CS 311: Computational Structures **Practice Final Exam** **December 2012**

The exam is a closed book exam. Answer questions in the space provided.

Student Name (neatly in block letters): _____

All questions from the midterm and quizzes are good study problems. You should familiarize yourself with these questions and their answers. A version of one of the questions in this practice exam will appear in the actual final exam. The actual final exam will also have a pumping lemma question.

1. **CFG to PDA**

   Consider the Context Free Grammar for a language we will call L:

   $$\begin{array}{rcl} S & \rightarrow & SaSb \\ & | & \Lambda \end{array}$$

   (a) Give a leftmost derivation for the string *abaabb*.

   (b) Construct a PDA M from the grammar that accepts L. (You may assume a generalized version of the PDA that makes multiple stack moves on a single transition.)

   (c) Show that M accepts the string *abaabb*.

2. **Proof by induction**. (20 points) In a earlier homework you showed by induction, that for one representation of the natural numbers, and for one inductive definition of addition, that addition was associative, that is $i + (j + k) = (i + j) + k$. In this problem we use the same representation of the natural numbers and addition, but instead show that the following property holds: for all x the following holds $x + (Sy) = S(x + y)$. Recall the definitions for the natural numbers and addition (infix +).

   - Z is a natural number (zero)
   - if n is a natural number, then $S\ n$ is a natural number (the successor function).
   - Nothing else is a natural number

   ```
   (1)  Z + n = n
   (2)  (S m) + n = S(m + n)
   ```

   Prove by structural induction that for all x the following property holds: $x + (Sy) = S(x + y)$.

   Be sure and structure your proof properly, breaking it into appropriate cases, and clearly stating the induction hypothesis where appropriate.

3. **Turing computable.**

   - State the Church-Turing Thesis.
   - List three computational systems that are Turing complete.
   - Describe a string acceptance problem that is computable by a Turing machine, but not by a context free grammar.

4. **Traces.** Correctness (acceptance) of many computational systems can be defined in terms of the existence of a trace.

   - What important property do traces have?
   - List three of the systems described in the Outline notes, and describe what their traces look like.

- Consider the context free grammars. Outline what a trace for a context free grammar might look like.
- Define a simple CFG, and provide a trace in the form you described above for a short string.

5. **Computability.**

   - What is a universal machine?
   - What important property does a universal machine have?
   - Give an example of a universal machine that we saw in the lecture notes.
   - Are all total functions primitive recursive? If not give an example.
   - What is the halting problem?