SI 413 Fall 2012: Homework 2

Your name:

Due: Friday, 31 August, before class

Instructions: Review the course honor policy for written homeworks.

This cover sheet must be the front page of what you hand in. Fill out the left column in the table to the right after we go over each problem in class, according to the rubric below.

This rubric is also on the website, in more detail, under "Other Stuff"→"Grading Rubrics".

Make sure all problems are submitted IN ORDER.

- **5**: Solution is completely correct, concisely presented, and neatly written.
- **4**: The solution is mostly correct, but one or two minor details were missed, or the presentation could be more concise.
- **3**: The main idea is correct, but there are some significant mistakes. The presentation is somewhat sloppy or confused.
- **2**: A complete effort was made, but the result is mostly incorrect. There may be some basic misunderstandings of the topic or the problem.
- **1**: The beginning of an attempt was made, but the work is clearly incomplete.
- **0**: Not submitted.

Problem	Self-assessment	Final assessment
1		
2		
3		
4		
5		
6		
7		

Comments or suggestions about this homework:

Comments or suggestions about the course so far:

Citations (other students, websites, . . .):

Note: Many of these exercises are programming exercises, but you do not need to submit them electronically. Everything should be turned in in one packet, all printed out for me to see.

1 List Basics

a) Using only `cons`, `null`, `car`, and `cdr`, write an expression to produce the nested list
`'(3 (4 5) 6)`.

b) Write a simple quoted expression that is equivalent to
`(cons (cons 3 (cons 'q null)) (cons 'a null))`.

c) Using only `cons`, `null`, `car`, and `cdr`, write a function `(get2nd L)` that takes a list `L` and returns the second element in the list.

2 Split Digits

Write a recursive function `split-digits` that takes a number `n` and returns a list with the digits of `n`, in reverse.

For example, `(split-digits 413)` should produce the list `'(3 1 4)`.

3 Append

Write a function called `my-append` which has the same behavior as the `append` function built-in to Scheme. (Your function only needs to handle the case when there are exactly two arguments, and both are lists.)

For example, calling `(my-append '(a b c) '(d e))` should produce `'(a b c d e)`.

4 Count Down

Write a function called `count-down` which takes a positive integer `n` and produces a list with the integers from n down to 1, in that order.

For example `(count-down 4)` should produce the list `'(4 3 2 1)`.

5 Symbol Mixup

Write a function `(mixup x)` that takes an argument `x`, which can be any symbol or any number, and produces the opposite type of thing, either the number 5 if `x` is a symbol, or the symbol `'num` if `x` is a number.

For example, `(mixup 20)` should produce `'num`, and `(mixup 'hello)` should produce `5`.

6 Nested Quotes

When you type `5` into the interpreter, it returns `5`.

When you type `(quote 5)`, it still returns the number `5`.

But when you type `(quote (quote 5))` or `''5`, it returns `'5`.

What do you think is going on here? Why do you need two quotes to make the symbol 5?

(Caution: this is pretty tricky. Think about how evaluation works. Play around, experiment, discuss.)

7 Building Blocks

In the C programming language, give an example of each of the following types of code fragments.

a) An atom (or literal)

b) A value that is not an atom

c) An expression that is not a value

d) A statement that does not end in a semicolon