Fall, 1999 Name: _____

My Grading TA: _____ My Section Day and Time : _____

Com S 342 — Principles of Programming Languages

Test on *SICP* Sections 1 to 1.3.2

This test has 7 questions and pages numbered 1 through 7.

Reminders

This test is closed book and 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.

For programs, indentation is important to us for "clarity" points; if your code is sloppy or hard to read, you will lose points. Correct syntax also matters. Check your code over for syntax errors. You will lose points if your code has syntax errors.

Of course, you may write helping procedures or methods whenever you wish. It may be helpful to give a comment that describes what they do, if it's not completely clear from the name.

You may assume that all Java code is in the same package on this test.

1. (10 points) For each of the following, put a check mark ($\surd$) in the appropriate column to say whether the language feature in its row is: a means of computation, a means of combination, or a means of abstraction.

	computation	combination	abstraction
linked list data structures			
interface declarations in Java			
floating point numbers			
if-expressions in Scheme			
procedure declarations			

2. (10 points) Briefly answer the following question. Why does `if` in Scheme have to be a special form instead of a procedure?

3. (10 points) Briefly answer the following question. Why does Scheme use Cambridge prefix syntax instead of a more normal notation?

4. (10 points) Translate the following Java expressions into Scheme.

   (a) `x + 7*y - 4`

   (b) `f(x - 2, g(y - 3 + 4), h(z))`

5. The following relates to a problem you will solve both in Scheme and Java. If you invest a principal of p dollars in a retirement account for n years, and on the average it earns interest at the rate i per year (expressed as a fraction), then with compound interest, at the end of the n years you have earned $p \cdot (1+i)^n - p$ dollars.

   (a) (10 points) Write a procedure `earnings` in Scheme that takes p, n, and i as arguments and computes the value of your earnings after n years at the interest rate i. Your Scheme code may *not* use the built-in Scheme `expt` function; instead, write it using only the following built-in procedures of Scheme:

   ```
   * + - = zero? < <= > >=
   ```

   and any helping procedures you wish to write. For example,

   ```
   (earnings 10000.0 0 0.10)  ==>       0.0
   (earnings 10000.0 1 0.10)  ==>    1000.0
   (earnings 11000.0 1 0.10)  ==>    1100.0
   (earnings 10000.0 2 0.10)  ==>    2100.0
   (earnings 10000.0 10 0.10) ==>   15937.424601
   (earnings 10000.0 40 0.10) ==>  442592.555681761
   (earnings 10000.0 50 0.10) ==> 1163908.52879696
   (earnings 10000.0 25 0.20) ==>  943962.16644069
   ```

(b) (10 points) Solve this problem (earnings, see the previous page) in Java by filling in the following code outline. Your Java code may *not* use the built-in `Math.pow` method; instead, write it using only standard expressions of Java.

```
public class Earnings {
   public static double earnings(double p, double n, double i) {









   }
}
```

6. (20 points) Consider the "$3x + 1$" function, t, whose value on a nonnegative integer argument x is as follows:

$$t(x) = \begin{cases} x/2 & \text{if } x \text{ is even,} \\ 3 \cdot x + 1 & \text{otherwise} \end{cases}$$

(Note, in Java, like C and C++, `x % 2 == 0` is true if `x` is even.)

In Java, define a public class `Transform` that implements the `LongIntFun` interface given below.

```
public interface LongIntFun {
    long val(long x);
}
```

Write your code for the class `Transform` below.

7. The following relates to a problem you will solve both in Scheme and in Java. The i^{th} repetition of a function f, f^i, is defined by the following, for $i \geq 0$ and for all arguments x.

$$f^i(x) = \begin{cases} x & \text{if } i = 0 \\ f(f^{i-1}(x)) & \text{otherwise} \end{cases}$$

For example, $g^0(342) = 342$ and $g^3(342) = g(g(g(342)))$.

(a) (10 points) Write a tail-recursive procedure, `repeat` in Scheme such that `(repeat f i x)` computes $\texttt{f}^{\texttt{i}}(\texttt{x})$. You may assume that `i` is nonnegative.

For example,

```
(repeat (lambda (x) (* x 2)) 0 1) ==> 1
(repeat (lambda (x) (* x 2)) 3 1) ==> 8
(repeat (lambda (x) (* x 2)) 4 1) ==> 16
(repeat (lambda (x) (+ x 10)) 5 9) ==> 59
```

Remember, your procedure must generate an iterative process. You may use helping procedures to do this, of course.

(b) (15 points) Solve this problem, repetition of a function, in Java using a while loop. Your code should implement the following interface.

```
public  interface RepeatProblem {
    /** Compute f.val^i(x).
     *  For example, if f.val(x) = 2*x, i is 3, and x is 1,
     *  then this returns 8.  You can assume that i >= 0.
     */
    long repeat(LongIntFun f, int i, long x);
}
```

Note that the above interface uses the interface `LongIntFun`, which is defined in problem 6.

Write your code for the public class `Repeat` that implements the `RepeatProblem` interface below.