Assignment: 3
Due: Tuesday, February 4, 9:00 pm
Language level: Beginning Student
Files to submit: debug-a03.rkt, rgb.rkt, collide.rkt, bonus-a03.rkt
Warmup exercises: 9.1.1, 9.1.2
Practice exercises: 9.1.3

• Make sure you read the OFFICIAL A03 post on Piazza for the answers to frequently asked questions.
• This assignment covers concepts up to Slide 18 of Module 06. Unless otherwise specified, you may only use Racket language features we have covered up to that point. You may also use the built-in functions second, third, fourth, fifth, if you find them useful. As well, you have not seen recursion up to this point in the course, and it is not necessary nor recommended for this assignment.
• 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 tests results will catch many, but not necessarily all of these types of errors.
• Policies from Assignment 2 carry forward.

1. [10% Correctness] Question 1

Consider the code in debug-a03.rkt, which may contain syntax, mathematical, and style errors. Correct the code and style.

2. 10% Correctness Question 2

To help reinforce the substitution rules we have discussed in class, we have provided an online tool, known as the “Stepper” at

https://www.student.cs.uwaterloo.ca/~cs135/assign/stepping/

You will need to authenticate yourself using your Quest/WatIAM ID and password. Once you are logged in, click on the question labelled “05_practice” under “Module 2a: Expressions” and complete it to familiarize yourself with the system. Note the “Show instructions” link at the bottom of each problem. Read the instructions!

Complete each stepping problem in the “Required questions” sections for “Module 2a: Expressions” and “Module 6: Lists”, using the substitution rules given in class.

Some additional notes:
• There are practice problems associated with each topic that you can choose to complete if you want more experience with a certain topic. These problems have a “Hint” button enabled to help you get started.

• You can re-enter a step as many times as necessary until you get it right, so keep trying until you completely finish every question.

• All you have to do is complete the questions online—we will be recording your answers as you go, and there is no file to submit. You are not done with a question until you see the message Question complete! You should see this once you have arrived at a final value and clicked on “simplest form” (or “Error,” depending on the question). The basic tests for this assignment will tell you whether or not we have a record of your completion of the stepper problems.

• For question with constant definitions, you are assuming that the constants have already been fully processed as though you have pressed Run in DrRacket. This means that you are not required to do any simplification of the constant definitions as part of the stepping.

• You should not use DrRacket’s Stepper to help you with stepper questions for several reasons. First, DrRacket’s evaluation rules are slightly different from the ones presented in class, but we need you to use the evaluation rules presented in class. Second, in an exam situation, you will not have DrRacket’s Stepper to help you, and there will definitely be step-by-step evaluation questions on at least one of the exams.

• If you get stuck on a stepping question, do not post to Piazza requesting the next step. This is a violation of the academic integrity policy. Review the substitution rules carefully to try to solve the problem yourself. If you still cannot find your error, then you are encouraged to ask a question in person during office hours. As a last resort, you may make a private post to Piazza describing where you are stuck. Course staff will provide guidance directing you to the next step, but they will not give you the answer.

3. [30% Correctness] Question 3

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 \([0, 255]\).

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 store the RGB-triplet as a three-element list. For example, \(\text{(cons} 255 \text{(cons} 0 \text{(cons} 0 \text{empty)})))\), is red.

(a) Write a function called \(\text{RGB->name}\) that consumes a three element list and produces the colour name as a symbol. \(\text{RGB->name}\) 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 '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 `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. `name->RGB` should be able to identify: red, green, blue, yellow, cyan, magenta, white, and black. If the provided symbol is not one of the specified colours, produce the list `(cons -1 (cons -1 (cons -1 empty)))`.

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

(d) When creating a website, colour is typically given not as an RGB triplet of integers, but as a 6-digit hexadecimal number (recall: hexadecimal is base 16). 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" 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.

(e) Write a function called `colour-difference` that consumes two three-element lists and produces the difference between those two colours. The difference can be computed as \(d = \sqrt{(r_0 - r_1)^2 + (g_0 - g_1)^2 + (b_0 - b_1)^2}\) where \((r_0, g_0, b_0)\) and \((r_1, g_1, b_1)\) are the RGB representations of two colours.
Place your functions for this question in rgb.rkt.

4. [30% Correctness] Question 4

When playing a video game like Unreal Tournament or 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 \((\text{cons} \ x \ (\text{cons} \ y \ (\text{cons} \ z \ \text{empty})))\). A sphere will be represented by a two element list, where the first element is a point—represented by a three element list, and the second element is the sphere radius which must be greater than 0.

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

(b) Write a function called \textbf{valid-sphere?} that consumes a list as \((\text{cons} \ (\text{cons} \ X \ (\text{cons} \ Y \ (\text{cons} \ Z \ \text{empty}))) \ (\text{cons} \ R \ \text{empty}))\) 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 \textbf{distance-between-points} that consumes two points and produces the distance between them. Hint: the function \textit{colour-difference} is very similar.

(d) Write a function called \textbf{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 \textbf{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 \texttt{collide.rkt}.

\textbf{Bonus Question} [5\%] A box is specified by a four element list as: \((\text{cons} \ x_{\text{min}} \ (\text{cons} \ x_{\text{max}} \ (\text{cons} \ y_{\text{min}} \ (\text{cons} \ y_{\text{max}} \ \text{empty}))))\). Write a function called \textbf{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 \(x_{\text{min}} < x_{\text{max}}\) and \(y_{\text{min}} < y_{\text{max}}\). Place your solution in \texttt{bonus-a03.rkt}. 

---

CS 135 — Winter 2020 Assignment 3 4