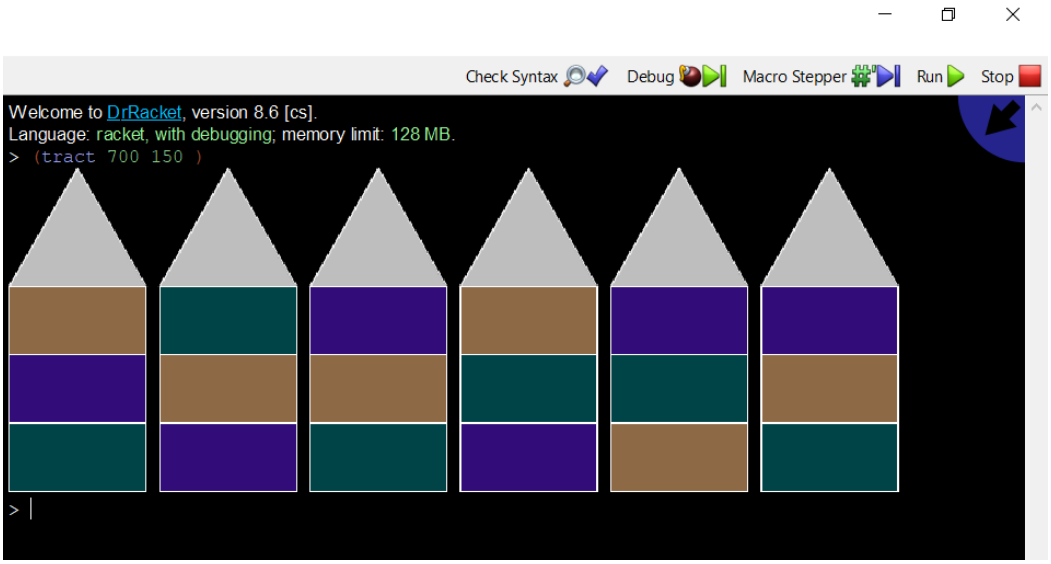
Racket Programming Assignment 2: Racket Functions and Recursion

Learning Abstract: This assignment contains programs which generate images using the 2htdp/images library, mostly using but not restricted to recursion.

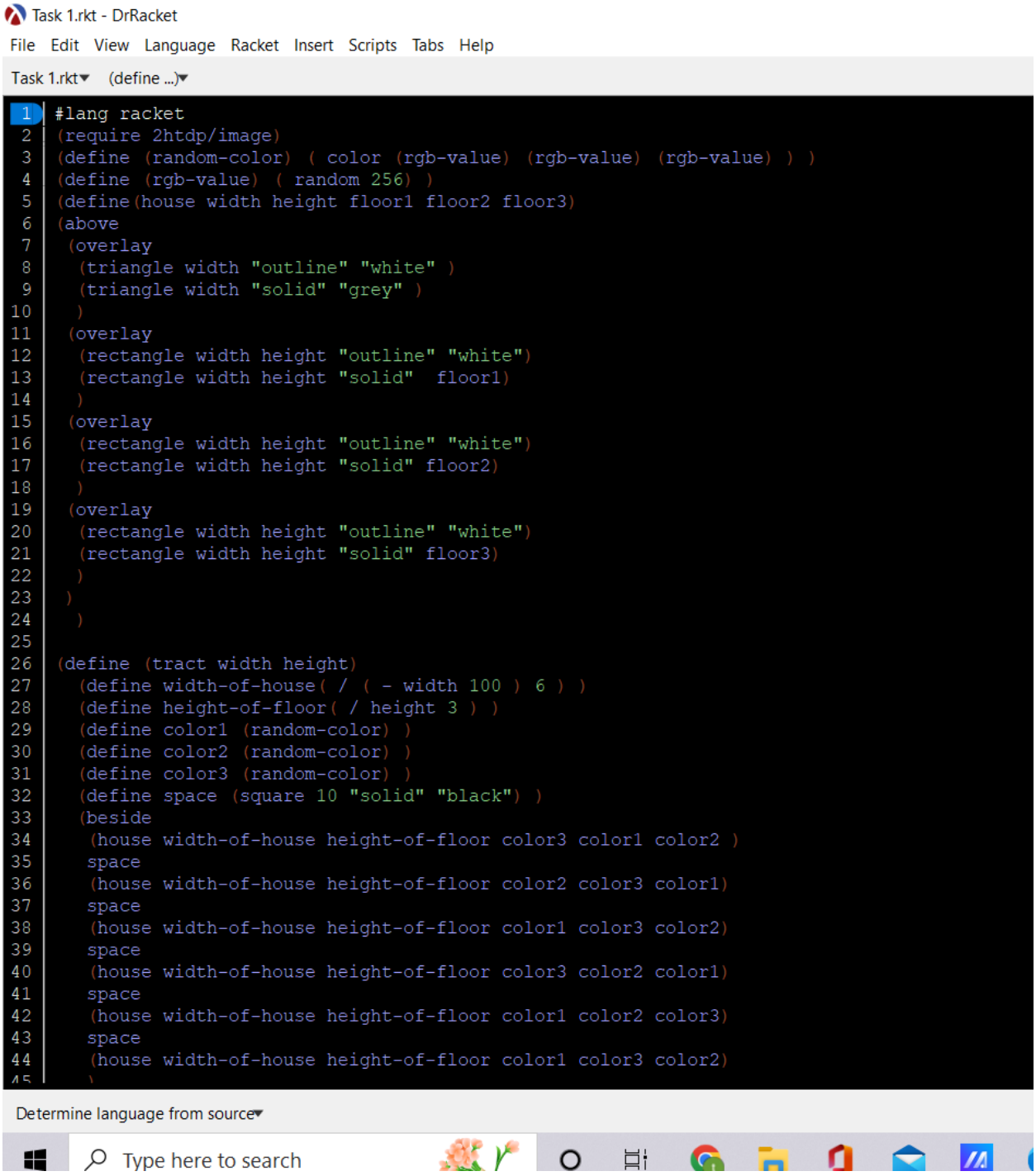
Task 1 Colorful Permutations of Tract Houses:

