CS 131 - Homework 2

Deadline:
5:00 pm on Friday, September 10

How to submit:
When you are done with the following problems:
• save your resulting Definitions window contents in a file with the suffix .rkt
• transfer/save that file to a directory on nrs-labs (preferably in a folder/directory named 131hw2)
• use ssh to connect to nrs-labs
• cd to the folder/directory where you saved it (cd 131hw2 for example)
• use the ls command to make sure your .rkt file is really there
• use ~st10/131submit to submit it, with a homework number of 2
• make sure that ~st10/131submit shows that it submitted your homework .rkt file

Purpose:
To provide practice writing functions using the design recipe and using named constants.

Important notes:
• Each student should work individually on this homework.
• You are expected to follow the Design Recipe for all functions that you write. So, each function is expected to include:
  – a signature comment, including the name of the function, the types of expressions it expects, and the type of expression is produces. This should be written as discussed in class (and you can find examples posted on the public course web page). For example,
    ; signature: rect-area: number number -> number
  – a purpose comment, describing what the function expects and describing what it produces. For example,
    ; purpose: expects the length and width of a rectangle, 
    ; and produces the area of that rectangle
  – [following the design recipe, you will be writing the function header next; note that you don't need to write it twice. Follow the function header with a body of . . . at this stage, and replace that . . . with its body later, at the appropriate step in the design recipe.]
  – check-expect expressions expressing the specific examples you devise before writing your function body. (These may be placed before or after your actual function, but you are expected to create these before writing the function body. I'll have no way of knowing if you really write these in the correct order, but note that I won't answer questions about your function body without seeing your examples written as check-expect expressions first...) For example,
(check-expect (rect-area 3 4) 12)

- How many check-expect expressions should you have? That is an excellent question, and a major topic.
  
  For this homework, I'll say how many you need, but we'll be discussing how you determine how many you need, and later you'll be graded based on whether you include a reasonable number and kind of check-expect expressions.
  
  The basic rule of thumb is that you need an example/check-expect for each "case" or category of data that may occur... and you can always add more if you'd like!

- [and, of course, your function definition itself!]

- You may include as many additional calls or tests of your function as you would like after its definition.

**Because the Design Recipe is so important**, you will receive **significant** credit for the signature, purpose, header, and examples/check-expects portions of your functions. Typically you'll get at least half-credit for a correct signature, purpose, header, and examples/check-expects, even if your function body is not correct (and, you'll **lose at least half-credit** if you omit these or do them poorly, even if your function body is correct).

**The Problems:**

**Problem 0**

Start up DrRacket, setting the language to **Beginning Student** and adding the HTDP/2e versions of the **image** and **universe** teachpacks by putting these lines at the beginning of your Definitions window:

```
(require 2htdp/universe)
(require 2htdp/image)
```

Put a blank line, and then type in a comment-line containing your name, followed by a comment-line containing CS 131 - HW 2, followed by a comment-line with no other text in it --- that is,

`; type in YOUR name
; CS 131 - HW 2
;`

**Problem 1**

Below what you typed in #0 above, type the comment lines:

`; Problem 1
`

Obtain an image -- of one of the formats .jpg, .png, or .gif -- no larger than 100 pixels by 100 pixels. (You can find one on the Web, or create it in some application, etc.) Write a Racket definition giving a descriptive name to this image (copying or inserting the image into your DrRacket definitions window).

Then, write an expression that uses **overlay**, **above**, or **beside** (your choice) with this name you've defined to use this image with another image of your choice.
**Problem 2**
Skip a line, and write a comment noting that what follows are your expressions for:

; Problem 2
;
Using the design recipe, design a function that produces the volume of a rectangular tank. (You'll need to consider: how many and what type of expressions should such a function expect, to be able to produce such a volume?)

One specific example/check-expect expression should suffice for this function.

**Problem 3**
Skip a line, and write a comment noting that what follows are your expressions for:

; Problem 3
;
Using the design recipe, design a function that produces the average gas consumption (in miles-per-gallon) used for a trip. (You'll need to consider: how many and what type of expressions should such a function expect, to be able to produce this average?)

One specific example/check-expect expression should suffice for this function.

**Problem 4**
Skip a line, and write a comment noting that what follows are your expressions for:

; Problem 4
;
Using the design recipe, design a function that expects a desired length in pixels and four colors, and produces a square image whose sides are each that length, but made up of four smaller squares each of which is one of the four given colors.

Provide at least two specific examples/check-expect expressions for this function.

**Problem 5**
Skip a line, and write a comment noting that what follows are your expressions for:

; Problem 5
;
Write a Racket definition that will define the name MIN-PER-HR to be the number of minutes in an hour. (This is a named constant, NOT a function!)

Then, using the design recipe and this named constant, design a function minutes->hours that expects a number of minutes and produces the number of hours equivalent to that number of minutes.

(Yes, that "arrow", the dash and right angle bracket ->, is intended -- it is not a typo, and it is permitted in a Racket identifier.)

Provide at least two specific examples/check-expect expressions, at least one for a number of minutes less than 60, and at least one for a number of minutes greater than 60.

**Problem 6**
Skip a line, and write a comment noting that what follows are your expressions for:
; Problem 6

; Write a Racket definition, another named constant, that will define the name IN-PER-FT to be the number of inches in a foot.

Then, using the design recipe and this named constant, design a function total-inches that expects a number of feet and a number of inches, and produces the total number of inches. (For example, the value of the expression (total-inches 4 5) should be 53, because 4 feet and 5 inches is 53 inches overall.)

Provide at least two specific examples/check-expect expressions, at least one for a number of feet of 0, and at least one for a number of feet greater than 1.

Problem 7

Skip a line, and write a comment noting that what follows are your expressions for:

; Problem 7

; Consider your named image from Problem 1. Also consider the overlay and circle operations.

Using the design recipe, design a function frame-it that expects a desired "matte" color, a desired "frame" color, and a desired length in pixels. Then it produces the following image: your named image from Problem 1 atop a circle of the desired "matte" color with that desired length as its diameter, atop a rectangle of the desired "frame" color with that desired length as its width and height.

Provide at least two specific examples/check-expect expressions.

Problem 8

Skip a line, and write a comment noting that what follows are your expressions for:

; Problem 8

; Consider your named image from Problem 1.

• Decide on a width and height you would like for an animation involving your named image. Define named constants for this width and height.

• Define a backdrop/background scene including at least 3 visible components within it.

• Paste in the function get-1-bigger from the Week 2-Lecture 2 or from your Week 2 Lab Exercise, or your choice of a function that takes a number and produces a number (to be the function for big-bang's on-tick expression).

• Using the design recipe, define a drawing function that expects a number and produces a scene based on that number, using your named image and your named backdrop.
  – Include at least two specific examples/check-expect expressions

• Include a big-bang expression that uses these two functions above to start up an animation.

NOTE that I hope to TRY to combine the animations from this problem into a single combined class animation, if is doesn't crash my computer to do so... 8-)}