CS 131 - Homework 1

Deadline:

5:00 pm on Friday, September 3

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 `131hw1`)
- use **ssh** to connect to **nrs-labs**
- `cd` to the folder/directory where you saved it (`cd 131hw1` 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 **1**
- make sure that `~st10/131submit` shows that it submitted your homework `.rkt` file
- (we will be practicing the above in lab on Friday, August 27th -- ASK ME if this is not clear after that, or if you have any problems with submission!)

Purpose:

To practice with simple and compound expressions of various data types in Racket.

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 1, followed by a comment-line with no other text in it --- that is,

```
; type in YOUR name
; CS 131 - HW 1
;
```

Problem 1

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

```
; Problem 1
;
```

Now type the expressions specified below in the Racket Definitions window, each starting on its own line. Run (push the Run button) and look in the Interactions window (in the lower window) to see if you get the expected results for each.

(a) a simple expression of type number representing the number 47
(b) a compound expression of type number representing the sum of 13 and 47
(c) a simple expression of type string that includes your name
(d) a simple expression of type string that represents a color of your choice
(e) a compound expression of type image that is a rectangle of the color of your expression from (d), of width 75 pixels and height 45 pixels

Problem 2

Next, in your definitions window, type the comment lines:

```
;  Problem 2
;
```

Recall `check-expect` from lab -- it expects two expressions of any type and tests to see if the second has the same value as the first. What is unusual about this operation is that it doesn't produce a boolean value -- instead, it has several side-effects. If the values of its expressions are the same for all of the `check-expect` expressions in your Definitions window, it just causes a message saying how many tests were passed to be displayed at the end of the Interactions window. But if the values of the expressions are **not** the same, a separate window is displayed, giving how the expressions differed, with links to each `check-expect` expression that "failed".

Figure out (by yourself) what you think the value is for each of the following Racket expressions.

- Are any using **incorrect** syntax? If so, type them within a **comment** in your Definitions window.
- For each of those using **correct** syntax, write a `check-expect` expression including the expression below and what you think its value should be.

Then push the Run button, and see if your tests -- your `check-expect` expressions -- pass. If any do not, feel free to tweak them until they do -- but try to figure out why their values differed from what you expected. The important thing here is to think about what **should** happen, and then compare it to what **really** happens.

(a) `(* (- 3 5) 20)`
(b) `(* pi (* 10 10))`
(c) `(+ 73 true)`
(d) `(/ 13 0)`
(e) `(or (< 1 2) (> 5 8))`
(f) `(and (< 1 2) (> 5 8))`
(g)
```
(< (image-width (circle 20 "solid" "red"))
   (image-width (rectangle 30 35 "solid" "green")))
```

Problem 3

Next, in your definitions window, type the comment lines:

```
;  Problem 3
;
```

Racket happens to have a built-in operation for finding the maximum of some set of numbers, called **max**. Write expressions in your definitions window, each on its own line, using **max** to see if it works:

* with 2 values,
* with more than two values,
* with just one value, and
* with no values.

If any of these do not work, **COMMENT OUT** that expression in your definitions window (put a **;** in FRONT of it) and re-run (so the subsequent expressions can execute!)

Problem 4

Next, in your definitions window, type the comment lines:

```
;  Problem 4
;
```

The following are currently **not** "proper" Racket expressions (expressions that follow Racket's syntax rules). **Correct** each, and then type each (now-"legal"-syntax) Racket expression in your definitions window (starting on its own line).

(a) `(17)`
(b) `image-width (circle 27 "solid" "green")`
(c) `(* (/ 7 3) (8) (image-height (circle 2 "blue"))`

Problem 5

Next, in your definitions window, type the comment lines:

```
;  Problem 5
;
```

Write an expression in DrRacket that meets all of the following requirements:

* it should result in an expression of type `image`
* it should be a **compound** expression
* it should use at least **three** image operations from the **image** teachpack
* it must be precisely **200** pixels wide and **200** pixels high

Other than that -- you may make your image as simple or as complex as you would like.

NOTE that all of the class' images from this problem are going to be combined into a single class image.

Problem 6

Write a `define` expression that defines a descriptive name (a descriptive identifier) of your choice for your image from Problem 5.

Now, using this name:

* Write a `check-expect` expression that will be true if your image is 200 pixels wide.
  * hint: use `image-width`
* Write a `check-expect` expression that will be true if your image is 200 pixels high
  * hint: use `image-height`