Demo for House:

Demo for Tract:

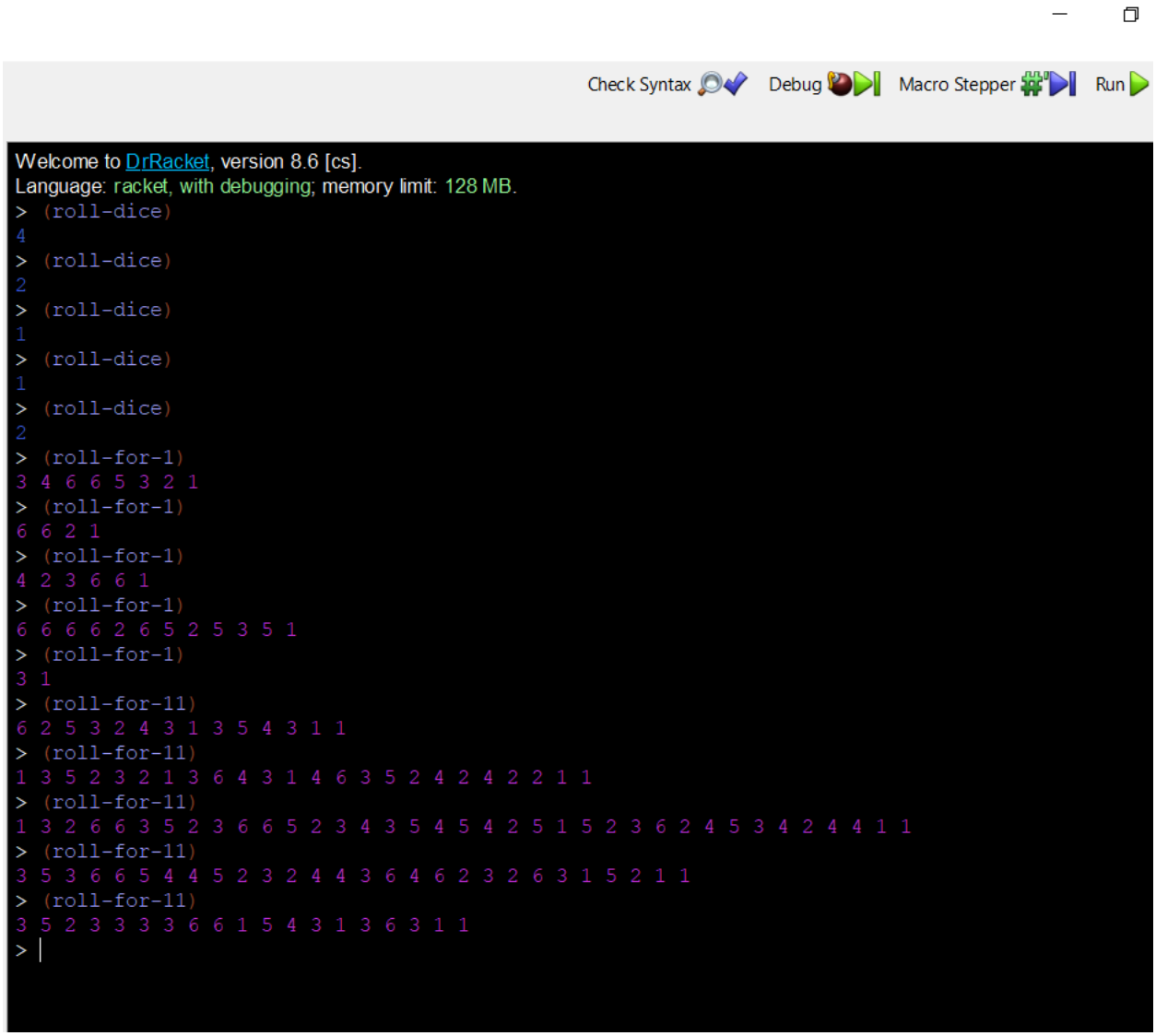


