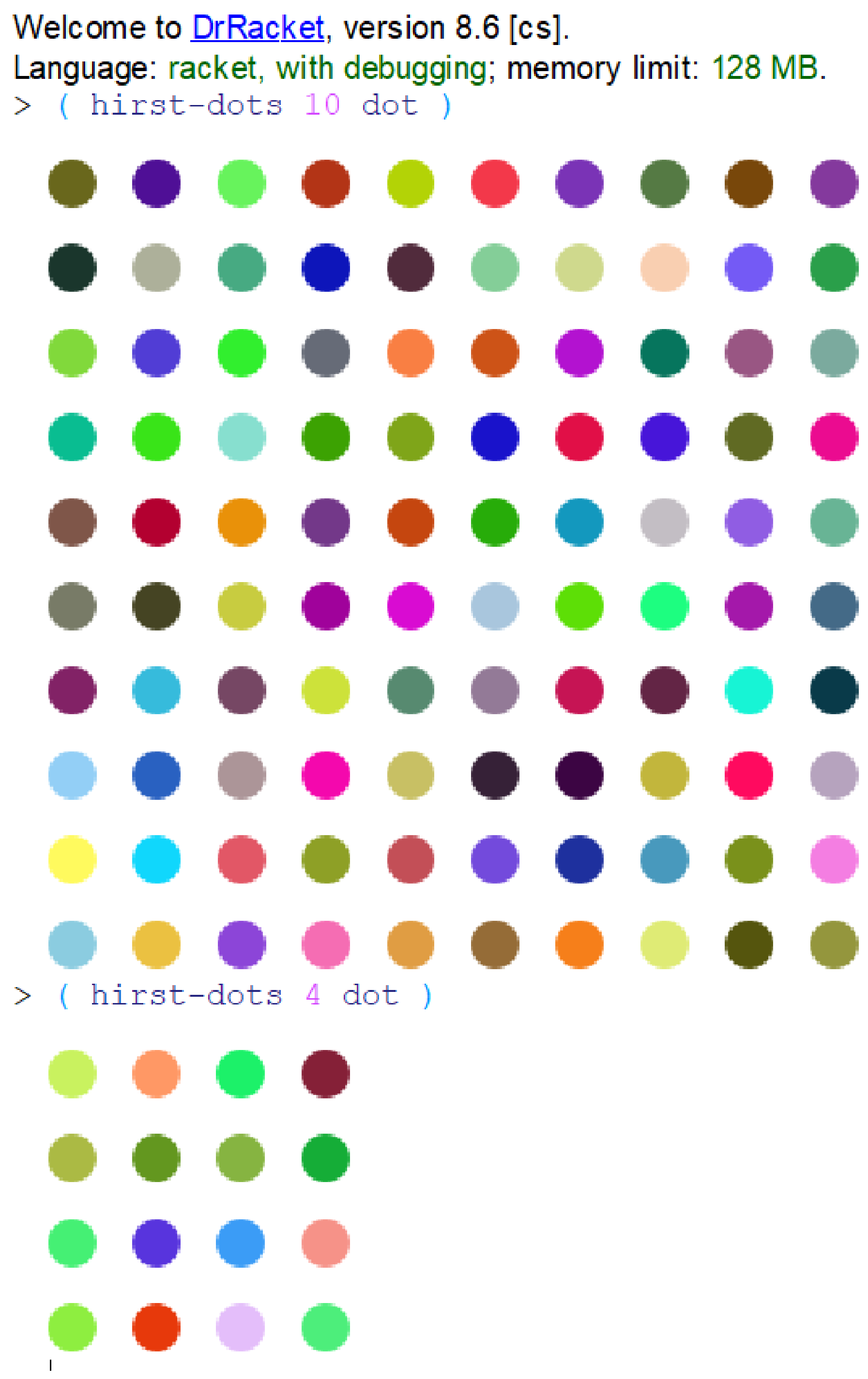
```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( square 5 )
25
> ( square 10 )
100
>  ( sequence square 15 )
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225
> ( cube 2 )
8
> ( cube 3 )
27
>  ( sequence cube 15 )
1 8 27 64 125 216 343 512 729 1000 1331 1728 2197 2744 3375
> ( 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
>  ( sigma 1 )
1
>  ( sigma 2 )
3
> ( sigma 3 )
4
> ( sigma 4 )
7
>  ( sigma 5 )
6
>  ( sequence sigma 20 )
1 3 4 7 6 12 8 15 13 18 12 28 14 24 24 31 18 39 20 42
```

