Bonus Racket Homework - worth up to 40 bonus homework points

Deadline

11:59 pm on **Friday, October 14, 2016**

Purpose:

To encourage those interested to practice programming using the design recipe in Racket a bit more during the time between Homework 5 (the last Racket homework) and Homework 6 (the first C++ homework, after Exam 1), including a bit more practice with lists and with file input/output, and even with writing a Racket-flavored `main` function.

How to submit

Each time you would like to submit your work:

- save your current Definitions window contents in a file with the name `111bonus.rkt`
- transfer/copy that file to a directory on `nrs-labs.humboldt.edu` (preferably in a folder/directory named `111bonus`)
- Now that your file is on nrs-labs.humboldt.edu, you need to log onto nrs-labs.humboldt.edu using `ssh`, so you can submit your file to me.
- WHILE you are logged onto nrs-labs:
  - IF you saved your file in a folder, use `cd` to change to the folder/directory where you saved it -- for example, if you saved it in the folder `111bonus`, then you would go to that directory by saying:

    ```
    cd 111bonus
    ```

  - use the `ls` command to make sure your `111bonus.rkt` file is really there:

    ```
    ls
    ```

  - type the command:

    ```
    ~st10/111submit
    ```

    ...and when asked, enter a homework number of **`55`**.

    ...and when asked, enter `y`, you do want to submit all files with an appropriate suffix (I don't mind getting some extra files, as long as I also get `111bonus.rkt`, `demo1.txt`, and `demo2.txt`; I'd rather receive too many files than too few due to typos.)

    ...make sure to carefully check the list of files submitted, and make SURE it lists `111bonus.rkt`, `demo1.txt`, and `demo2.txt` as having been submitted! (The most common error is to try to run `~st10/111submit` while in a different directory than where your files are...)

Problem 1 - worth up to 5 bonus points

Start up DrRacket, (if necessary) setting the language to How To Design Programs - **Beginning Student** or **Beginning Student with List Abbreviations** level, and adding the HTDP/2e versions of the image, universe, and batch-io teachpacks by putting these lines at the beginning of your Definitions window:

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

```
(require 2htdp/universe)
(require 2htdp/batch-io)
```

Put a blank line, and then type in:

- a comment-line containing your name,
- followed by a comment-line containing `CS 111 - Bonus HW`,
- followed by a comment-line giving the date you last modified this homework,
- followed by a comment-line with no other text in it --- for example:

```
; type in YOUR name
; CS 111 - Bonus HW
; last modified: 2016-10-06
;
```

Below this, after a blank line, now type the comment lines:

```
;
; Problem 1
;
```

Then, use `write-file` to create two files of at least 5 lines worth of contents of your choice, making sure that at least two lines contain more than one word each, in files `demo1.txt` and `demo2.txt`.

Problem 2 - worth up to 5 bonus points

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

```
;
;  Problem 2
;
```

Write a data definition comment for a `StringList`, a list containing just strings.

Then, develop the comment for the template for a function that "walks through" a `StringList`.

Copy over the data definition comment for an `ImageList` from the posted Week 6 Lecture 1 examples.

And, copy over the comment for a template for a function that "walks through" an `ImageList`.

Problem 3 - worth up to 10 bonus points

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

```
;
;  Problem 3
;
```

Using the design recipe, write a function `words-to-images` that expects a list of words, and produces a list of image-versions of those words. (You may choose the font size and color for the image-versions of the words -- they may be always the same, or you may creatively use `random`, your choice!)

Problem 4 - worth up to 10 bonus points

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

```
;
;  Problem 4
;
```

Now copy over the following from the posted Week 6 Lecture 1 examples:

- function `return-current` - (remember to grab its signature, purpose, and all of its tests, also!)
- function `draw-images-randomly` - (remember to grab its signature, purpose, and all of its tests, also!)

Recall: given a file, how can you read the contents of that file so that you get a list of the "words" in that file?

Using the design recipe, write a function named `main` that expects the name of a desired file and a desired speed (given as desired number of seconds between ticker ticks), and it has the side-effect of calling `big-bang` with an initial world that is the result of calling Problem 3's `words-to-images` function with the given file name's contents read in as a list of "words", and with appropriate `to-draw` and `on-tick` clauses so the world will tick at the specified speed -- it will also return the value of the current world when `big-bang` is stopped.

Call your resulting `main` at least twice, for at least your files `demo1.txt` and `demo2.txt`, each with a different speed value.

Problem 5 - worth up to 10 bonus points

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

```
;
;  Problem 5
;
```

Would you like to add keystroke-handling to this? Perhaps typing a letter key could add that letter to the world, or typing an arrow key could add or remove words from the current world; you can decide, and design a function that expects the current list-of-words world and a keystroke string, and returns the list-of-words world that should result.

Then modify Problem 4's `main` function so that it also includes an `on-key` clause including your function, and call that resulting `main` at least twice, for at least your files `demo1.txt` and `demo2.txt`, each with a different speed value.