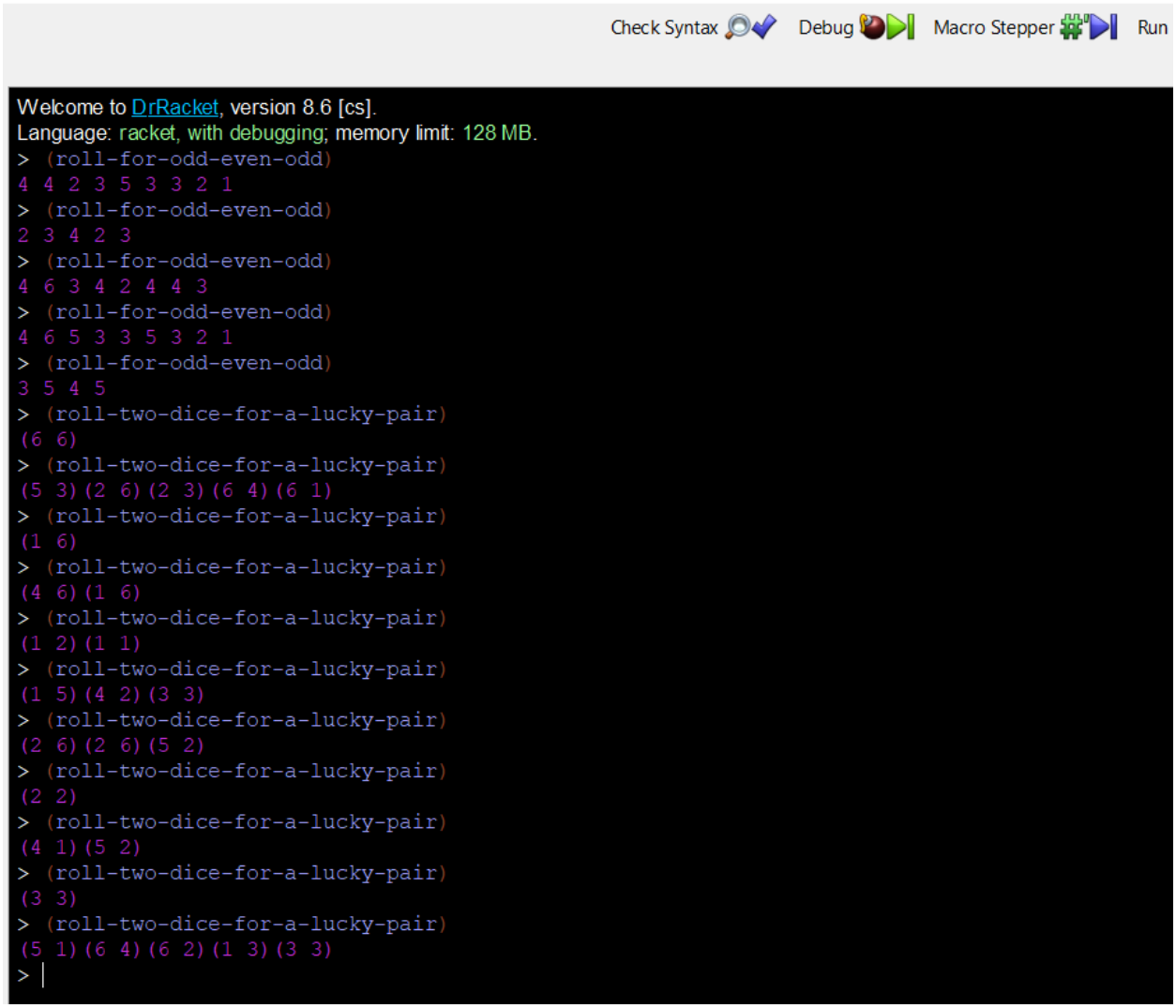
Code:



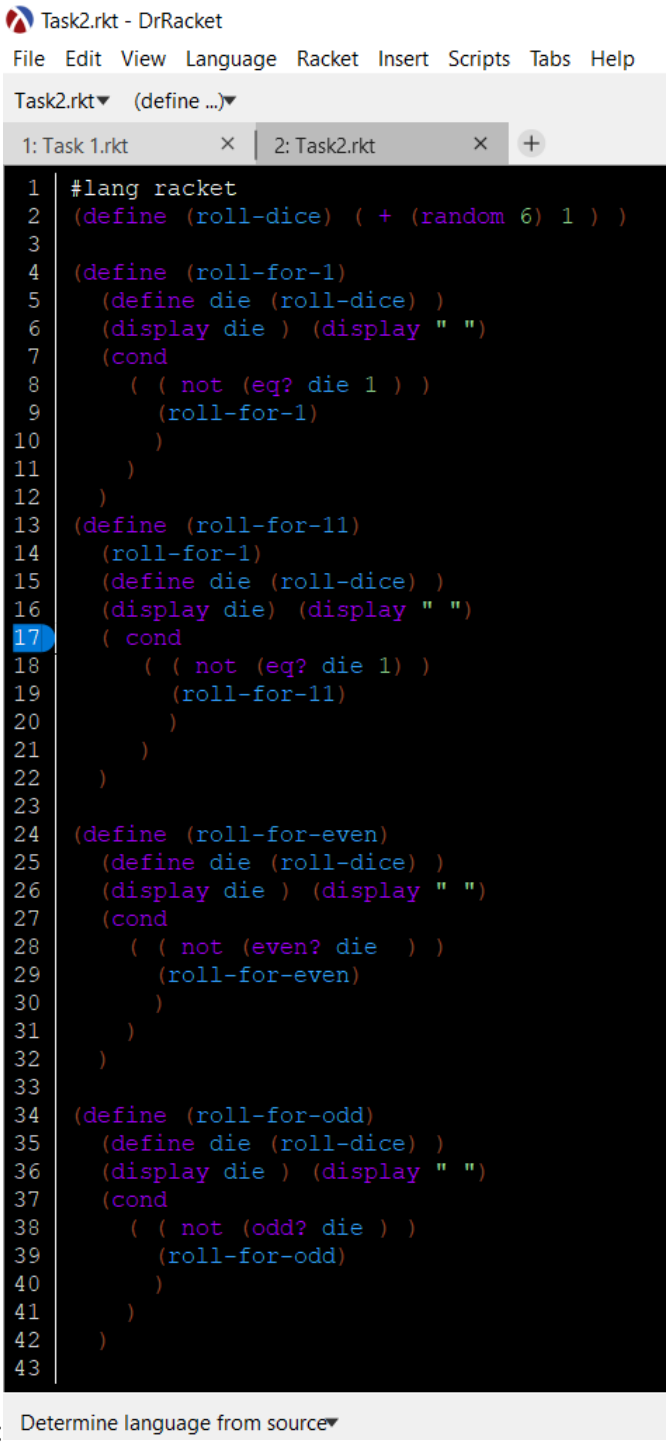
Task 2 Dice

Demos:


