Homework 3

1. Give NFAs with the specified number of states recognizing each of the following languages. In all cases, the alphabet is $\Sigma = \{0, 1\}$.
   
   (a) The language $\{ w \in \Sigma^* \mid w \text{ ends with } 00 \}$ with three states.
   
   (b) The language $\{ w \in \Sigma^* \mid w \text{ contains the substring } 0101, \text{i.e., } w = x0101y \text{ for some } x, y \in \Sigma^* \}$ with five states.
   
   (c) The language $\{ w \in \Sigma^* \mid w \text{ contains at least two } 0\text{s, or exactly two } 1\text{s} \}$ with six states.
   
   (d) The language $\{\varepsilon\}$ with one state.
   
   (e) The language $0^*1^*0^*0$ with three states.

2. (a) Show by giving an example that, if $M$ is an NFA that recognizes language $C$, swapping the accept and non-accept states in $M$ doesn’t necessarily yield a new NFA that recognizes $\overline{C}$.
   
   (b) Is the class of languages recognized by NFAs closed under complement? Explain your answer.

3. Use the construction given in Theorem 1.39 to convert the following NFA $N$ into an equivalent DFA.

4. Give regular expressions that generate each of the following languages. In all cases, the alphabet is $\Sigma = \{a, b\}$.
   
   (a) The language $\{ w \in \Sigma^* \mid |w| \text{ is odd} \}$.
   
   (b) The language $\{ w \in \Sigma^* \mid w \text{ has an odd number of } a\text{'s} \}$.
(c) The language \{ w \mid w \text{ contains at least two } a \text{'s, or exactly two } b \text{'s} \}.
(d) The language \{ w \in \Sigma^* \mid w \text{ ends in a double letter} \}. (A string contains a \textit{double letter} if it contains aa or bb as a substring.)
(e) The language \{ w \in \Sigma^* \mid w \text{ does not end in a double letter} \}.
(f) The language \{ w \in \Sigma^* \mid w \text{ contains exactly one double letter} \}. For example, \textit{baaba} has exactly one double letter, but \textit{baaaba} has two double letters.

5. Suppose we define a restricted version of the Java programming language in which variable names must satisfy all of the following conditions:

- A variable name can only use Roman letters (i.e., a, b, …, z, A, B, …, Z) or Arabic numerals (i.e., 0, 1, 2, …, 9); i.e., underscore and dollar sign are not allowed.
- A variable name must start with a Roman letter: a, b, …, z, A, B, …, Z
- The length of a variable name must be no greater than 8.
- A variable name cannot be a keyword (e.g., if). The set of keywords is finite.

Let \( L \) be the set of all valid variable names in our restricted version of Java.

(a) Let \( L_0 \) be the set of strings satisfying the first 3 conditions above; i.e., we do not require the last condition. Give a regular expression for \( L_0 \).
(b) Prove that \( L \) has a regular expression, where \( L \) is the set of strings satisfying all four conditions.
(c) Give a DFA for the language \( L_0 \) in part (a), where the alphabet \( \Sigma \) is the set of all printable characters on a computer keyboard (no control characters), except for parentheses to avoid confusion.

6. Define \( L \) to be the set of strings that represent numbers in a modified version of Java. The goal in this problem is to define a regular expression and an NFA for \( L \). To precisely define \( L \), let the set of digits be \( \Sigma_1 = \{ 0, 1, 2, \ldots, 9 \} \), and define the set of signs to be \( \Sigma_2 = \{ +, - \} \). Then \( L = L_1 \cup L_2 \cup L_3 \), where

- \( L_1 \) is the set of all strings that are decimal integer numbers. Specifically, \( L_1 \) consists of strings that start with an optional sign, followed by one or more digits. Examples of strings in \( L_1 \) are “02”, “+9”, and “-241”.
- \( L_2 \) is the set of all strings that are floating-point numbers that are not in exponential notation. Specifically, \( L_2 \) consists of strings that start with an optional sign, followed by zero or more digits, followed by a decimal point, and end with zero or more digits, where there must be at least one digit in the string. Examples of strings in \( L_2 \) are “13.231”, “-28.” and “.124”. All strings in \( L_2 \) have exactly one decimal point.
• $L_3$ is the set of all strings that are floating-point numbers in exponential notation. Specifically, $L_3$ consists of strings that start with a string from $L_1$ or $L_2$, followed by “E” or “e”, and end with a string from $L_1$. Examples of strings in $L_3$ are “-80.1E-083”, “+8.E5” and “1e+31”.

Assume that there is no limit on the number of digits in a string in $L$. Also, we do not allow for the suffixes $L$, $l$, $F$, $f$, $D$, $d$, at the end of numbers to denote types (long integers, floats, and doubles). Define $\Sigma$ as the alphabet of all printable characters on a computer keyboard (no control characters), except for parentheses to avoid confusion.

(a) Give a regular expression for $L_1$. Also, give an NFA and a DFA for $L_1$ over the alphabet $\Sigma$.

(b) Give a regular expression for $L_2$. Also, give an NFA for $L_2$ over the alphabet $\Sigma$.

(c) Give a regular expression for $L_3$. Also, give an NFA for $L_3$ over the alphabet $\Sigma$.

(d) Give a regular expression for the language $L$. Also, give an NFA for $L$ over the alphabet $\Sigma$. 