CISC280 Midterm Learning Experience 2, Spring 2007

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 5 pages worth a total of 120 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 simple procedures we use regularly in class, and the following procedures (unless you are asked to define them):
apply
append
reverse
map
accumulate
filter
enumerate
eq?
equal? (only if needed)
member

1. (10 pts) Write the tail recursive procedure repeat-proc (similar to the one you've used in lab) that takes a **count**, a **procedure**, and an **argument** and calls the procedure on the argument **count** times. The procedure will take only one argument. Repeat-proc is for timing runs, so do not introduce extraneous code. This example squares the argument 2 five times.

```
> (define (square x) (* x x))
> (repeat-proc 5 square 2)
()
```

2. (5 pts) Write another version of repeat-proc, but this one takes a *list* of arguments for the procedure. In this example call, all the elements of the argument list are multiplied together five times.

```
> (repeat-proc 5 * '(1 2 3 4))
()
```

3. (4 pts) Fill in the blank so that repeat-proc from 2 will multiply the argument list elements, but will also display them each time it does:

```
> (repeat-proc 5 _____________________________________________ '(1 2 3 4))
(1 2 3 4)(1 2 3 4)(1 2 3 4)(1 2 3 4)(1 2 3 4)()
```

Generic Operations

For the next two questions, assume you are the manager of a widget department. Your company is about to merge with another, and must merge widget departments. You are told:

- All widget departments keep track of the same data (weight, color, gender, etc.), but in different representations;
- You will probably merge with some more companies soon.
- You are in charge of implementing the generic operations that will function on data from multiple widget departments.

4. (4 pts) What are your options for implementation, as discussed in class?

5. (15 pts) Which would you choose? BRIEFLY justify your decision in terms of **1)** the problem description **2)** the features of your chosen method, and **3)** disadvantages of methods you do not choose.

6. (2 pts) Fill in the blank: Tower and graph hierarchies enable type systems to (sometimes)

   perform _____.

7. (4 pts) Tower and graph hierarchies are supposed to help address a problem that occurs in generic type systems as they grow, as discussed in class. What was the problem? Be specific.

8. (6 pts) Assume there are multiple representations of complex numbers in a data-directed system. Write a high-level **real-part** procedure for a data-directed programming system that takes a single argument of any representation.

9. (12 pts) Write a message-passing constructor for a rational number. The rational should respond to messages to return the numerator or denominator, and to multiply itself with another rational number. **Do not do any type checking** on the argument rational.

10. (5 pts) Assume you have one rational **r1**. Show calls to create another rational and multiply them using the code you've written.

11. (6 pts) How can you test if a program was designed using proper data abstraction? Be concise.

Trees

12. (5 pts) How many nodes do we expect in a balanced binary tree of height 9? Show your calculations. Show your answer as an integer.

13. (5 pts) How many nodes do we expect in the **bottom row** of a balanced binary tree of height 22? Show your calculations.

14. (5 pts) Assume a tree of 185 nodes is as balanced as it can be. What is the height of the tree? Show your calculations. Show your answer as an integer.

15. (5 pts) Consider the procedure adjoin-tree that adds one element to a binary search tree. What is big O? Explain.

16. (6 pts) The procedures `tree->list` and `list->tree` can be used to improve the performance of union-tree. Explain how.

17. (5 pts) Xenon is proposing using `tree->list` and `list->tree` to improve the performance of adjoin-tree. Explain your reaction to this proposal.

18. (8 pts) Assume you are given procedures **make-tree**, **entry**, **left**, and **right**. Write the procedure **sum-tree**, a tree-recursive procedure which sums all the elements in a tree.

19. (4 pts) Suppose a tree contains K nodes. What is the big O of sum-tree in terms of K? Explain.

20. (4 pts) Would you expect the typical run time to be better than, worse than, or the same as big O predicts? Explain.