Name:_____

Exam 1

Foundations of Computer Science
Fall 2007

This exam is **open book, open notes, open computer**. You may use your computer to access general documentation, but not to search for answers to questions on this exam. You may use the Scheme and Python interpreters to test code snippets, but I strongly discourage you from trying to write and test your solutions unless you have time to spare.

Note: in each section, I have tried to write at least one question that is relatively easy, and at least one that is moderately challenging. So if you find something easy, you shouldn't (necessarily) worry that you are missing something.

1 Regular expressions/languages (24 points)

1. Draw a state diagram for a DFA with alphabet $\{0, 1\}$ that recognizes

$$\{w | \ w \text{ doesn't contain the substring } \texttt{110}\}$$

2. Draw a state diagram for an NFA that recognizes `((00)*(11))` $\cup$ `01`.

3. The reverse of string w, denoted w^R, contains the symbols from w in reverse order. For a language A, let $A^R = \{w^R \mid w \in A\}$. Show that if A is regular, so is A^R.

4. A palindrome is a word that reads the same forward and backward. Let B be the language that contains all and only palindromes for a given alphabet. If you think that this language is regular, sketch a proof that it is. If you think it is not regular, sketch a proof that it is not.

2 Scheme (18 points)

1. One way to represent arbitrary-precision numbers is using a list of digits in reverse order. For example, the list `(3 2 1)` might represent the decimal number 123.

   Write a Scheme function named `bignum-to-int` that takes a list of integers as an argument and returns the corresponding integer.

   Hint: storing the digits in reverse order lends itself to a recursive implementation; for example, notice that $123 = 12 \cdot 10 + 3$.

2. One way to represent a polynomial is with a list of coefficients in reverse order. For example, the list `(3 2 1)` might represent the polynomial $x^2 + 2x + 3$.

   Write a Scheme function named `poly` that takes a list of integers as an argument and returns a function that evaluates the corresponding polynomial. Here is an example of how `poly` should work:

   ```
   > ((poly (list 3 2 1)) 2)
   11
   ```

   Hint: storing the coefficients in reverse order lends itself to a recursive implementation; for example, notice that $x^2 + 2x + 3 = x(x + 2) + 3$.

3. Every list of lists can be regarded as a tree, but not every tree can be regarded as a list of lists. Draw a box and pointer diagram of a tree that is not a list of lists.

4. Draw a box and pointer diagram for the following expression.

```
(cons 1 (cons (cons 2 '()) (list 3)))
```

3 Lambda calculus (18 points)

1. Reduce the following lambda expression to the simplest form:

$$(\lambda xy.xy)(\lambda x.x)(\lambda x.x)$$

2. Reduce the following lambda expression to the simplest form:

$$(\lambda xy.yx)(\lambda x.fx)(\lambda x.xx)$$

3. Using lambda calculus syntax, write a definition for NAND. You can use TRUE and FALSE, but you can not use the other logical operators.

4. Using lambda calculus syntax, write a function EXP that takes two Church numerals, m and n as arguments and returns m^n as a Church numeral. You may use any of the functions we defined in the lambda calculus exercise.