Your name:

Due: Wednesday, August 28

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 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. 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.

Comments or suggestions for the instructor:

What other students did you work with?

Citations (be specific about websites):
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 Non-Functional Programming

In class we talked about the three features that typically define a functional programming language: referential transparency, functions as first-class objects, and types as first-class objects.

Show that Java is not a functional programming language by means of a small code example. Your example should be valid Java code that demonstrates Java does not have referential transparency. Briefly state why your code demonstrates this fact.

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)
3 List Basics

a) Using only `cons`, `'( )`, `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

```lisp
(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).