1. (**30 points**) Using the recursion theorem, write a short proof for Rice's theorem.
   Note that Sipser states Rice's theorem in problem 5.28 and gives a proof. Sipser's proof does not use the recursion theorem. Your solution must use the recursion theorem (and your proof should be shorter than Sipser's).

2. (**30 points**, from Sipser problem 5.29) Use Rice's theorem to prove the undecidability of the two languages below:

   (a) (**15 points**) $\{[M] \mid [M]$ describes a TM and $1011 \in L(M)\}$.

   (b) (**15 points**) $ALL_TM = \{[M] \mid [M]$ describes TM and $L(M) = \Sigma^*\}$.

3. (**40 points**) Consider a variation on Post's Correspondence Problem where each pair of strings can be used at most once. We'll call this 1PCP. An instance of 1PCP consists of $k$ pairs of strings, $(x_1, y_1)$, $(x_2, y_2)$, $\ldots (x_k, y_k)$. A solution to 1PCP consists of $m$ *distinct* integers $i_1$, $i_2$, $\ldots i_m$, with $m \leq k$ such that

$$x_{i_1} \cdot x_{i_2} \cdots x_{i_m} \quad = \quad y_{i_1} \cdot y_{i_2} \cdots y_{i_m}$$

   A non-deterministic Turing machine can guess the values for $m$ and $i_1 \ldots i_m$, verify that for $a \neq b$, $i_a \neq i_b$, and verify that the concatenation of the $x$ strings matches the concatenation of the $y$ strings. Clearly, these verification steps can be done in polynomial time. Thus, 1PCP is in NP.

   In this problem, you will show that 1PCP is NP hard and thus NP complete. In particular, show how the Hamiltonian cycle problem can be reduced to $1PCP$. Let $G = (V, E)$ where $V = v_1, v_2, \ldots v_h$, and $E \subseteq V \times V$ be a graph. $G$ has a Hamiltonian cycle iff there exists a permuation $p_1, \ldots p_h$ of $1 \ldots h$ such that for each $i \in 1 \ldots h-1$, there is an edge connecting $v_{p_i}$ and $v_{p_{i+1}}$ (i.e. $(v_{p_i}, v_{p_{i+1}}) \in E$) and there is an edge from $v_{p_h}$ to $v_{p_1}$.

   Show a simple (and it must be polynomial time!) reduction from Hamiltonian cycle to 1PCP.

4. (**50 points**) We can formalize the description of DFAs in a manner very similar to how we formalized the description of TMs.

   (a) (**10 points**) Let $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{,}, \texttt{\#}, \texttt{(}, \texttt{)}\}$ be an alphabet for writing descriptions of DFAs. To describe a DFA, $D = (Q_D, \Sigma_D, \delta_D, q_{0,D}, F_D)$, write the string

$$s_{Q,D}, s_{\Sigma,D}, s_{\delta,D}, s_{q_0,D}, s_{F,D}$$

   where

   $s_{Q,D}$ is the binary string for $|Q_D$.

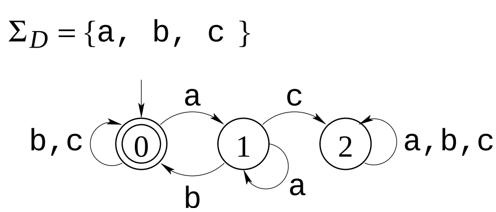
   $s_{\Sigma,D}$ is the binary string for $|\Sigma_D$.

   $s_{\delta,D}$ is a string of tuples of the form $(q, c, q')$ where $q$ is a binary string of length $\lceil \log_2 |Q_D| \rceil$, $c$ is a binary string of length $\lceil \log_2 |\Sigma_D| \rceil$, and $q'$ is a binary string of length $\lceil \log_2 |Q_D| \rceil$. The tuple $(q, c, q')$ indicates that $\delta_D(q, c) = q'$. Furthermore, these tuples are listed in $s_{\delta,D}$ in lexigraphical order.

   $s_{q_0,D}$ is a binary string for the state $q_{0,D}$.

   $s_{F,D}$ is a comma separated list of binary strings representing the states in $F_D$. This list is in ascending order.

   Give the string that describes the DFA below:



(continued on next page)

(b) (**20 points**) Let

$$A_{REG,n,m} \quad = \quad \{D\#w \mid D \in \Sigma^* \text{ describes a DFA with at most } n \text{ states and an input alphabet with at most } m \text{ symbols that accepts the string described by } w\} \;.$$

Note that $\Sigma_D$ may have symbols that are not in $\Sigma$. Thus, to describe an input string for $D$, use a comma separated list of binary strings, where each binary string has length $\lceil \log_2 |\Sigma_D| \rceil$. For example, the string $abbac$ in $\{a, b, c\}^*$ is encoded as $00, 01, 01, 00, 10$.

Show that for any fixed $n$ and $m$, $A_{REG,n,m}$ is a regular language.

(c) (**20 points**) Use a diagonalization argument to prove that any DFA that recognizes $A_{REG,n,m}$ must have more than $n$ states. (Assume $n, m > 0$.)