Check Syntax
Debug
Macro Stepper
Run
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (roll-dice)
4
> (roll-dice)
2
> (roll-dice)
1
> (roll-dice)
1
> (roll-dice)
2
> (roll-for-1)
3 4 6 6 5 3 2 1
> (roll-for-1)
6 6 2 1
> (roll-for-1)
4 2 3 6 6 1
> (roll-for-1)
6 6 6 6 2 6 5 2 5 3 5 1
> (roll-for-1)
3 1
> (roll-for-11)
6 2 5 3 2 4 3 1 3 5 4 3 1 1
> (roll-for-11)
1 3 5 2 3 2 1 3 6 4 3 1 4 6 3 5 2 4 2 4 2 2 1 1
> (roll-for-11)
1 3 2 6 6 3 5 2 3 6 6 5 2 3 4 3 5 4 5 4 2 5 1 5 2 3 6 2 4 5 3 4 2 4 4 1 1
> (roll-for-11)
3 5 3 6 6 5 4 4 5 2 3 2 4 4 3 6 4 6 2 3 2 6 3 1 5 2 1 1
> (roll-for-11)
3 5 2 3 3 3 3 6 6 1 5 4 3 1 3 6 3 1 1
>


Check Syntax
Debug
Macro Stepper
Run
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (roll-for-odd-even-odd)
4 4 2 3 5 3 3 2 1
> (roll-for-odd-even-odd)
2 3 4 2 3
> (roll-for-odd-even-odd)
4 6 3 4 2 4 4 3
> (roll-for-odd-even-odd)
4 6 5 3 3 5 3 2 1
> (roll-for-odd-even-odd)
3 5 4 5
> (roll-two-dice-for-a-lucky-pair)
(6 6)
> (roll-two-dice-for-a-lucky-pair)
(5 3) (2 6) (2 3) (6 4) (6 1)
> (roll-two-dice-for-a-lucky-pair)
(1 6)
> (roll-two-dice-for-a-lucky-pair)
(4 6) (1 6)
> (roll-two-dice-for-a-lucky-pair)
(1 2) (1 1)
> (roll-two-dice-for-a-lucky-pair)
(1 5) (4 2) (3 3)
> (roll-two-dice-for-a-lucky-pair)
(2 6) (2 6) (5 2)
> (roll-two-dice-for-a-lucky-pair)
(2 2)
> (roll-two-dice-for-a-lucky-pair)
(4 1) (5 2)
> (roll-two-dice-for-a-lucky-pair)
(3 3)
> (roll-two-dice-for-a-lucky-pair)
(5 1) (6 4) (6 2) (1 3) (3 3)
>

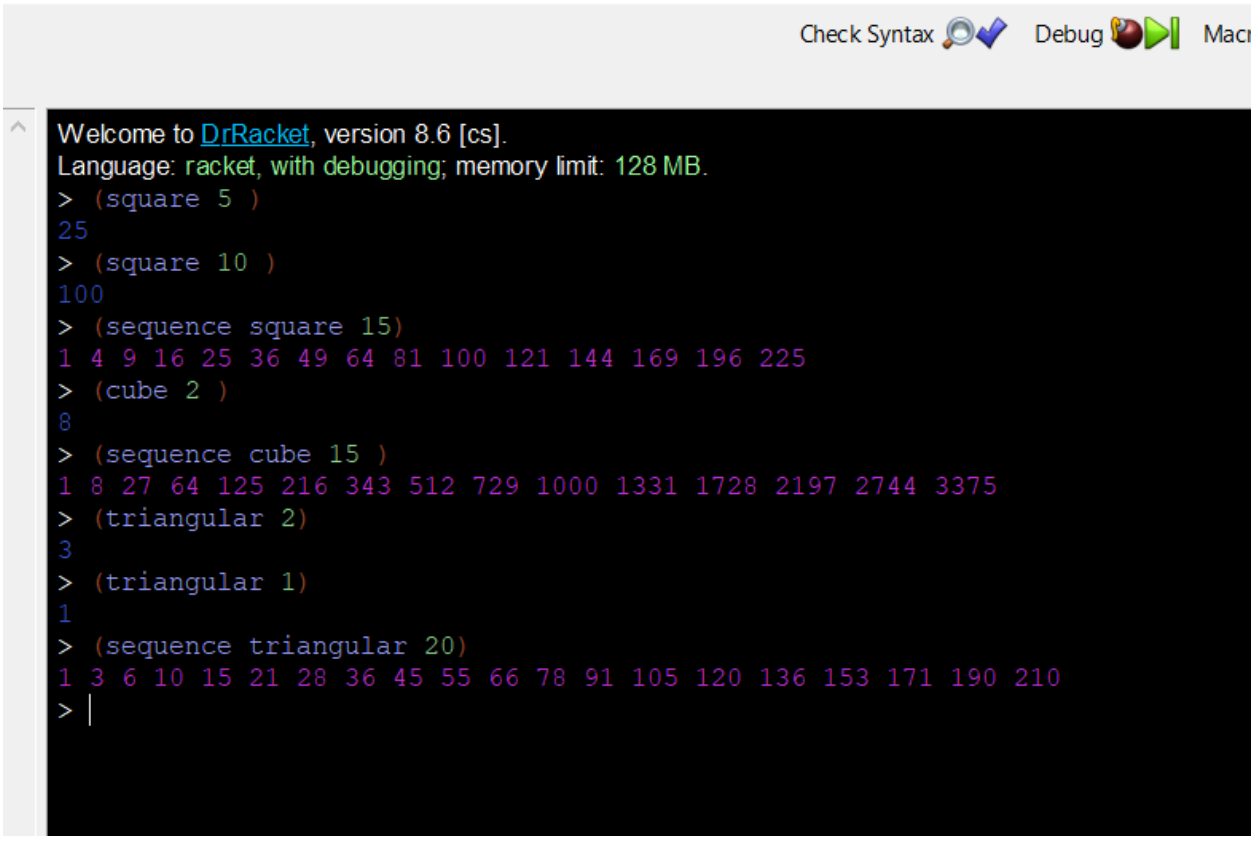
Code:


