COS 487: Theory of Computation	**Fall 2000**

Assignment #1

Due: Monday, October 2 — *Sanjeev Arora*

Suggested reading (for lectures 1,2,3): Sipser Chapter 1.

A hint for this assignment: keep in mind the properties of regular languages. For instance, if you are trying to show that a language L is regular, it suffices to show that $\overline{L}$ is accepted by a nondeterministic automaton.

Problems (from lectures 1, 2, 3):

1. (*This is a practice problem; do not hand it in*) Build a finite automaton that accepts language $L = \{x : x \in \{0,1\}^*$ and is a multiple of 3$\}$. Also write a regular expresssion that describes L.

2. Let L be a regular language. Show that the language L' is also regular, where
$$L' = \{x : \text{ no } w \in L \text{ is a substring of } x\}\,.$$

3. Let L be a regular language. Show that the language $L_{\frac{1}{2}-}$ is also regular, where
$$L_{\frac{1}{2}-} = \{w : \text{ for some } z \in L, x \in \{0,1\}^*,\ z = wx \text{ and } |w| = |x|\}\,.$$

4. Consider a new kind of finite automaton, an *All-Paths-NFA*. The automaton is is defined just like an NFA, except an input x is said to be accepted iff *all* the states that the NFA is in at the end are accept-states. Note, in contrast, that an ordinary NFA is said to accept the string iff *at least one* of the states it is in at the end is an accept-state. Prove that the class of languages accepted by All-Paths-NFA are exactly the regular languages.

5. Describe an algorithm that, given any two finite automata M_1 and M_2, decides whether or not M_1 and M_2 accept the same language. (*Note: you do not need to write pseudo-code. A description in English will do.*)

6. Show that the following language is not regular.
$$L = \{0^p : p \text{ is a prime}\}\,.$$

7. In class we gave a way to convert DFA's into equivalent regular expressions. *(a)* Give a reasonable estimate of how large an expression this may generate from a DFA with n states. (You may, if you wish, ignore the symbols (,), *, and $\cup$ in your answer.) *(b)* We also gave a way to convert regular expressions into NFA's. Give a reasonable estimate of how large an NFA this may generate from an expression with n symbols. Justify your answer.