SI 413 Fall 2021: Homework 2

Due Tuesday, September 7

Your name:

Citations and collaborators:

Comments, suggestions, or questions for your instructor:

Fill out the first row of the table on a 0-5 scale before turning in.

This rubric is also available on the website under "Admin":

- **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 better.
- **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.
- **1**: The beginning of an attempt was made, but the work is clearly incomplete.
- **0**: Not submitted.

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

Note: Some 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 Bytecode

In class we talked about the difference between compiled and interpreted languages. Many modern languages such as Java and Python take a compromise approach: they partially compile source code to an intermediate representation called *bytecode*, and then this bytecode is interpreted at runtime.

What are some advantages and disadvantages of this approach compared to fully compiled languages such as C++?

2 Reversible Compilation?

For each of these main steps in the compilation process, explain whether that step is always reversible. For example, the scanning phase reads in source code and produces a token stream. Is it always possible to reverse this process and produce the original source code from the token stream? Briefly explain why or why not.

a) Scanning (source code to token stream)

b) Parsing (token stream to parse tree)

c) Semantic analysis (parse tree to abstract syntax tree)

d) Code generation (AST to machine code)

3 List Basics

a) Using only `cons`, `'()`, `car`, and `cdr`, write a Scheme 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 '())) (cons 'a '()))`.

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

4 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)`.

5 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)`.

6 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)`.