Assignment: 03  
Examples Due: Friday, September 29, 2023 8:00 am  
Remainder Due: Tuesday, October 3, 2023 9:00 pm  
Coverage: Slide 17 of Module 05  
Language level: Beginning Student  
Files to submit: examples-a03.rkt, rgb.rkt, robot.rkt, collide.rkt, bonus-a03.rkt

• Make sure you read the OFFICIAL A03 post on Piazza for the answers to frequently asked questions.

• Unless otherwise specified, you may only use Racket language features we have covered up to the coverage point above (Slide 17 of Module 05). As well, you have not seen recursion up to this point in the course, and it is not allowed for this assignment.

• For each function you are required to write, you are also required to submit the design recipe.

• The names of functions we tell you to write, and symbols and strings we specify must match the descriptions in the assignment questions exactly. Any discrepancies in your solutions may lead to a severe loss of correctness marks. Basic test results will catch many, but not necessarily all of these types of errors.

• Policies from Assignment A02 carry forward, including:
  – You are required to submit examples by **Friday, September 29 at 8:00AM** (before the due date of assignment A03) in the file examples-a03.rkt.
  – More details about the design recipe grading are specified in the A03 Official Post on Piazza.

Here are the assignment questions you need to solve and submit.

1. (15%): In this question you will perform step-by-step evaluations of Racket programs, as you did in assignment one. Please review the instructions on stepping in A01.

   To begin, visit this web page:

   [https://www.student.cs.uwaterloo.ca/~cs135/stepping](https://www.student.cs.uwaterloo.ca/~cs135/stepping)

   When you are ready, complete the three required questions under the "Module 5a: Lists" category, using the semantics given in class for Beginning Student.
2. (30%): In computer graphics and image processing colour is typically represented as a triplet $(R,G,B)$ of whole numbers representing the red, green, and blue components respectively. In most images, the values for each colour are in the range $[0, 2^{8} - 1] = [0, 255]$.

Humans, however, do not typically think of colours according to their RGB values but by their names. For example, red, is $(255, 0, 0)$ and yellow is $(255, 255, 0)$.

In this question, we’ll store the RGB-triplet as a three-element natural number list. For example, $(\text{cons} \ 255 \ (\text{cons} \ 0 \ (\text{cons} \ 0 \ \text{empty})))$, is red.

The data definition for an RGB triplet is as follows:

```scheme
;; An RGB Triplet (RGB) is a (cons Nat (cons Nat (cons Nat empty)))
;; requires: each element in the list must be <= 255
```

Place your solutions to the following in rgb.rkt.

(a) Write a function called \textbf{RGB->name} that consumes a three element list and produces the colour name as a symbol. \textbf{RGB->name} should be able to identify: red, green, blue, yellow, cyan, magenta, white, and black. Symbol names must be in lowercase. For any other colour, the symbol ‘\textbf{unknown}’ should be produced. You may find the following table mapping RGB values to names helpful.

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>0</td>
<td>0</td>
<td>red</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>0</td>
<td>green</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>255</td>
<td>blue</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>black</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>white</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>0</td>
<td>yellow</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
<td>magenta</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>255</td>
<td>cyan</td>
</tr>
</tbody>
</table>

(b) Write a function called \textbf{name->RGB} that consumes a symbol, which is the name of a colour, and produces a three element list that is the $(R, G, B)$ triplet for that colour. \textbf{name->RGB} should be able to convert: ‘red’, ‘green’, ‘blue’, ‘yellow’, ‘cyan’, ‘magenta’, ‘white’, and ‘black’. If the provided symbol is not one of the specified colours, produce the list $(\text{cons} \ -1 \ (\text{cons} \ -1 \ (\text{cons} \ -1 \ \text{empty})))$.

(c) Some filters applied to images convert the RGB colour to a single value known as grayscale. Write a function called \textbf{RGB->luminosity} that consumes a three element list and produces a number that is the corresponding luminosity. Luminosity is computed as $L = 0.3 \times R + 0.59 \times G + 0.11 \times B$. 

CS 135 — Fall 2023 Assignment 03
(d) Write a type predicate called valid-RGB? that consumes *anything* and produces *true* if the consumed argument is a valid RGB Triplet according to the data definition above and *false* otherwise.

