CS 111 - Week 4 Lab Exercise - 2016-09-16

Deadline

Due by the end of lab today. (Submit whatever you have by the end of lab, even if incomplete.)

How to submit

Submit your Definitions window (`.rkt` file) for this lab using `~st10/111submit` on nrs-labs, with a homework number of `84`.

Purpose

To practice using the Design Recipe to create a function that involves enumeration data.

Important notes

- You are required to work in **pairs** on this lab exercise (unless there are special circumstances, such as an odd number of students in lab, in which case a trio may be approved... 8-)).
  - Remember, in pair programming both people work at one computer: one types ("drives"), while the other tells him/her what to type ("navigates").
  - You may switch roles while working on this lab exercise if you would like -- but I will be trying to see if people are really pair-programming.
  - If you are not pair-programming, then you may not receive full credit for your lab exercise.
- If you have a question during lab, and I am helping another pair, add one or both of your names to the "Next:" list on the board, and I will get to you as soon as I can.

Your tasks:

- Begin a *new* DrRacket **Definitions** window, and type in **comments** containing your names and today's date. Save this in a file with the name `lab4.rkt`
- Put in the expressions to add the `image` and `universe` teachpacks:

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

- Write a data definition COMMENT for an enumeration type `ArrowKey`, noting that its four possible values are the strings `"up"`, `"down"`, `"left"`, or `"right"`:

```
; DATA DEFINITION
; an ArrowKey is a string whose value is either "up", "down",
;     "left", or "right"
```

  - Why? Because based on our CS 111 course coding standards, we can then consider `ArrowKey` to be a new datatype, and we are then "allowed" to use it in function signatures... 8-)
- Then -- decide how many pixels vertically that a single push of an arrow key should change a current

world number. Define a named constant `HOP-AMOUNT` to be that value.

- Using the design recipe, now write a function `shift-penguin` that expects the current world number and a string representing an `ArrowKey`, and:
  - if the key is `"up"` it should return the result of subtracting `HOP-AMOUNT` from the current world number,
  - if the key is `"down"` it should return the result of adding `HOP-AMOUNT` to the current world number
  - and if any other key or string is given, it should return the current universe number unchanged.
  - Be careful to include at least the minimum number of tests one should have for this kind of data! AND WRITE THE TESTS *BEFORE* WRITING YOUR FUNCTION'S BODY!
  - ALSO: don't make this function "more" than it is -- its purpose is JUST to "ADJUST" the world value based on the key given -- that's all!
- When your function is completed and passes its tests:
  - copy the penguin-related named constants and function `draw-penguin` from Week 4 Lecture 2 (under "In-class Examples" from the public course web site)

    ...and make sure all of `draw-penguin`'s tests still pass!
  - NOW try this out, including your `shift-penguin` function:

```
(big-bang 0
          (to-draw draw-penguin)
          (on-tick add1)
          (on-key shift-penguin))
```

  - ... and while it is running, try typing the `"up"` and `"down"` arrow keys, and see how the penguin's position changes as a result!
  - (feel free to modify your `HOP-AMOUNT`'s value if you would like -- that's a benefit of using a named constant, you can change that one amount to "tweak" how much the penguin's position changes...)
- When you are happy with the above (OR at the end of lab), save your Definitions window to the U: drive, use `ssh` to connect to nrs-labs, and use `~st10/111submit` with a homework number of **84** to submit your completed lab exercise.
  - Make sure you BOTH have a copy, also!