Task2.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help
Task2.rkt (define ...)
1: Task 1.rkt
2: Task2.rkt
#lang racket
(define (roll-dice) (+ (random 6) 1))
(define (roll-for-1)
(define die (roll-dice))
(display die) (display " ")
(cond
((not (eq? die 1))
(roll-for-1)
)
)
)
(define (roll-for-11)
(roll-for-1)
(define die (roll-dice))
(display die) (display " ")
(cond
((not (eq? die 1))
(roll-for-11)
)
)
)
(define (roll-for-even)
(define die (roll-dice))
(display die) (display " ")
(cond
((not (even? die))
(roll-for-even)
)
)
)
(define (roll-for-odd)
(define die (roll-dice))
(display die) (display " ")
(cond
((not (odd? die))
(roll-for-odd)
)
)
)
Determine language from source

```
43
44 (define (roll-for-odd-even-odd) (roll-for-odd) (roll-for-even) (roll-for-odd) )
45
46 (define (roll-two-dice-for-a-lucky-pair)
47   (define die1 (roll-dice))
48   (define die2 (roll-dice))
49   (define total-7 ( =(+ die1 die2 ) 7 ) )
50   (define total-11 (= (+ die1 die2 ) 11) )
51   (define pairs ( = die1 die2 ) )
52   (display "(")
53   (display die1)
54   (display " ")
55   (display die2)
56   (display ")")
57   (cond
58     ( (not (or total-7 total-11 pairs) )
59     (roll-two-dice-for-a-lucky-pair) ) ) )
60
61
62
63
64
```

Task 3 Number Sequences

Demo:

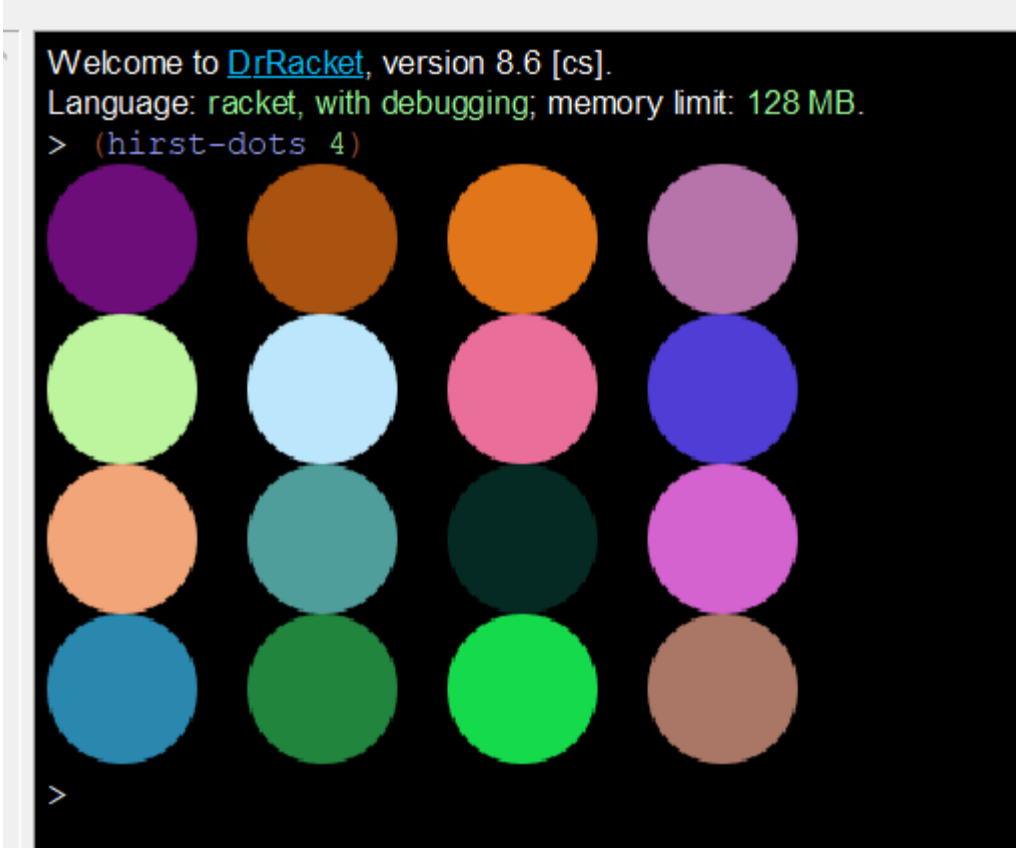


Code:

```
#lang racket

(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 )
  (cond
    ( ( = n 1 ) 1 )
    ( ( > n 1 )
      ( + n ( triangular ( - n 1 ) ) )
      )
    )
  )
```

