Scheme Lab 00000001

Create the following functions, and also provide tests to prove that the functions work. (You do not need to create tests for any helper functions that you create along the way.)

For tests, you may use RackUnit if you wish. If you want to go with something simpler, you can create tests like:

; #|

; Tests for imaginary-function:

(display "(imaginary-function 2 7) -> 79: ")

(imaginary-function 2 7)

(display "(imaginary-function 1 3) -> -2: ")

(imaginary-function 1 3)

(display "(imaginary-function 2 0) -> 1: ")

(imaginary-function 2 0)

(display "(imaginary-function 0 2) -> -1: ")

(imaginary-function 0 2)

; |#

You do not need to catch or test exceptions. Well-formed tests should make it obvious which test has failed.

I do not want *many* tests. The rule is:

* 1 test for each general case (since there is no branching on this lab, there should only be one of these for each function, though that will not be the case for future labs.)
* 1 test for each edge case. (Think about 0s, negative numbers, nulls, etc.)

Please submit all responses to your Dropbox in a file called lab01.rkt.

*Warm-up: Lines*

1. (Extremely Easy) Make a function, y-intercept that takes two arguments, m, and b, and determines the y-intercept of a line.

2. (Easy) Make a function, x-intercept, that takes two arguments, m, and b, and determines the x-intercept of a line.

3. (Medium-Easy) Make a function, triangle-area, that takes two arguments, m, and b, and determines the area of the triangle defined by the y-axis, the x-axis, and the line formed by m and b.

*Moving Along: Mod-circles*

4. (Medium) Consider the function:

(define (modadd low high current addnum)

This function will return the result of adding addnum to current within a modular circule from low to high. You may assume that current is within the bounds of low to high (inclusive). Here are some sample inputs and outputs:

(modadd 4 7 5 0)

(modadd 4 7 5 1)

(modadd 4 7 5 2)

(modadd 4 7 5 3)

(modadd -2 1 0 0)

(modadd -2 1 0 1)

(modadd -2 1 0 2)

(modadd -2 1 0 3)

5

6

7

4

0

1

-2

-1

5. (Medium (possibly Medium-Hard)) Consider the function:

(define (modsub low high current subnum)

This function will return the result of subtracting subnum from current within a modular circule from low to high. You may assume that current is within the bounds of low to high (inclusive). *A word of warning! Depending on what function you used to find your modulus, Scheme will either calculate the lowest positive congruence, or might delve into negative numbers (like Java.* Here are some sample inputs and outputs:

(modsub 4 7 5 0)

(modsub 4 7 5 1)

5

4

*Final Challenge: Moving Clocks.*

6. (Hard) Make a function called addtime that takes a properly formatted hour and minute int (such as 130 or 1245) and a number of minutes to add. It should return a properly formatted int with the new time returned.

(define (addtime time mins)

... you do the rest

Here are some sample inputs:

(addtime 1230 1)

(addtime 1230 2)

(addtime 1230 29)

(addtime 1230 30)

(addtime 1230 91)

(addtime 545 60)

(addtime 545 135)

And their proper outputs:

1231

1232

1259

100

201

645

800

Create helper functions along the way. Doing all of this within one function would end up with crazy, unmaintainable (and unreadable) code!

If you get stuck, here are a few hints:

1. Try to break the problem down into smaller and smaller sections that can be described as functions. What are the smallest functions that you can create?
2. Name your functions carefully. It is very easy to get lost in a sea of functions calling functions if they don’t have very descriptive names.
3. Ask for help early! Particularly on this first lab, I am more concerned with you getting a “feel” for this language than I am with helping each other. For the first three problems, you can freely show each other your code. On problems 4, 5 and 6, you may not look at anyone else’s code until you have completed the problem yourself. At that point, you may help them try to figure things out, but you may not guide them towards your answer. Try to understand the approach that they are taking and help guide them to the next step.

Functional languages are a joy once you get to know them. I hope you have fun with this lab. Now go forth, and good luck!