Here are some examples:

```
(check-expect (valid-RGB? 'red) false)
(check-expect (valid-RGB? (cons 255 (cons 0 (cons 0 empty)))) true)
(check-expect (valid-RGB? (cons 314 (cons 159 (cons 26 empty)))) false)
(check-expect (valid-RGB? "CMYK is the superior colour model") false)
```

Although it is not disallowed, you will receive a **1% bonus** on this assignment for writing valid-RGB? without using **cond**.

(e) When creating a website, colour is typically given not as an RGB triplet of integers, but as a 6-digit hexadecimal number. For example, red is FF0000 and yellow is FFFF00.

Write a function called RGB->hex that consumes an RGB value in the form of a three element natural number list and produces a six element list that contains the hexadecimal value of that colour. Each digit should be presented as a *string*. A string is text and should be enclosed by double quotes. For example, the hexadecimal value A would be represented by "A".

Red, (255,0,0) would produce a list: (cons "F" (cons "F" (cons "0" (cons "0" (cons "0" (cons "0" empty)))))).

For this function only, you may use the built-in function number->string.

Note: A hexadecimal number is also known as a base-16 number. There are 16 possible values for each digit, 0 – 9 and A,B,C,D,E,F where A = 10, B = 11, C = 12, D = 13, E = 14, F = 15. 255 in decimal is FF in hexadecimal because 255 = 16*15 + 15.

Hint: use the quotient and remainder functions.

3. **(25%)**: A robot’s state is given by its (x,y) position on an integer grid and the direction it’s facing, one of *North*, *South*, *East*, or *West*. Due to power distribution requirements, robots are currently restricted to the square integer grid defined by the opposite corners (0,0) and (10,10) inclusive (that is, the robot may be at (10,10)).

In this problem, we will be representing a robot’s state as a three-element list storing its x coordinate, y coordinate, and direction (in that order).

Place your solutions to the following in robot.rkt.

(a) Write a data definition for a robot’s State.

(b) Write a function robot-ctl. The robot control function consumes a State and a command. Commands are the symbols ‘forward, ‘turn-left, and ‘turn-right. robot-ctl produces a new State.
A robot command of 'turn-left or 'turn-right always succeeds. The produced State will be the same as the consumed State except that the direction will be different. For example, a robot facing 'North and told to 'turn-left will then face 'West. Additional 'turn-left commands will cause it to face 'South, 'East, and finally, 'North again.

A robot command of 'forward changes the y coordinate by 1 if facing 'North and by -1 if it is facing 'South and similarly for the x coordinate if it is facing 'East or 'West. However, if the robot is already at the edge of its power grid, the state does not change. The 'forward command does not change the robot’s direction.

Just for fun (and absolutely no effect on your assignment mark), you may play with a primitive graphical user interface for your robot controller. Download the file robot-gui.rkt to the same folder as robot.rkt. Add the following line to your robot.rkt file:

(require "robot-gui.rkt")

Then, in your interactions window, type:

(robot-gui (cons 0 (cons 0 (cons 'North empty))) robot-ctl)

The first parameter to robot-gui is the initial state of the robot. The second parameter is your solution to part (b) (yes, Racket allows you to pass functions as arguments to functions; this is a really cool part of Racket that we’ll explore in detail later in the term).

4. (30%): When playing a video game like Super Mario Bros. or Legend of Zelda, your character may run into objects or get shot by projectiles. These events are known as collisions. It is very expensive, computationally, to calculate exactly when and where two objects collide. So, we cheat. We place game objects, called assets, into simple bounding shapes, such as a sphere, and compute whether the bounding shapes collide—a much easier/faster process.

In this question, a point will be represented by a three element list of numbers as (cons x (cons y (cons z empty))). A sphere will be represented by a two element list, where the first element is a point representing the center of the sphere —represented by a three element list, and the second element is the sphere radius which must be greater than 0. For example, (cons (cons 0 (cons 1 (cons 2 empty))) (cons 5 empty)) represents a sphere with radius 5 centered at (0,1,2). You may consider writing data definitions for Point and Sphere to use in your contract.