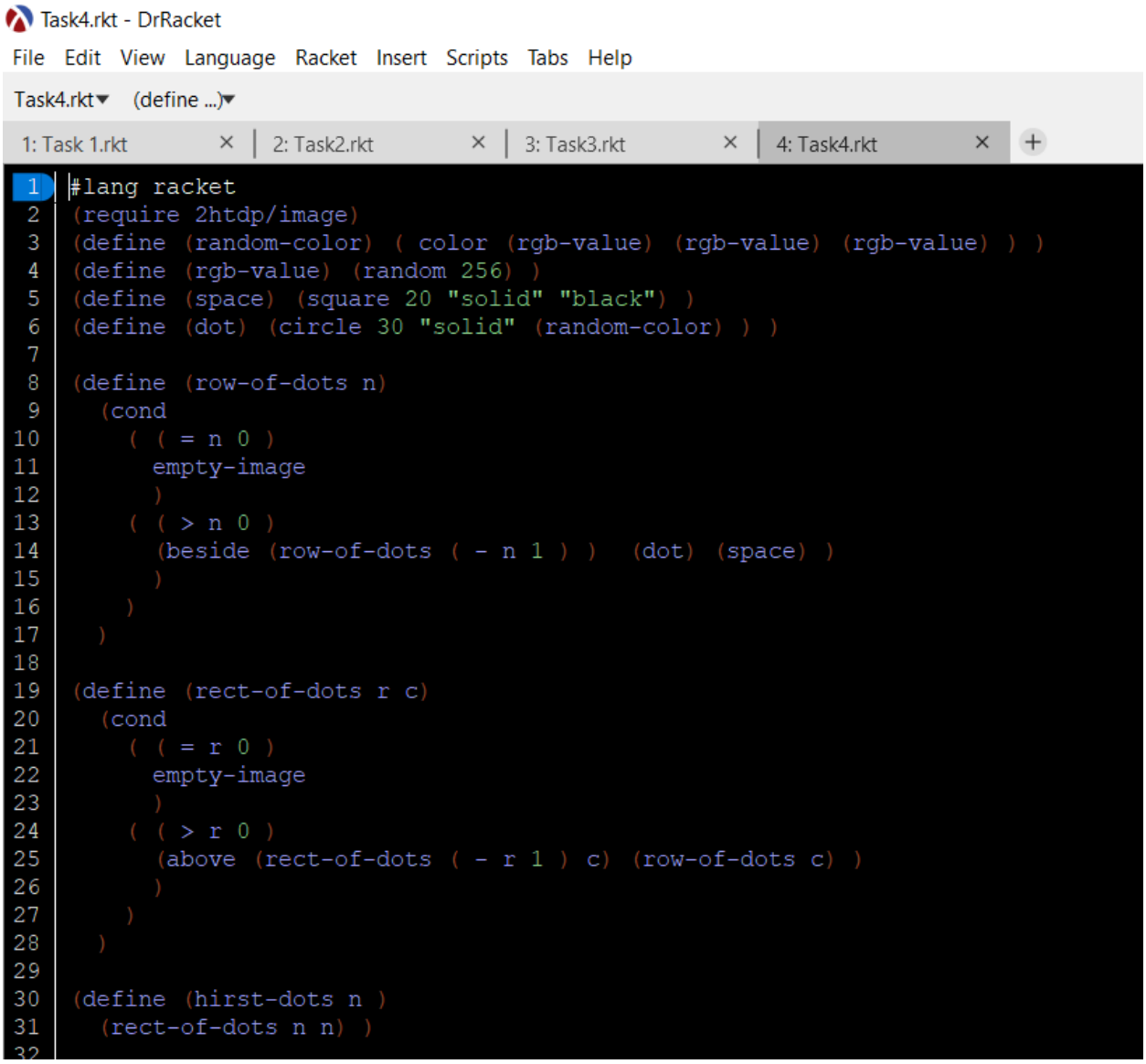
Task 4 Hirst Dots

Demo:


Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (hirst-dots 4)
>
DrRacket


Check Syntax
Debug
Macro Stepper
Run
Stop
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (hirst-dots 10)
>

Code:

