NAME & ID#: ________________________________

Instructions

1. This is an open-notes, open-book, open-computer, open-Internet exam.
2. The exam is due in class on November 10, 2008.
3. **You can work on this exam as much as you like. There is no time limit.**
4. Please make sure you have all 5 pages of this exam.
5. You will not be penalized on trivial syntax errors, such as a missing parenthesis. Multiple errors or errors that lead to ambiguous code will have points deducted, however.
6. In all of the questions, feel free to write additional helper methods to get the job done.
7. The emphasis is on correctness of the code, not efficiency or on simply generating the right result.
8. You are free to use any code that was given to you in the lectures and labs, or that you wrote in this exam.
9. You do not have to write generic code in this exam.

Please write and sign the Rice Honor Pledge here:

1a	1b	Total
/10 pts	/10 pts	/20 pts

Lazy Evaluation:

a) (10 pts) **You are to write an `ALazyEval` called `LazyNCopiesEval` that will create a finite (ending!) list using the lazy list framework. `LazyNCopiesEval` takes an `Object o` and an `int n`, and produces a list of length `n` filled with the object `o`.**

- If n is 0 or less, the list created by `LazyCombineEval` is empty.
- You may use an if statement to deal with `n>=0`.
- Example:
    - o=1, n=0 → ()
    - o=1, n=1 → (1)
    - o=1, n=2 → (1, 1)
    - o=1, n=3 → (1, 1, 1)

Notes:

a. Think delegation always.
b. There are no `if` statements needed, unless you are working with primitive data (ints, doubles, etc.)!
c. Don't forget about the possibility that the source lists may be or become empty.

Please insert your code for `LazyNCopiesEval.java` below:

```
package lrs.lazyLRSEvaluators;

import lrs.*;

/**
 * Given an element e and a number n, creates a list of length n,
 * containing elements of e.
 */
public class LazyNCopiesEval extends ALazyEval {
    public LazyNCopiesEval(Object e, int n) {
      _e = e;
      _n = n;
    }

    // FOR STUDENT TO COMPLETE: add fields and methods here
```

Lazy Evaluation:

b) (10 pts) **You are to write an `ALazyEval` called `LazyListNCopiesEval` that will create a possibly infinite. `LazyListNCopiesEval` takes two `LRStructs`: the first `LRStruct li` contains `Integers` (i1, i2, i3, …), the second `LRStruct lo` contains `Objects` (o1, o2, o3, …). The lazy list created by `LazyListNCopiesEval` contains i1 copies of o1, i2 copies of o2, i3 copies of o3, and so on. If one if the integers is 0, then the corresponding object will be skipped.**

- Example:
  - li = (), lo= () → result = ()
  - li = (1), lo= () → result = ()
  - li = (), lo= (1) → result = ()
  - li = (1), lo= (1) → result = (1)
  - li = (2), lo= (1) → result = (1, 1)
  - li = (3), lo= (1) → result = (1, 1, 1)
  - li = (1, 1), lo= (1) → result = (1)
  - li = (1), lo= (1, 1) → result = (1)
  - li = (1, 2), lo= (1, 2) → result = (1, 2, 2)
  - li = (0), lo= (1) → result = ()
  - li = (0, 1), lo= (0, 1) → result = (1)
  - li = (0, 1, 0), lo= (0, 1, 0) → result = (1)
  - li = (1, 1, 1, …), lo= (1, 2, 3, …) → result = (1, 2, 3, …)

Please insert your code for `LazyListNCopiesEval.java` below:

```
package lrs.lazyLRSEvaluators;

import lrs.*;
import fp.*;

/**
 * Takes two lists, one list of integers Li(i1, li2, li3,...) and one list
 * of objects Lo(o1, o2,...). It lazily creates a list that contains i1
 * elements of o1, followed by i2 elements of o2, and so on.
 * If one of the two lists Li or Lo ends, this list ends.
 */
public class LazyListNCopiesEval extends ALazyEval {
    /** Constructs a LazyListNCopiesEval, which creates a lazy list.
      * @param li input list with integers
      * @param lo input list with objects
      */
    public LazyListNCopiesEval(LRStruct li, LRStruct lo) {
        this.li = li;
        this.lo = lo;
    }

    // FOR STUDENT TO COMPLETE: add fields and methods here
```