CISC280 Midterm Learning Experience 2, Spring 2006

NAME _____

General instructions:

Turn off all noise-making electronic devices, such as cell phones. Disturbance by such a device during the exam may result in a penalty not to exceed a full letter grade.

You may leave the classroom once the exam has begun, but you may not return.

There are 3 pages worth a total of points, and an extra credit problem worth 3 points. Read the problems very carefully. Identify what kind of answer the problem asks for. If writing a procedure, carefully look at requirements for input and output, and any restrictions on how it must be written or which other procedures may be used.

You may assume a list argument will be flat unless it is otherwise specified, and that the elements will not produce errors for the procedures described.

Do not do unnecessary testing. For example, testing for both list? and null? instead of using one test and then else would be considered unnecessary testing.

Do not make code unnecessarily inefficient. An extra procedure call here or there is ok, but do not make an $O(n)$ problem into $O(n^2)$.

Do problems you are confident about first. If you finish the problems you know, write what you do know about other problems to gain partial credit; but erroneous information may detract from that credit, so don't make stuff up.

You may use any of the Scheme primitives we use in class.

1. (5 pts) Define two lists and draw them using box and pointer notation. Then show the box and pointer list that would result from joining the lists with `append`.

2. (10 pts) Write the procedure **intersection** for an ordered list representation of a set.

3. (6 pts) What are the three methods shown in class to evaluate a sequence of expressions in Scheme?

4. (6 pts) Define a fcn last-pair that walks through a proper list and returns the terminating pair.

5. 14 pts. Assume *all* the following expressions are evaluated **in order**. Fill in the blanks with the value or message printed by the interpreter. If there is an error or other Scheme message, you do not have to be exact.

```
> (define (f x)
     (set! x (/ x 2))
     x)
> (define a 80)
> (f a)

1a. _____

> (f a)

1b. _____

> (define b 50)
> (define (g x)
     (/ x 2)
     x)
> (g b)

1c. _____

> (g b)

1d. _____

> (define (h x)
     (define (k)
        (set! x (/ x 2))
        x)
   k)
> (h 100)

1e. _____

> (define k (h 100))
> (k)

1f. _____

> (k)

1g. _____
```

6. (20 pts) What is O() for:

7. (20 pts) Demonstrate the transformation of a “let” into a procedure.

8. (20 pts) Define data abstraction. Describe reasons to include it in software design.

9. (30 pts) Compare and contrast data-directed programming, message-passing, and dispatching-on-type. Explain their advantages and disadvantages when used to implement generic operations. In particular, what transactions are relatively easy or hard, and why?

10. (20 pts) Write code to simulate a bank account with features for deposit, withdraw, and password checking. Account variables must be local.

11. (10 pts) Consider the problem ”Give the big O (time) for union of two sets.” Is this question meaningful? Explain. How would you go about answering it?

12. (10 pts) Comment on this quote from the GNU C tutorial:

> The assignment operator: No operator such as addition (+) or multiplication (*) would be useful without another operator that attaches the values they produce to variables. Thus, the assignment operator = is perhaps the most important mathematical operator.