(a) Write a function called build-sphere that consumes a list of four numbers as (cons x (cons y (cons z (cons r empty)))) and produces a sphere that has the form (cons (cons x (cons y (cons z empty))) (cons r empty)).

(b) Write a function called valid-sphere? that consumes a list as (cons (cons x (cons y (cons z empty))) (cons r empty)) and produces true if that list is a valid sphere, that is, the radius is greater than 0. The function should produce false otherwise.
(c) Write a function called `distance-between-points` that consumes two points and produces the distance between them.

The distance between \((x_1, y_1, z_1)\) and \((x_2, y_2, z_2)\) is given by:
\[
d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.
\]

(d) Write a function called `point-in-sphere?` that consumes a point and a sphere and produces true if the point is inside the sphere and false otherwise. Recall that the equation of a sphere is:
\[
(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = r^2
\]
where \((c_x, c_y, c_z)\) is the position of the sphere’s center, \(r\) is the sphere’s radius, and \((x, y, z)\) is the point.

Note: a point \((x, y, z)\) is in the sphere if \((x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 \leq r^2\).

(e) Write a function called `collide?` that consumes two spheres and produces true if the spheres collide and false otherwise. Two spheres collide if there is a point inside the first sphere that is also inside the second sphere.

Place your functions for this question in a file `collide.rkt`.

This concludes the list of questions for you to submit solutions. Don’t forget to always check the basic test results after making a submission.

Assignments will sometimes have additional questions that you may submit for bonus marks.

5. **5% Bonus:** A box is specified by a four element list as: \((\text{cons } xmin \ (\text{cons } xmax \ (\text{cons } ymin \ (\text{cons } ymax \ empty))))\). Write a function called `overlap-area` that consumes two boxes and produces the size of the overlapping region. If the boxes do not overlap, then 0 should be produced. A negative value should never be produced.

Note: for all inputs you may assume that \(xmin < xmax\) and \(ymin < ymax\). Place your solution in `bonus-a03.rkt`. 
Enhancements: Reminder—enhancements are for your interest and are not to be handed in.

The textbook upon which CS135 is based has graphical examples. We’ve tended to avoid them because they aren’t purely functional and are hard to test. But they are excellent enhancement material!

Start by adding (require 2htdp/image) at the top of an empty Racket program.

You can make a solid blue circle with the function application (circle 25 'solid 'blue). The arguments are the radius (in pixels), the “mode” (one of ‘solid, ‘outline, or an integer between 0 and 255 to indicate the transparency), and the colour (e.g. ‘blue, ‘black, and ‘purple; or see the documentation for a much longer list). There are similar functions available to create lines, ellipses, text, and polygons.

Multiple shapes can be combined with the overlay or underlay functions and their variants. For example, the following program will create the image on the right.

(require 2htdp/image)
(underlay (rectangle 80 80 'solid 'mediumseagreen) (polygon (list (make-posn 0 0) (make-posn 50 0) (make-posn 0 50) (make-posn 50 50)) 'outline (make-pen 'darkslategray 10 'solid 'round 'round)))

Image manipulation can be extended to simple animations by using (require 2htdp/universe.rkt). It provides functions to initialize a canvas and schedule repeated updates at the “tick” of a clock. The updates occur through the use of an update function, which you write and which consumes an integer representing the current tick and produces an image of the universe at that time.

You can read the related documentation through DrRacket’s Help Desk. We invite you to explore the universe of functional animation.

A very simple example:

(require 2htdp/image)
(require 2htdp/universe)
(define max-size 100)

;; (update tick) updates the image for each tick of the animation's clock.
(define (update tick)
  (underlay
(empty-scene (* 2 max-size) (* 2 max-size))
(circle (modulo tick max-size) 'solid 'red))

(animate update)

You might also look for techniques that allow you to program responses to graphical updates through keyboard events.

Consider what you would need on top of this to implement the basics of some of your favourite games or interactive applications. What sort of computations do you need to describe? And what Racket vocabulary are you lacking in order to express those computations?

Notice that animate is consuming a function as an argument in order to get the updates done. This violates the rules of Beginning Student, but hints at the power of Racket that will be revealed in the upcoming weeks.

The robot-gui that is provided uses more advanced techniques than described above. Take a look, if you want, but be fore-warned that it’s different.