Problems:

1. Sipser’s book, Exercise 1.3 (same in both editions.) Make sure you include everything that a state diagram should include!

2. The rule for valid names for variables in C programs is the following. Variables must begin with a character (that is, a letter in the English alphabet) or underscore and may be followed by any combination of characters, underscores, or the digits 0 – 9. Design a DFA that accepts strings that are valid variable names (For simplicity assume that \( \Sigma = \{< c >, < d >, < u >, \#\} \) where \(< c >\) denotes a character, \(< d >\) denotes a digit, and \(< u >\) denotes and underscore, and \(\#\) denotes any other possible ASCII character.

3. Give state diagrams of DFAs recognizing the following languages. In each parts the alphabet is \( \Sigma = \{0, 1\} \). As documentation for you DFA, for each state, give a brief informal description of the set of strings which reach this state.

   (a) \( \{ w \mid w \text{ contains at least three 1s.} \} \)
   
   (b) \( \{ w \mid w \text{ has length at least 3 and its third symbol is a 0.} \} \)
   
   (c) \( \{ w \mid w \text{ has an even number of 0s and an odd number of 1s.} \} \)
   
   (d) \( \{ w \mid w \text{ begins with a 1, and which, interpreted as the binary representation of a positive integer, is divisible by 4 } \}

For this problem assume that the DFA starts reading the string from its most significant bit. For example if \( w = 1000 \), then \( w \) is the binary representation of the (decimal) number 8 (and thus, is in the language). and the DFA starts by reading the bit 1.

4. The reversal of a string \( w \) denoted by \( w^R \), is the string when you look at it backwards: for example, \( \text{homer}^R = \text{remoh} \). Here is the formal inductive definition (where the alphabet is \( \Sigma \)):

   \[ \text{Base case. If } w = \epsilon, \text{ then } w^R = \epsilon. \]

   \[ \text{Inductive step. If } w = va \text{ for } v \in \Sigma^* \text{ and } a \in \Sigma, \text{ then } w^R = av^R. \]

   Prove by induction (on the length of \( y \)) that for all strings \( x, y \in \Sigma^* \),

   \[ (xy)^R = y^Rx^R \]

5. **Extra Credit** (minimal points, do it for the glory!) Sipser’s book, Problem 1.37 in second edition (Problem 1.30 in the first edition.)