Problem 1 Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Let $w = w_1w_2 \cdots w_n$ be a string accepted by $M$, for which let $r_0, r_1, \ldots, r_n$ be the corresponding accepting path. (See the section titled “Formal Definition of Computation,” page 40, in Sipser.)

Suppose that $r_i = r_j$ for some $i$ and $j$ such that $i < j$. Prove that, in addition to accepting $w$, $M$ also accepts some string $w'$ that is shorter than $w$, i.e., such that $|w'| < |w|$. 

Problem 2 In class, we showed that swapping the accepting and nonaccepting states of a DFA whose language is $L$ gives a DFA whose language is $\bar{L} = \Sigma^* \setminus L$.

a. Show (by construction) that swapping the accepting and nonaccepting states of an NFA whose language is $L$ does not necessarily give an NFA whose language is $\bar{L}$. 

Hint: There are examples with a very small number of states.

b. Explain, given an NFA whose language is $L$, how to construct another NFA whose language is $\bar{L}$.

Problem 3 In this problem, we consider a generalization of DFAs called second-order deterministic finite automata, or DFA(2)s for short. The transition function of a DFA(2) depends not only on the current symbol and the current state, but also the previous state. That is, the transition function is a function $\delta^{(2)}: Q \times Q \times \Sigma \to Q$, that maps every pair of states $(q_{\text{prev}}, q_{\text{curr}})$ and every input symbol $a$ to a next state $q_{\text{next}} = \delta^{(2)}(q_{\text{prev}}, q_{\text{curr}}, a)$. There is a special case for processing the very first input symbol, when there is no previous state; we define the next state to be $\delta^{(2)}(q_0, q_0, a)$, where $q_0$ is the start state and $a$ is the first input symbol.

Clearly any DFA can be transformed into an equivalent DFA(2) in which the transition function does not depend on the previous state, i.e., $\delta^{(2)}(q_{\text{prev}}, q_{\text{curr}}, a) = \delta(q_{\text{curr}}, a)$ regardless of $q_{\text{prev}}$.

Show that every DFA(2) can be transformed into an equivalent DFA.

Hint: Recall the transformation from NFA to DFA.