
Racket Assignment #2: Interactions, Definitions, Applications

What's It All About?

This assignment affords you an opportunity to do some relatively simple Racket programming. You will perform various interactions, write a number of function definitions, and engage in computational problem solving, bits of which feature the reuse of code, imaginative constructions, and the reconfiguration of existing code.

Task 0: Template for Your Solution Document

Craft a template for your solution document for this programming assignment, one that is structured like the one that I am providing along with this assignment.

Task 1: Interactins - Scrap of Tin

Working within the DrRacket PDE, do the following:

1. Mimic the demo presented in the “Arithmetic Expressions” subsection of the “Interactions” section of Racket Lesson #1. Find a way to copy/paste the demo that you created into the appropriate location of your solution document.
2. Mimic the demo presented in the “Solve a Simple Problem (Area of Scrap)” subsection of the “Interactions” section of Racket Lesson #1. Find a way to copy/paste the demo that you created into the appropriate location of your solution document.
3. Mimic the demo presented in the “Rendering an Image of the Problem Situation” subsection of the “Interactions” section of Racket Lesson #1. Find a way to copy/paste the demo that you created into the appropriate location of your solution document.

Task 2: Definitions - Inscribing/Circumscribing Circles/Squares

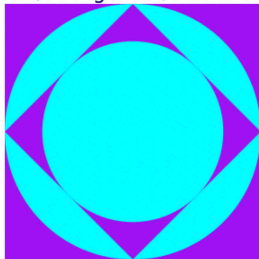
1. Establish a file called `CirclesAndSquares.rkt` in which to define some functions relating to circles and squares.
2. Within the Definitions buffer, write the definition of a function called `cs` to compute the side length of the circumscribing square of a circle whose radius is given by the sole parameter to the function. Thus, for example: $(cs\ 5) \Rightarrow 10$, and $(cs\ 5.5) \Rightarrow 11.0$. Test the function in the Interactions area.
3. Within the Definitions buffer, write the definition of a function called `cc` to compute the radius of the circumscribing circle of a square whose side length is given by the sole parameter to the function. Thus, for example (rounding the results some): $(cc\ 12) \Rightarrow 8.485$, and $(cc\ 17.2) \Rightarrow 12.162$. Test the function in the Interactions area.
4. Within the Definitions buffer, write the definition of a function called `ic` to compute the radius of the inscribing circle of a square whose side length is given by the sole parameter to the function. Thus, for example: $(ic\ 20) \Rightarrow 10.0$, and $(ic\ 25) \Rightarrow 12.5$. Test the function in the Interactions area.

5. Within the Definitions buffer, write the definition of a function called `is` to compute the side length of the inscribing square of a circle whose radius is given by the sole parameter to the function. Thus, for example (rounding the results some): $(\text{is } 15) \Rightarrow 21.213$, and $(\text{is } 70.44) \Rightarrow 99.617$. Test the function in the Interactions area.
6. Within the Definitions buffer, write the definition of a function called `cs-demo` which draws a purple circumscribing square around a blue circle whose radius is given as the sole parameter of the function. Then create a demo of this function by running the following form 3 times in the Interactions buffer: `(cs-demo (random 50 150))`. Arrange for this demo to appear in the appropriate location of your solutions document.
7. Within the Definitions buffer, write the definition of a function called `cc-demo` which draws a purple circumscribing circle around a blue square whose side length is given as the sole parameter of the function. Then create a demo of this function by running the following form 3 times in the Interactions buffer: `(cc-demo (random 50 150))`. Arrange for this demo to appear in the appropriate location of your solutions document.
8. Within the Definitions buffer, write the definition of a function called `ic-demo` which draws a purple inscribing circle within a blue square whose side length is given as the sole parameter of the function. Then create a demo of this function by running the following form 3 times in the Interactions buffer: `(ic-demo (random 50 150))`. Arrange for this demo to appear in the appropriate location of your solutions document.
9. Within the Definitions buffer, write the definition of a function called `is-demo` which draws a purple inscribing square within a blue circle whose radius is given as the sole parameter of the function. Then create a demo of this function by running the following form 3 times in the Interactions buffer: `(is-demo (random 50 150))`. Arrange for this demo to appear in the appropriate location of your solutions document.
10. Copy/paste the definitions from the Definitions area to the appropriate location in your solutions document.

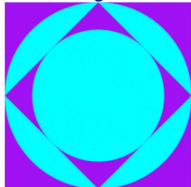
Task 3: Inscribing/Circumscribing Images

1. Within `CirclesAndSquares.rkt`, write the definition of a function called `image-1` taking one parameter that corresponds to the side of a square, which paints an image consisting of a purple square corresponding to the given side, inscribed by a cyan circle, inscribed by a purple square rotated 45 degrees, inscribed by a cyan circle. Please see the accompanying demo for clarification. Furthermore, create your own demo that mimics mine, knowing that the sizes of your images will probably differ from mine, even though, by requirement, you will be typing in the same commands as you see in the demo. Finally, arrange for your demo to appear in the appropriate location in your solutions document. **Constraint: Use appropriate functions from the previous task.**

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



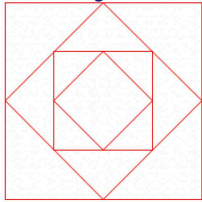
```
> ( image-1 ( random 200 300 ) )
```



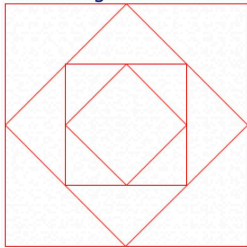
```
> |
```

- Within `CirclesAndSquares.rkt`, write the definition of a function called `image-2` taking one parameter that corresponds to the side of a square, which draws an image consisting of four nested red squares, where the corners of each nested square touch the midpoints of the sides of its immediately containing square. Please see the accompanying demo for clarification. Furthermore, create your own demo that mimics mine, knowing that the sizes of your images will probably differ from mine, even though, by requirement, you will be typing in the same commands as you see in the demo. Finally, arrange for your demo to appear in the appropriate location in your solutions document. **Constraint:** Use appropriate functions from the previous task, and some imaginative constructions.