The code for Number Sequences:

```
( 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 " " )
)
)
)



( define ( triangular n )
   ( / ( * n ( + n 1 ) ) 2 )
   )


 (define ( sigma n )
 ( sigma-number n n )
)

( define ( sigma-number m n )
   ( cond
      ( ( = n 1 ) 1)
      ( else
        ( cond
           ( ( = ( remainder m n ) 0 )
             ( + ( sigma-number m ( - n 1 ) ) n ) )
           ( else
             ( sigma-number m ( - n 1 ))
             )
           )
        )
      )
   )
```

Task 4: Hirst Dots

Demo of Hirst Dots:



```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( hirst-dots 10 dot )
```

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

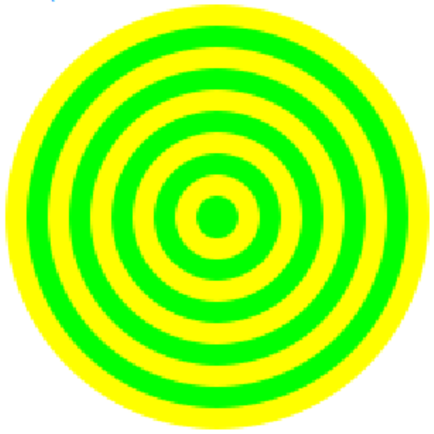
The code for Hirst Dots:

```
#lang racket

;1. Generate a random rgb-value for dot type.
;2. Generate a row of dots of specified length and type
;3. Generate a rectangle of dots of specified row count,
;column count, and dot type
;4. Generate a square of dots of specified side length and dot type

;1.
(require 2htdp/image)

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

( define ( dot ) ( circle 10 "solid" ( random-color ) ) )

( define block ( square 15 "solid" "white" ) )


;2.
( define ( row-of-dots n dot )
( cond
((= n 0)
empty-image
)
((> n 0)
( beside ( row-of-dots ( - n 1 ) dot ) block ( dot ) )
)
)
)



;3.
( define ( rectangle-of-dots r c dot )
( cond
((= r 0)
empty-image
)
((> r 0)
( above
( rectangle-of-dots ( - r 1 ) c dot ) block ( row-of-dots c dot ) )
)
)
)

;4.
( define ( hirst-dots n dot )
( rectangle-of-dots n n dot )
   )
```

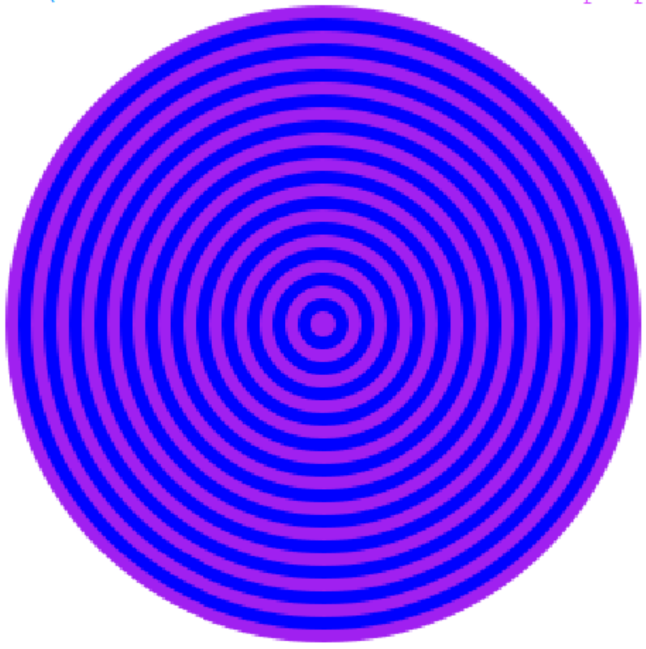
Task 5: Channeling Frank Stella:

Demo of Stella Circle:

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( nested-circles 100 10 "yellow" "green" )
```



