Fall, 1993 Name: ______________________

Grading TA's name: ______________________ Section Day and Time : ____________

Com S 227 — Introduction to Computer Programming

Test on Chapters 3 and 4 (Version 1)

This test has 4 questions and pages numbered 1 through 5.

Reminders

This test is closed book and closed notes.

If you need more space, use the back of a page. Note when you do that on the front.

This test is timed. We will not grade your test if you try to take more than the time allowed. Therefore, before you begin, please take a moment to look over the entire test so that you can budget your time.

Indentation is important to us for "clarity" points; if your code is sloppy, you will lose points.

Subset of Scheme You May Use to Define Other Procedures

Unless otherwise stated, when defining procedures you may only use: helping procedures that you define yourself, and the procedures and keywords that are included in the following list. (That is to say, you may *not* use Scheme procedures and keywords that are absent from the following list.) The notation `c...r` means `caar`, `cadr`, `cddr`, `cdar`, `caaar`, etc. You may only use `define` at the top level.

```
#t, #f, *, +, -, /, <, <=, =, >=, >,
abs, acos, add1, and, append, asin, atan, atom?, atomic-item?, begin, boolean?,
car, cdr,  c...r, ceiling, cond, cons, cos, define, denr
else, eq?, equal?, eqv?, error, exp, expt, floor, if, integer?,
lambda, length, list, list-ref, log, make-ratl, max, min, modulo,
negative?, not, null?, numr, number?, or, pair?, positive?, procedure?,
quote, quotient, real?, remainder, reverse, round, sin, sqrt, sub1, symbol?,
tan, truncate, writeln, zero?
```

Directions from the "Course Specification"

You are encouraged to use helping procedures.

For programming problems, there is usually more than one right answer, hence it is important to write a clear solution. Answers to programming questions will be scored with the following breakdown of points:

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

You should try to eliminate all syntax errors from your programs, as they affect both the correctness and clarity of your solution; you may receive no points for programs with major syntax errors. Checking of external inputs, and other conventions for dealing with humans are not important (for this class), unless the problem says so.

1. (5 points) Define a procedure `checked-sqrt` that takes a number `n` and returns its square root. For example,

```
(checked-sqrt 4) ==> 2
(checked-sqrt 16) ==> 4
(checked-sqrt 2) ==> 1.4142135623730951
(checked-sqrt 0) ==> 0
```

(Hint: look at the list of procedures you can use on the previous page.) The procedure `checked-sqrt` must give an error message when its argument is negative. That is, when `n` is negative `checked-sqrt` signals the error

```
Error: in checked-sqrt: the argument is negative.
```

and has no specified return value. For example, `(checked-sqrt -4)` signals this error.

2. (15 points) This is a problem about the rational number type "ratl". Recall that the basic procedures for the type ratl have the following types.

`make-ratl`	: ($\rightarrow$ (integer integer) ratl)
`numr`	: ($\rightarrow$ (ratl) integer)
`denr`	: ($\rightarrow$ (ratl) integer)

(Don't ask us what these do, you're supposed to know that.) If you need other procedures on ratls, you'll have to write them yourself (that is part of this problem).

Write a procedure `r-sphere-vol` that takes a ratl, `r`, representing the radius of the sphere, and returns its volume as a `ratl`. Using 355/113 as an approximation to π, the formula for the volume of a sphere is,

$$(4/3) \times (355/113) \times (\mathtt{r}^2).$$

Recall that $(a/b) \times (c/d) = ((a \times c)/(b \times d)$. For example, using the procedure `rprint` defined in the book to show answers:

```
(rprint (r-sphere-vol (make-ratl 1 1))) =prints=> 1420/339
(rprint (r-sphere-vol (make-ratl 2 3))) =prints=> 5680/3051
(rprint (r-sphere-vol (make-ratl 5 1))) =prints=> 35500/339
(rprint (r-sphere-vol (make-ratl 8 9))) =prints=> 90880/27459
```

You must use the basic procedures for ratls correctly in your solution. Your code should be independent of the representation of ratls; that is, low level solutions will lose points. (Don't ask us what "representation independent" or "low level" means, you're supposed to know that.)

3. (20 points) Define a procedure `tree-add` that takes two trees of numbers, `t1` and `t2` which have the same shape, and returns a tree with the same shape whose leaves are the sums of the corresponding leaves of `t1` and `t2`. (Don't ask us what a "tree" is, you're supposed to know that.) For example,

```
(tree-add '() '()) ==> ()
(tree-add '(() ()) '(() ())) ==> (() ())
(tree-add '(1 2 3) '(4 5 6)) ==> (5 7 9)
(tree-add '((1 2) (3 4)) '((10 20) (30 40))) ==> ((11 22) (33 44))
(tree-add '((1 7 8) (5 9 (6 (2 ()) 3)) ((4)))
          '((2 5 9) (7 1 (5 (2 ()) 3)) ((4))))
       ==> ((3 12 17) (12 10 (11 (4 ()) 6)) ((8)))
```

Assume that the trees have the same shape. That is assume that, if one is empty then so is the other, if `(car t1)` is a number then so is `(car t2)`, and if `(car t1)` is a pair, then so is `(car t2)`, etc. For example, `(2 () 3)` and `(9 () 11)` have the same shape, but `(2 () 3)` and `((4) (5))` do not.

4. (15 points) This is a problem about iteration. Define an *iterative* procedure `count-matching` that takes a symbol, `item`, and a list of symbols, `ls`, and returns the number of top-level items in `ls` that are the same as `item`. For example,

```
(count-matching 'a '(a one and a two and a three)) ==> 3
(count-matching 'me '()) ==> 0
(count-matching 'me '(me me me and again I say me)) ==> 4
(count-matching 'time '(what time is it no time to sit)) ==> 2
(count-matching 'plant '(a plant at the plant is no plant of mine)) ==> 3
```

Only solutions that are iterative will get points. Non-iterative solutions will get zero (0) points. (Don't ask us what "iterative" or "top-level" mean, you're supposed to know that.)