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



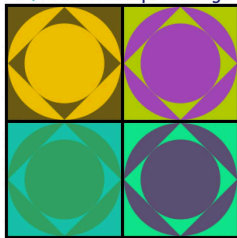
```
> ( image-2 ( random 200 300 ) )
```



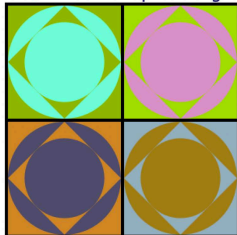
```
>
```

- Within `CirclesAndSquares.rkt`, write the definition of a function called `Warholesque-image` taking one parameter that corresponds to the side of a “square canvas,” which paints a 2x2 checkerboard of randomly colored two-tone images like that generated by the `image-1` function, except for colors, in which black borders of width 4 provide the “finishing touch” for the “checkerboard.” Please see the accompanying demo for clarification. Furthermore, create your own demo that mimics mine, knowing that the colors of your images will differ from mine, even though, by requirement, you will be typing in the same commands as you see in the demo. Finally, arrange for your demo to appear in the appropriate location in your solutions document.

```
> ( warholesque-image 300 )
```



```
> ( warholesque-image 300 )
```




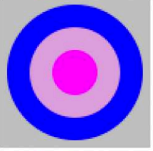
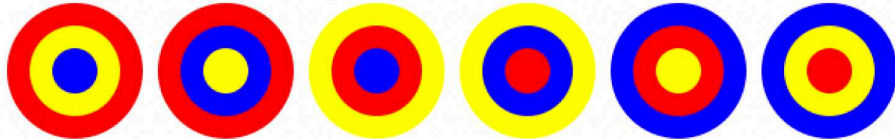
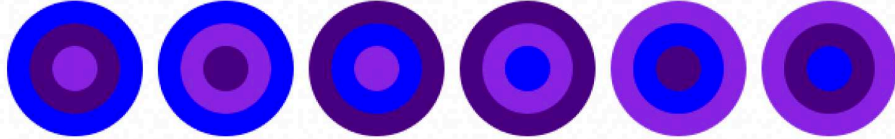
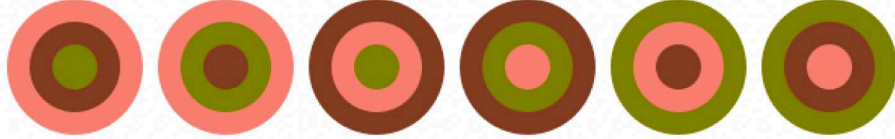
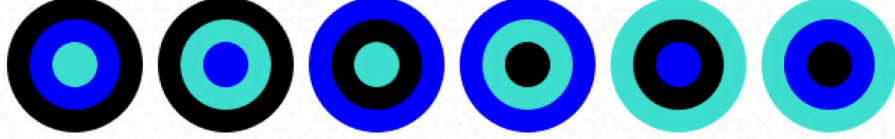
```
>
```

4. Copy/paste the definitions that you wrote for this task to the appropriate location in your solutions document.

Task 4: Permutations of Randomly Colored Stacked Dots

Programming constraint: For this part of your assignment, you are not permitted to use any form of repetition (recursion/iteration) or any form of conditional statement (e.g., if, cond).

Prior commencing with the work on this task, please consider the following demo:

```
Welcome to DrRacket, version 8.1 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( tile "black" "cornflowerblue" "cyan" "teal" )

> ( tile "silver" "blue" "plum" "magenta" )

> ( dots-permutations "red" "yellow" "blue" )

> ( dots-permutations "blue" "indigo" "blueviolet" )

> ( dots-permutations "salmon" "brown" "olive" )

> ( dots-permutations "black" "blue" "turquoise" )

>
```

Establish a file called `permutations.rkt` within which to define the functions required for this task. Then proceed by methodically doing the following sequence of steps:

1. Please study the “Patchwork House” program, and the way it was developed, in Racket Lesson #1. This program is intended to serve as a resource for you to consult as you engage in the programming of the “tiles” decorated with randomly colored stacks of dots that are featured in this task.
2. Write a program called `tile` which takes four parameters, each presumed to represent a color, which creates an image representing a square tile of side 100 with background defined by the first color, on which are concentrically piled a disk of diameter 90 of the second color, a disk of diameter 60 of the third color, and a disk of diameter 30 of the fourth color. If the words are overwhelming, just look at the examples presented in the accompanying demo.
3. Write a program called `dots-permutations` taking three parameters, each presumed to represent a color, which creates a row of tiles representing the permutations of three colors, where each permutation is rendered as a stack of dots of diameters 90, 60, and 30. Please look to the accompanying demo for clarification.
4. Generate a demo that is just like the demo provided, **except that the colors in your demo should differ from those in mine.** Arrange for this demo to appear in the proper location of your solution document. For a list of available colors, look here:
https://docs.racket-lang.org/draw/color-database_-.html
5. Copy/pasted the code for this task to the appropriate locatin of your solution document.

Task 4: Post Your Work

Post your work to your web site by Tuesday, February 21, 2023.