Introduction to Theory of Computing	**Problem Set 2**
CS 481 Fall 2004	**Due Wednesday, September 15, 2004**

Reading. To review the material covered so far, read sections 1-6.

For each problem set please write both your name and your cornell.edu email address on top.

When designing a DFA or NFA provide the required definitions (a transition diagram is good enough), and also a brief English explanation of the main idea (e.g., what the states represent, etc). In designing NFA you may use ε-transitions defined on page 36 of Kozen, but it is not necessary.

(1) Let $\Sigma = \{0, 1\}$.

(a) In class we gave a nondeterministic finite automata with $k+1$ states that accepts the language $L_k = \Sigma^* \cdot 1 \cdot \Sigma^{k-1}$ (all words where the letter k from the end is a 1). Give a deterministic finite automata that accepts the language L_k. How many states did you use?

(b) Give a deterministic finite automata that accepts the language $L_k = \Sigma^* \cdot 1^k$ (the last k characters are all 1). Do this with less than 2^k states.

(2) Show that for any regular language $L \subseteq \Sigma^*$ the language Even(L) is also regular, where Even(L) is defined as

$$\text{Even}(L) = \{\sigma_2\sigma_4 \ldots \sigma_{2n} | \exists \sigma_1, \sigma_3, \ldots, \sigma_{2n-1} \in \Sigma \text{ such that } \sigma_1\sigma_2\sigma_3\sigma_4 \ldots \sigma_{2\text{n}} \in \text{L}\}$$

(3) Let Σ be an alphabet with k characters. Define the language

$$Incomplete(\Sigma) = \{w \in \Sigma^* | \exists a \in \Sigma \text{ such that w does not contain a}\}.$$

Design a NFA using at most k states that accepts $Incomplete(\Sigma)$.

(4) The *Hamming distance* between two equal length strings x and y is the number of places they differ. We will use $H(x, y)$ to denote this distance. For any language $L \subset \Sigma^*$ we say that $H(x, L) = \min_{y \in L} H(x, y)$ (where the Hamming distance of strings x and y with $|x| \neq |y|$ is infinite. Now let

$$N_k(L) = \{x \in \Sigma^* | H(x, L) \leq k\}.$$

Prove that if L is regular than so is $N_1(L)$. **Hint:** use two copies of the machine M accepting L, and use nondeterministic transitions to guess where the error occurs.