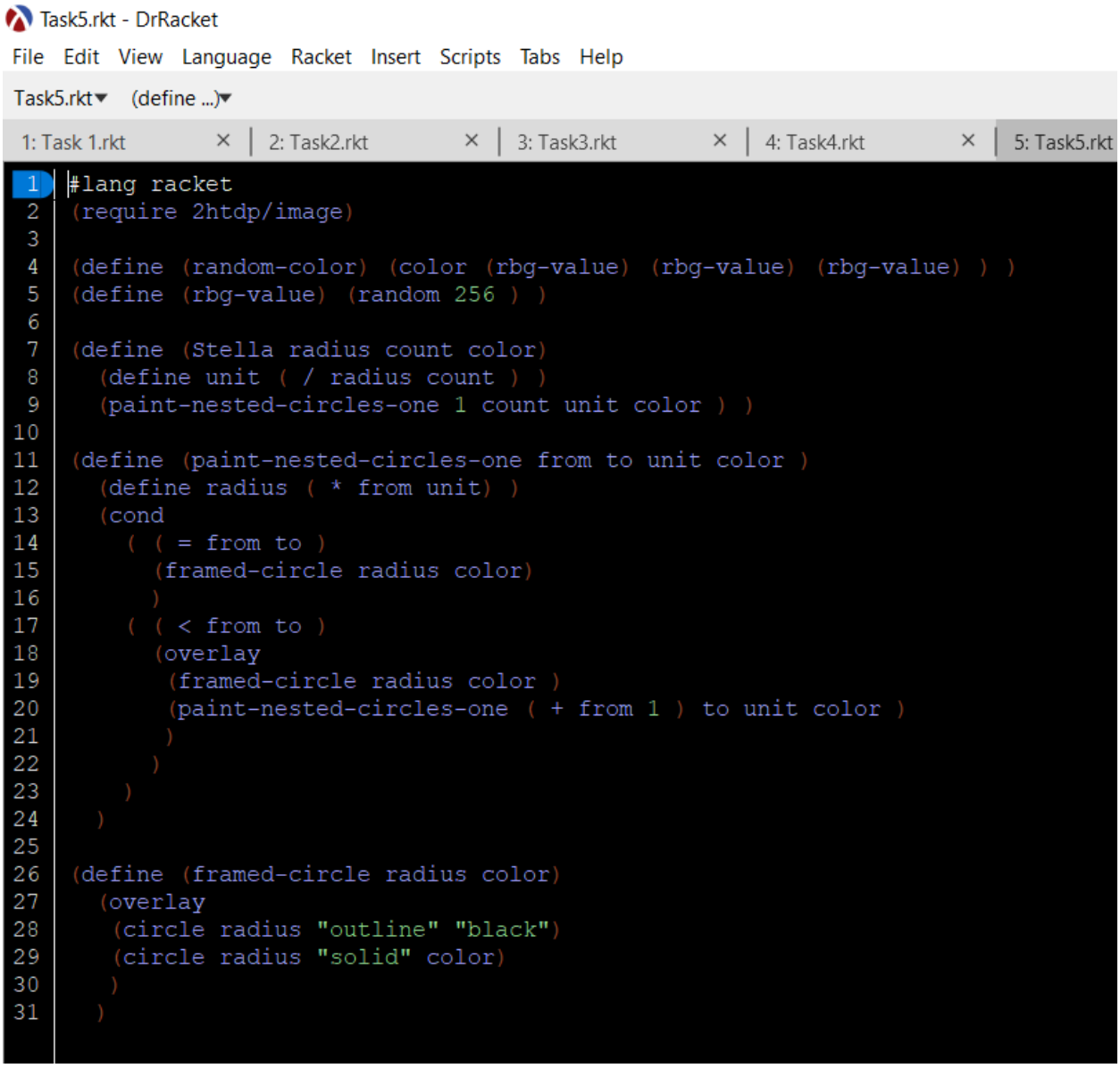


Task 5 Chanelling Frank Stella

Demo:

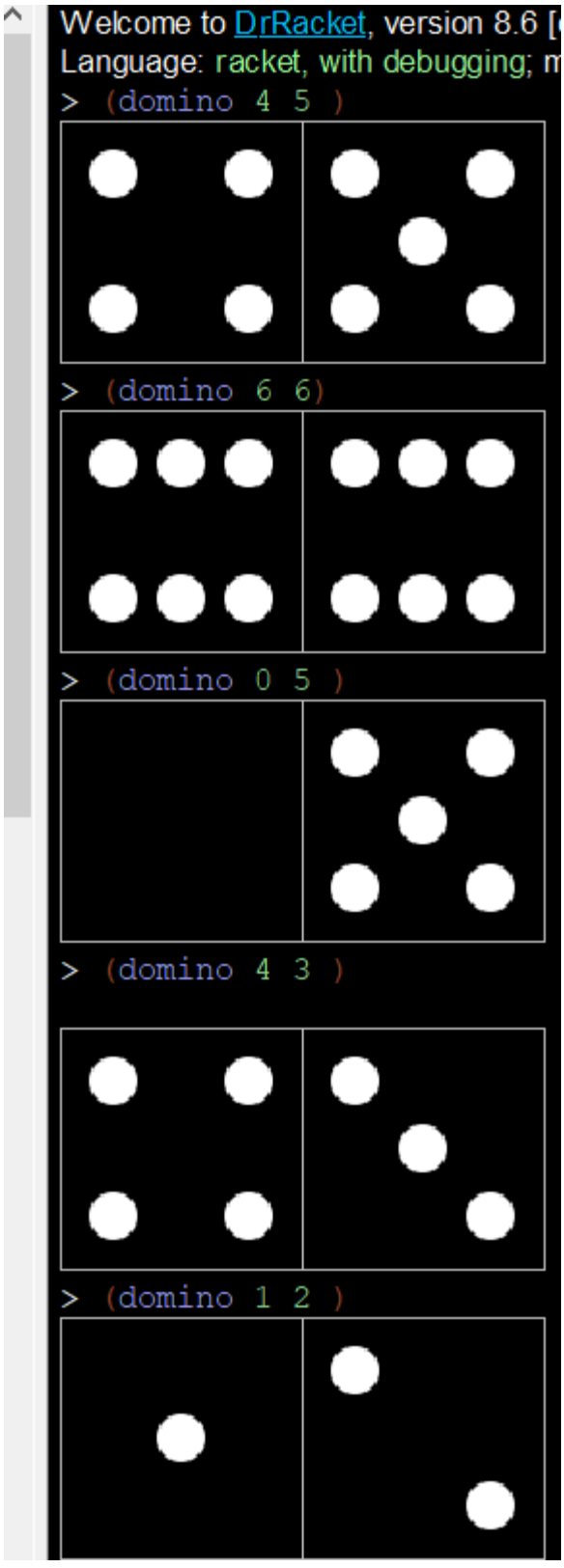

Check Syntax
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (Stella 50 5 "red")
> (Stella 100 10 (random-color))
>

Code:



Task 6 Dominos

Demo:


Welcome to DrRacket, version 8.6 [
Language: racket, with debugging; m
> (domino 4 5)
> (domino 6 6)
> (domino 0 5)
> (domino 4 3)
> (domino 1 2)

Code:

1: Task 1.rkt | 2: Task2.rkt | 3: Task3.rkt | 4: Task4.rkt | 5: Task5.rkt

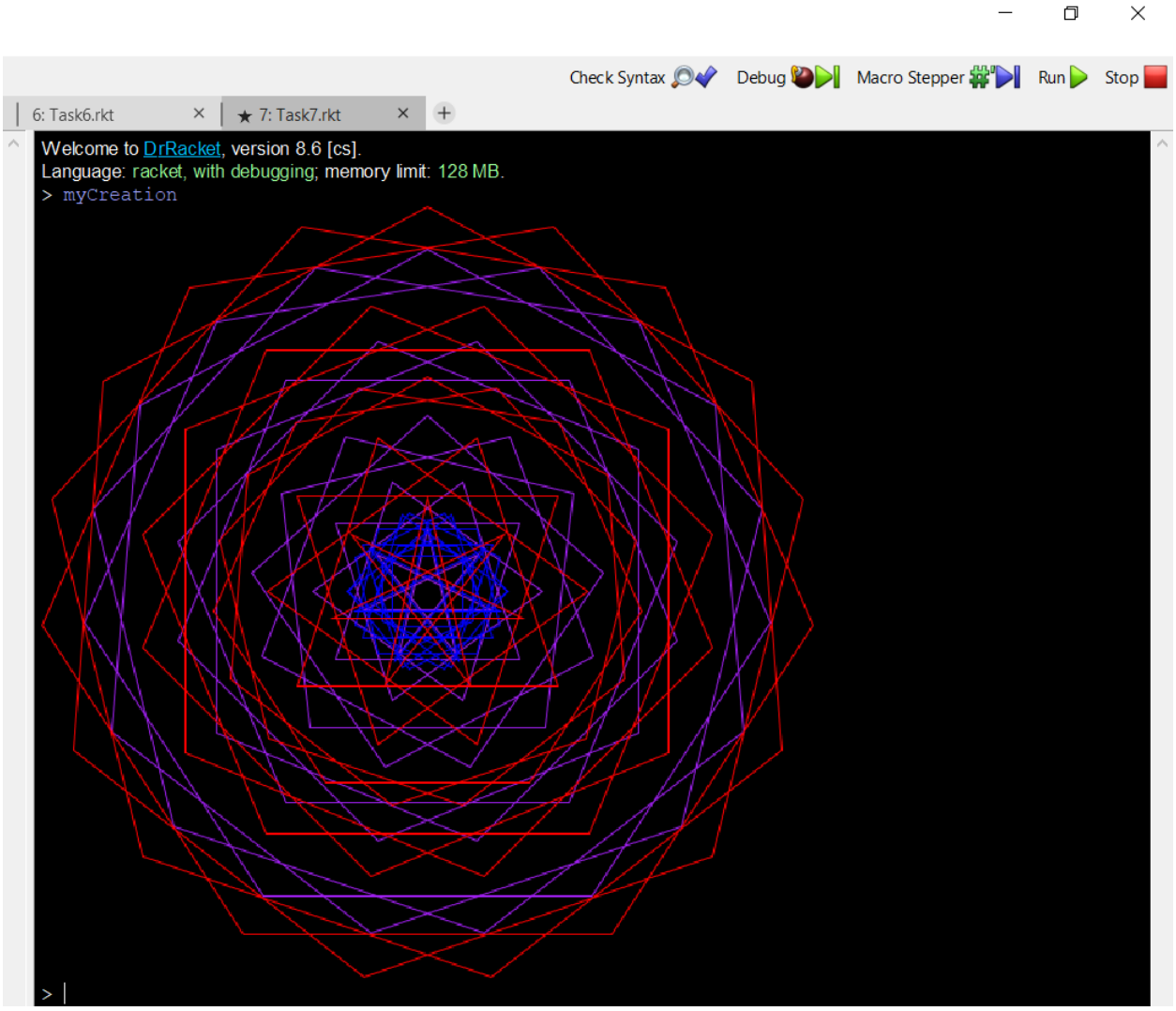
```
#lang racket
(require 2htdp/image)

;problem parameters
;-variables to denote the side of a tile and the dimensions of a pip

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

;Numbers used for offsetting pips from the center of a tile
;- d and nd are used as offsets in the overlay/offset function applications

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

;The blank tile and pip generator
;- Bind one variable to a blank tile and another to a pip

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

;The basic tiles
; Bind one variable to each of the basic tiles

(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)
                                   )
                   )
   )
```

Determine language from source

```
45
46  (define basic-tile5 (overlay (pip) basic-tile4 ) )
47
48  (define basic-tile6
49    (overlay/offset (pip) 0 d
50                    (overlay/offset (pip) 0 nd
51                                    (overlay basic-tile4 basic-tile2)
52                                    )
53                    )
54    )
55
56  ;The framed framed tiles
57  ; - Bind one variable to each of the six framed tiles
58
59  (define frame (square side-of-tile "outline" "gray" ) )
60
61  (define tile0 (overlay frame blank-tile) )
62  (define tile1 (overlay frame basic-tile1) )
63  (define tile2 (overlay frame basic-tile2) )
64  (define tile3 (overlay frame basic-tile3) )
65  (define tile4 (overlay frame basic-tile4 ) )
66  (define tile5 (overlay frame basic-tile5 ) )
67  (define tile6 (overlay frame basic-tile6 ) )
68
69  ;Domino generator
70  ;- Function to generate a domino
71
72  (define (domino a b )
73    (beside (tile a) (tile b ) )
74    )
75
76  (define (tile x )
77    (cond
78      ( ( = x 0 ) tile0 )
79      ( ( = x 1 ) tile1 )
80      ( ( = x 2 ) tile2 )
81      ( ( = x 3 ) tile3 )
82      ( ( = x 4 ) tile4 )
83      ( ( = x 5 ) tile5 )
84      ( ( = x 6 ) tile6 )
85      )
86    )
```

Task 7 My Creation:

Demo:


Check Syntax
Debug
Macro Stepper
Run
Stop
6: Task6.rkt
★ 7: Task7.rkt
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> myCreation
>

Code:

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

(define myCreation
  (underlay
  (star-polygon 40 7 3 "outline" "purple" )
  (star-polygon 50 10 3 "outline" "purple" )
  (star-polygon 60 13 3 "outline" "purple" )
  (star-polygon 70 16 3 "outline" "purple" )
  (star-polygon 80 19 3 "outline" "purple" )
  (star-polygon 50 4 3 "outline" "blue" )
  (star-polygon 45 7 3 "outline" "blue" )
  (star-polygon 35 10 3 "outline" "blue" )
  (star-polygon 25 14 3 "outline" "blue")
  (star-polygon 15 16 3 "outline" "blue" )
  (star-polygon 60 7 3 "outline" "red" )
  (star-polygon 70 10 3 "outline" "red" )
  (star-polygon 80 16 3 "outline" "red" )
  (star-polygon 50 19 3 "outline" "red" )
  (star-polygon 90 19 3 "outline" "red")
   )
  )


```