CS 111 - Homework 1

Deadline

11:59 pm on **Friday, September 5, 2025**

Purpose

To make sure you have read over the course syllabus, and to practice with simple and compound expressions of various data types in Racket.

How to submit

You complete **Problems 1 and 2** on the course Canvas site (syllabus confirmation and reading questions, and some short-answer questions mostly related to BSL Racket expression basics so that you can see if you are on the right track).

Then, you will submit your work for **Problems 3 onward** on the course Canvas site.

Each time you would like to submit your work for Problems 3 onward:

- Save your current DrRacket Definitions window contents in a file with the name **`111hw1.rkt`**
  - Note: please use that **exact** name -- do not change the case, add blanks, etc. If I search for a file with that name in your submission, I want it to show up!
  - (Canvas DOES add your name and some numbers to your submitted file on the graders' end, including numbers on the end of the submitted file name if you submit more than once -- that's fine! Just please start out, on your end, with the file name **`111hw1.rkt`**.)
- Open https://canvas.humboldt.edu in a web browser, and click on the icon for the CS 111 course (or otherwise connect to the course Canvas site)
  - Click **Modules** on the left-hand-side of the page
  - Scroll down to the **Homeworks** section
  - Choose the **rest of Homework 1** link
  - Follow the Canvas instructions for uploading your Racket file **`111hw1.rkt`**.

NOTE: If you have trouble submitting, please let me know!

Important notes - 10 points

- Be sure to use the file name specified above (and note that Canvas does add your name and numbers to your file's name, and that's OK!).
- To make your BSL Racket code easier for the graders to grade, include the string expressions specified in each problem along with your answers for that problem.
- To make your BSL Racket code more readable, **put a blank line before *and* after:**
  - **each Racket comment or "block" of comments**
  - **each Racket multi-line expression**
  - Also, IF you would like, you can add an additional comment "border" before and/or after Racket comments.

For example, something like:

```
;========
```

or

```
;--------
```

- To make your BSL Racket code more readable, when you write a long compound expression:
  - **indent** the continued part to make it clearer it is still part of a long expression
  - **line-up** its arguments
- Please let me know if you have any questions or concerns about the above requirements.

Problem 1 - 20 points

Problem 1 is correctly answering the "HW 1 - Problem 1 - required syllabus confirmation and reading questions" on the course Canvas site.

Problem 2 - 20 points

Problem 2 is correctly answering the "HW 1 - Problem 2 - Short-answer questions on BSL Racket expression basics" on the course Canvas site.

Problem 3 - 10 points

Complete the remainder of Homework 1's problems in a file named **`111hw1.rkt`** (that is, save your DrRacket Definitions window to a file named **`111hw1.rkt`**). Be sure to save frequently!

Start up DrRacket, set the language to How To Design Programs - **Beginning Student** level, and add the HTDP/2e image module by putting this expression at the beginning of your Definitions window:

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

Put a blank line, followed by these comments, adding in your name, and follow these with another blank line:

```
;  put your name here
; CS 111 - HW 1
; last modified: 2025-09-02
```

Below this -- after the blank line -- type this `string` expression:

```
"=== Problem 3 ==="
```

...also followed by another blank line.

Now type the comments and expressions specified below in the Racket Definitions window.

After you type in your answer for each part, "run" the Racket file (push the **Run** button) and look in the Interactions window (in the lower window) to see if you get the expected results for each.

3 part a

After a blank line, type the string expression:

```
"--- 3 part a ---"
```

...followed by another blank line.

Consider the temperature at which water boils. This value varies based on the temperature scale -- it is different in Fahrenheit than it is for Celsius than it is for Kelvin, for example.

Choose a temperature scale, and then:

- Write a Racket **comment** containing the **name** of that temperature scale.
- Then, leave a blank line, and on the line after that comment, write a BSL Racket **simple** expression of type **`number`** whose value is the temperature at which water boils in your chosen temperature scale.

3 part b

After a blank line, type the string expression:

`"--- 3 part b ---"`

...followed by another blank line.

Consider a person's name. This might be used to sign something, or to label something as belonging to someone; one might want to grab its initials (to monogram a thing) or have it displayed in all-uppercase or mixed case, or even in different colors and sizes of letters.

Then:

- Write a BSL Racket **simple** expression, of a reasonable type for doing actions like those described above, that represents your name. (You get to decide how much of your name to include, and what version of your name to include.)

3 part c

After a blank line, type the string expression:

`"--- 3 part c ---"`

...followed by another blank line.

Then:

- Write a **simple** OR **compound** expression -- your choice! -- of type **`boolean`**.

3 part d

After a blank line, type the string expression:

`"--- 3 part d ---"`

...followed by another blank line.

Then:

- Write a **simple** OR **compound** expression -- your choice! -- of type **`image`**.

Problem 4 - 9 points

Next, in your definitions window, after a blank line, type this string expression:

`"=== Problem 4 ==="`

...followed by another blank line.

The goal here is to practice **FIXING** some "**broken**" Racket expressions.

