Spring, 1992 Name: ____________________

CS 342 — Principles of Programming Languages

Chapter 1 Test (Version 1)

Before you begin, please take a moment to look over the entire test so that you can budget your time.

Short answer questions will be graded on the basis of completeness (did you list all the relevant parts of the answer?) soundness (did you make any errors of fact or contradict yourself?) and clarity. You can help ensure clarity by giving examples where possible.

For essays, critical justifications, and the like there may be no one right answer; so the criteria for judgement will be whether your essay is:

- clear (you use examples where appropriate, you write clear sentences, diagrams, and you avoid excess verbosity),
- sound (you start from facts or reasonable assumptions, your logic is convincing), and
- complete (you consider relevant aspects of the problem and alternatives).

The highest grade will be earned by an answer that is, in addition, concise and elegant; that is, answers that say the most with the least amount of excess. Extra consideration will be given to answers that are especially creative. For programming problems, there is usually more than one right answer, hence it is important to present with your program evidence that it is correct. Answers to programming questions will be scored on the following points:

- elegance and clarity (conciseness, using relevant techniques to modularize the solution, making good use of language features, the creativity of your solution, and how clear and convincing your argument is that the code is correct)
- correctness (whether your program solves the given problem correctly in the given programming language)

You should try to eliminate all syntax errors from your programs, as they will affect both the correctness and clarity of your solution; you may receive no points for programs with major syntax errors. Run-time space and time efficiency of programs is not important, unless the problem says so. Also, checking of external inputs, prompting, labeling output, and other conventions for dealing with humans are not important (for this class), unless the problem says so.

The breakdown of points between these categories is roughly as follows.

- 60% for elegance and clarity (using relevant techniques to modularize the solution, making good use of language features, the creativity of your solution),
- 40% for correctness (whether your program solves the given problem correctly in the given subset of the given language)

You are encouraged to use auxiliary subroutines whenever they are helpful.

Please show any intermediate steps in computations or calculations for which you want partial credit.

1 Terms

1. (3 points) Define an environment.

2. (3 points) How are environments are used in the language of chapter 1?

3. (3 points) How are environments of the chapter 1 language implemented in the interpreter?

4. (6 points) Briefly define the term actual parameter and give an example in the language of chapter 1.

5. (6 points) Briefly define the term formal parameter and give an example in the language of chapter 1.

2 Problem Solving

The following are to be solved in the language of Chapter 1. You may define auxiliary functions whenever they help you.

1. Write a function `quad-eval` such that `(quad-eval` a b c x`)` returns $(a \times x^2) + (b \times x) + c$.

2. Write a function `T` such that

$$T(n) = \begin{cases} (3n+1)/2, & \text{if } n \equiv 1 \pmod 2, \\ n/2, & \text{if } n \equiv 0 \pmod 2. \end{cases}$$

   which for $n > 0$ takes odd n to $(3n+1)/2$ and even n to $n/2$. You may use the following function in your solution.

```
(define mod (m n) (- m (* n (/ m n))))
```

3. Write a function, `T-iterate` such that if $k \geq 0$, `(T-iterate` n k`)` is $T^{(k)}(n)$; for example, $T^{(2)}(n)$ is $T(T(n))$. More formally, the iterates of the function T, are mathematically defined as follows. Let $T^{(0)}(n) = n$, and for all integers $k > 0$, let $T^{(k)}(n) = T(T^{(k-1)}(n))$. Your code should call a function `T` to compute T; thus you need *not* to solve the previous problem to solve this problem.