```
> ( nested-circles 150 25 "blue" "purple" )
```

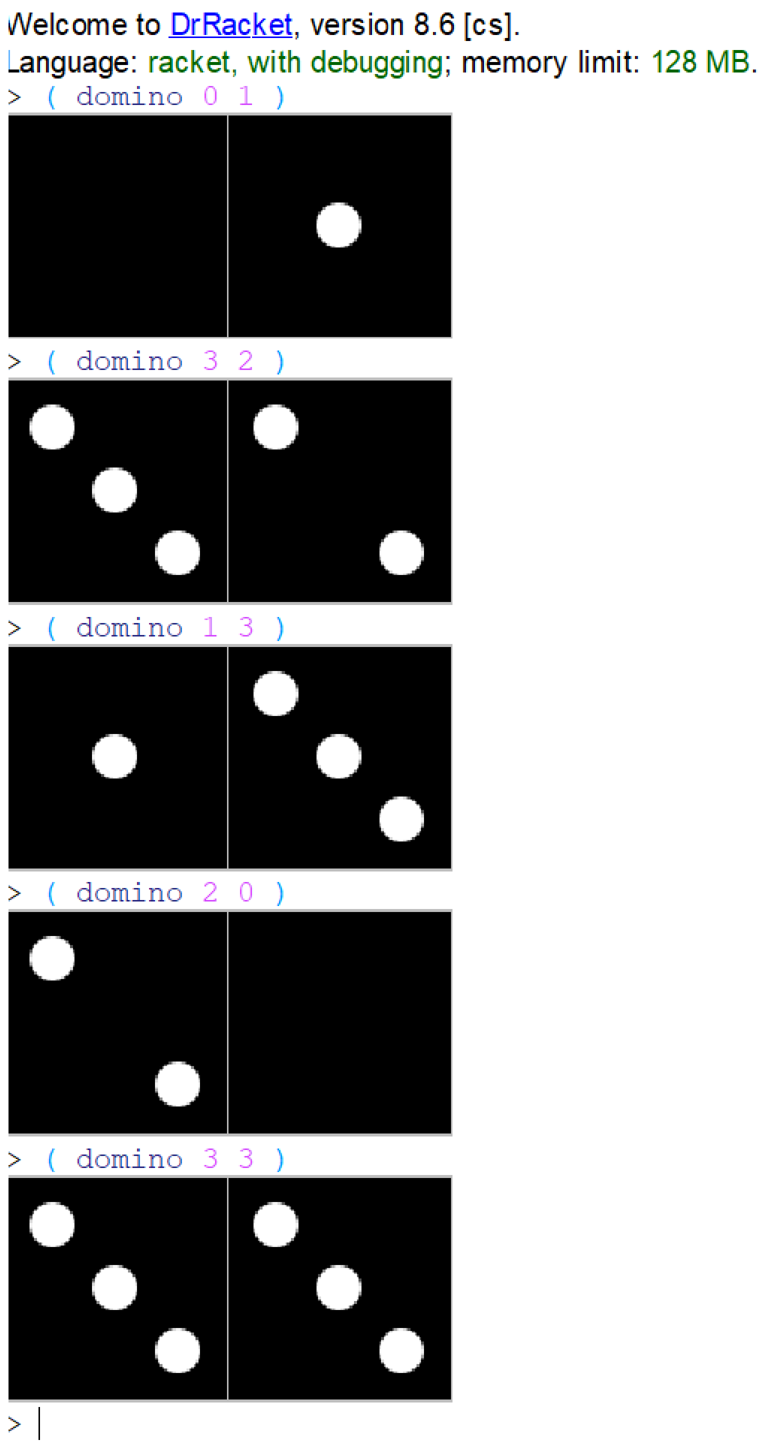


Code for Stella Circle:

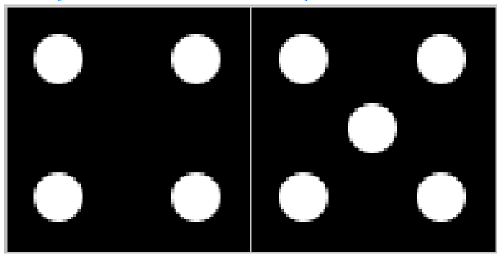
```
#lang racket
( require 2htdp/image )
( define ( nested-circles side count color1 color2 )
( define delta ( / side count ) )
( paint-nested-circles 1 count delta color1 color2 )
)
( define ( paint-nested-circles from to delta color1 color2 )
( define radius ( * from delta ) )
( cond
( ( = from to )
( if ( even? from )
( circle radius "solid" color1 )
( circle radius "solid" color2 )
)
)
( ( < from to )
( if ( even? from )
( overlay
( circle radius "solid" color1 )
( paint-nested-circles ( + from 1 ) to delta color1 color2 )
)
( overlay
( circle radius "solid" color2 )
( paint-nested-circles  ( + from 1 ) to delta color1 color2 )
)
)
)
)
)
```

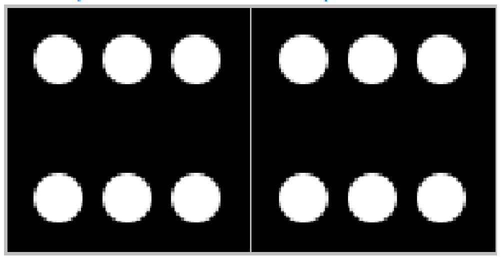
Task 6: Domino:

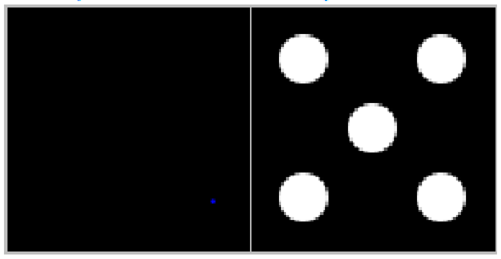
Demo for Dominos:

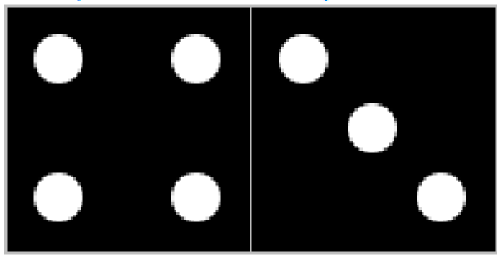


