COMP4141 Tutorial 2

Exercise 1 (Regular Expressions) (If not already done in tute 1) Write a regular expression for each of the following languages over alphabet $\Sigma = \{a, b\}$:

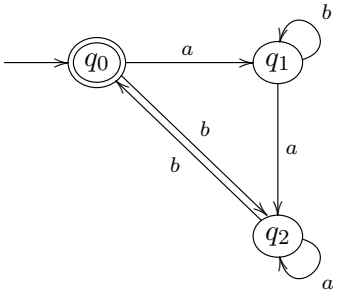
1. the set E of words of even length,
2. the set O of words of odd length,
3. the set of words of even length that contain both an a and a b. (Hint: first argue that a word in this language must contain an *adjacent* a and b.)

Exercise 2 (Regular Expression to NFA) Build a NFA that accepts the language represented by the regular expression

$$((abc)^* \cup (ba \cup (d^*)))^*$$

for alphabet $\Sigma = \{a, b, c, d\}$. Show your working.

Exercise 3 (DFA to Regular Expression) Consider the following DFA in graphical representation:



Using the construction from lectures/Sipser, convert this to a regular expression that accepts the same language. Delete states in the order q_2, q_1, q_0, and show your working.

Exercise 4 (Pumping) Show that the set of strings of `0`'s and `1`'s of the form $w\overline{w}$, where $\overline{w}$ is formed from w by replacing all `0`'s by `1`'s and vice versa, is not a regular language.

Exercise 5 Consider the following context free grammar for a simple programming language: $G = (N, \Sigma, P, S)$, where $N = \{S, I, A, C, E, V\}$, $\Sigma = \{x,\ y,\ z,\ <,\ :=,\ ;,\ \{,\ \},$ **if**, **then**, **else** $\}$[1] and P contains the following productions:

$$\begin{aligned} S &\to I \mid A \mid C \\ I &\to \textbf{if } E \textbf{ then } S \\ I &\to \textbf{if } E \textbf{ then } S \textbf{ else } S \\ A &\to V := V \\ C &\to \{S; S\} \\ E &\to V < V \\ V &\to x \mid y \mid z \end{aligned}$$

1. Show the parse tree for a simple program in $L(G)$ that sorts the variables x, y into ascending order.
2. Show that this grammar is ambiguous, by showing the parse trees for a string in the language that has two distinct parse trees. (Hint: consider nested uses of the I productions.)

[1] Here the inner braces are terminal symbols and the outer braces are part of the set notation. Treat the keywords **if**, **then**, **else** as if each is a single terminal symbol rather than a sequence of letters.