Each of the following is ***intended*** to be a single BSL Racket expression, but they are currently **NOT** "correct" Racket expressions -- they do NOT follow Racket's syntax rules.

For each, give the specified string and then give a **corrected** version of the "broken" expression (**with a minimum of alteration**) such that:

- the result is a single expression that **DOES** follow Racket's syntax rules
- all of the simple expressions in the "broken" expression are included in the "fixed" expression

After you type in your answer for each part, "run" the Racket file (push the **Run** button) and look in the Interactions window (in the lower window) to see if you get a result (and not an error message) for your corrected version.

(Note: sometimes there is more than one reasonable way to correct each of these!)

4 part a

After a blank line, type the string expression:

```
"--- 4 part a ---"
```

...followed by another blank line.

Now "fix" the expression:

```
(287.65)
```

4 part b

After a blank line, type the string expression:

```
"--- 4 part b ---"
```

...followed by another blank line.

Now "fix" the expression:

```
rectangle "outline" (30 15) "purple"
```

4 part c

After a blank line, type the string expression:

```
"--- 4 part c ---"
```

...followed by another blank line.

Now "fix" the expression:

```
((/ (32 - 21) (3 + 9) (16))
```

Problem 5 - 8 points

Next, in your definitions window, after a blank line, type this string expression:

```
"=== Problem 5 ==="
```

...followed by another blank line.

Look over the handout, "Some DrRacket Tidbits", posted along with this homework handout.

Choose **at least FOUR different** image or image-related functions **from this handout** that you will try out.

Then, after a blank line, write:

- **at least FOUR** *different* compound expressions **of type image**,
- amongst these compound image expressions, use **at least four** *different* image or image-related function **from this handout**.

(And, when you "run" the Racket file (push the **Run** button) and look in the Interactions window (in the lower window), you should see **at least four image results** from your at least four compound expressions of type image for this problem.)

Problem 6 - 9 points

Next, in your definitions window, after a blank line, type this string expression:

`"=== Problem 6 ==="`

...followed by another blank line.

Remember that an argument expression **within** a compound expression may *itself* be a compound expression.

That is, as long as the **`triangle`** operation gets three expression arguments of the expected types, those expression arguments may be simple or compound:

```
(triangle 30 "solid" "red")

(triangle (+ 10 (string-length "MOOOOOOOOOOOO"))
          (string-append "out" "line")
          "darkgreen")
```

After a blank line, write a **SINGLE compound expression** in DrRacket that meets **all** of the following requirements:

- It should be an expression that you have **not** already given as an answer to a previous Homework 1 problem.
  - It is OK if some of its **arguments** are expressions you have given as answers to previous Homework 1 problems.
- It should be of type `image` -- it should result in an image.
- It should use **at least THREE** *different* image or image-related operations from the `2htdp/image` module. (More is fine!)
  - They can be image operations from the "Some DrRacket Tidbits" handout, OR they can be image operations (from the `2htdp/image` module) that you find by exploring DrRacket's **Help** feature, described a bit at the end of the "Some DrRacket Tidbits" handout.

(And, when you "run" the Racket file (push the **Run** button) and look in the Interactions window (in the lower window), you should see **JUST ONE** image result from this problem's expression.)

Problem 7 - 14 points

Next, in your definitions window, after a blank line, type this string expression:

`"=== Problem 7 ==="`

...followed by another blank line.

We are going to make a class "quilt" image of 200-pixels-wide, 200-pixels-high images, with each class member designing one of the 200-pixel-by-200-pixel images.

After a blank line, write **one or more expressions** that meet the following requirements:

- the final resulting image should involve at least one expression that you have **not** already given as an answer to a previous Homework 1 problem.
  - It is OK if some of the expressions or arguments involved are expressions you have given as answers to previous Homework 1 problems.
- the final resulting image somehow should be the result of **at least THREE** *different* image or image-related operations from the `2htdp/image` module. (More is fine!)
  - They can be image operations from the "Some DrRacket Tidbits" handout, OR they can be image operations (from the `2htdp/image` module) that you find by exploring DrRacket's **Help** feature, described a bit at the end of the "Some DrRacket Tidbits" handout.
- the final resulting image should be an expression of type image that is **precisely** 200 pixels wide and 200 pixels high
  - You might ask -- how can I be **SURE** my final image is the right size?
  - There's a function **`image-width`** expects an image argument (in this case, the compound expression that draws your image), and returns its width in pixels (a number);

    and a function **`image-height`** that expects an image argument, and returns its height in pixels (a number).
  - OPTIONAL: IF you would like: you can use these functions to test the width and height of your final image.
- ACCEPTABLE "extended" version of this problem: IF you would like to use **`define`** to give names to some expressions to make your task easier, that is allowed and encouraged (but not required!) for this problem.
- As long as you meet the above requirements, your image may be as simple or as complex as you would like.

NOTE: You will have a chance to revise this image in the next homework assignment. Then, after Homework 2, eventually your image will be placed into a class "quilt" and posted on the public course web site. That's why it is important that your "quilt square" be precisely 200 pixels wide and 200 pixels high – so that it will fit in the quilt!