```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( domino 0 1 )
> ( domino 3 2 )
> ( domino 1 3 )
> ( domino 2 0 )
> ( domino 3 3 )
> 
```

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


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


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


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


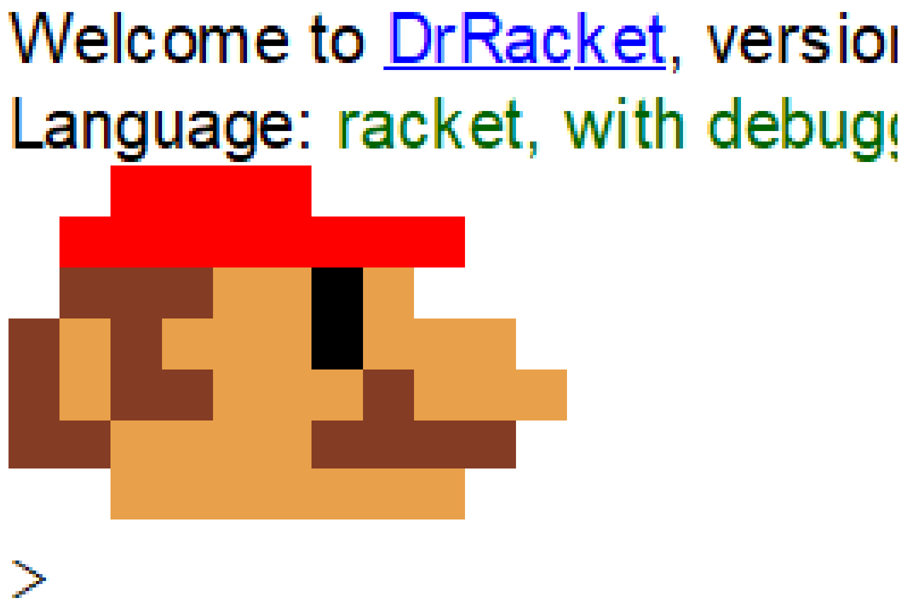
```
>
```

Code for Dominos:

```
#lang racket
( 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 basic-tile4  )))


( 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 )
)
)
```

Task 7: Creation

Creation (image)(Mario Head):



```
Welcome to DrRacket, versio
Language: racket, with debug
```

```
>
```

Creation (code):

```
 #lang racket

 ( require 2htdp/image )

 (overlay/xy(overlay/xy(overlay/xy(overlay/xy
             (overlay/xy (overlay/xy( rectangle 70 10 "solid" ( make-color 233 160 74) )
            -20
            -10
            ( beside(rectangle 20 10 "solid" "brown")
                    (rectangle 40 10 "solid" ( make-color 233 160 74))
                    (rectangle 40 10 "solid" "brown")
            ))
              0
            -10
            (beside(rectangle 10 10 "solid" "brown")
                   (rectangle 10 10 "solid" ( make-color 233 160 74))
                   (rectangle 20 10 "solid" "brown")
                   (rectangle 30 10 "solid" ( make-color 233 160 74))
                   (rectangle 10 10 "solid" "brown")
                   (rectangle 30 10 "solid" ( make-color 233 160 74))
             ))

    0
  -10
   (beside(rectangle 10 10 "solid" "brown")
          (rectangle 10 10 "solid" ( make-color 233 160 74))
          (rectangle 10 10 "solid" "brown")
          (rectangle 30 10 "solid" ( make-color 233 160 74))
          (rectangle 10 10 "solid" "black")
          (rectangle 30 10 "solid" ( make-color 233 160 74))
          ))
    10
   -10

   (beside(rectangle 30 10 "solid" "brown")
   (rectangle 20 10 "solid" ( make-color 233 160 74))
   (rectangle 10 10 "solid" "black")
   (rectangle 10 10 "solid" ( make-color 233 160 74))
   ))

 10
-10
   (rectangle 80 10 "solid" "red")
   )
   20
  -10
   (rectangle 40 10 "solid" "red")
   )
```