1. (14 pts) Complete or answer the following.

\[ L_1 = \{a, ba\}, \Sigma = \{a, b\} \]
\[ L_2 = \{b, bb, ba\}, \Sigma = \{a, b\} \]
\[ L_3 = \{b^n | n > 0\}, \Sigma = \{b\} \]
\[ L_4 = (aa)^* (bb)^*, \Sigma = \{a, b\} \]
\[ L_5 = \{w \in \Sigma^* | n_a(w) < n_b(w)\}, \Sigma = \{a, b\} \]

(a) \[ L_1 \cap L_2 = \{ ba \} \]

(b) \[ L_4 \cap L_5 = \{ (aa)^n (bb)^m | n \geq 0, m > 0, m > n \} \]

(c) \[ L_2 \setminus L_1 = \{ b, bb \} \]

(d) \[ L_4 \circ L_3 = (aa)^* (bb)^* bb \]

(e) \[ |L_1 \circ L_1 \circ L_1| = 8 \]

(f) \[ L_1 \times L_2 = \{ (a, b), (a, bb), (a, ba), (ba, b), (ba, bb), (ba, ba) \} \]

(g) \[ |2^{L_2}| = 8 \]

2. (22 pts) Answer TRUE or FALSE to each of the statements below.

(a) \( \{a\} \in \{a, \{a, b\}\} \) (TRUE or FALSE?) \( \text{FALSE} \)

(b) The language of valid arithmetic expressions in Java can be recognized by a regular expression. (TRUE or FALSE?) \( \text{FALSE} \)

(c) In the definition of an NFA, the states, the alphabet and the set of transitions for \( \delta \) must all be finite sets. (TRUE or FALSE?) \( \text{TRUE} \)

(d) \( \lambda \) is accepted by a DFA if only if the start state is a final state. (TRUE or FALSE?) \( \text{TRUE} \)
(e) Consider the conversion of an NFA to a DFA. A state in this DFA is a final state if and only if all the states that it represents from the NFA were final states in the NFA. (TRUE or FALSE?) FALSE

(f) If M is a DFA, then there exists a CFG G such that L(M) = L(G). (TRUE or FALSE?) TRUE

(g) If L_1 and L_2 are regular languages, then there exists a regular grammar for L_1 \cap L_2. (TRUE or FALSE?) TRUE

(h) L_1 = \{w \in \Sigma^* | n_a(w) \text{ is even and } n_b(w) \text{ is odd}\} where \Sigma = \{a, b\}, and L_2 = a^*b^*a^*. L_1 \cap L_2 \text{ is a regular language.} (TRUE or FALSE?) TRUE

(i) L = \{a^{2n}b^m | n > 0, m \geq 0\}. L \text{ is regular.} (TRUE or FALSE?) TRUE

(j) L = \{w \in \Sigma^* | n_a(w) \text{ is even and } n_b(w) > 2 \times n_a(w)\} where \Sigma = \{a, b\}. L \text{ is regular.} (TRUE or FALSE?) FALSE

(k) L = \{a^n b^m | n \text{ mod } 5 = 0 \text{ and } m \text{ mod } 3 = 2\}. L \text{ is regular.} (TRUE or FALSE?) TRUE

3. (6 pts) Consider the following parse tree.

```
       S
      / \  \
   a    b
  / \   / \ \
S   B  S  S
 / \ / \ \\
B   B  S  b
 / \ / \ / \\n a  c a c a b
```

a) What is the string this parse tree represents? abcabcab

b) Give the smallest CFG G = (V, T, S, P) that could generate this parse tree. The start symbol S has already been identified.

V = { S, B }
T = { a, b, c }
P = (list the rules below here)

- S \Rightarrow aSb
- S \Rightarrow BS
- S \Rightarrow a
- B \Rightarrow b
- B \Rightarrow c
4. (3 pts) Suppose $M$ is a deterministic PDA (DPDA) with $\Sigma = \{a, b\}$ and $\Gamma = \{Z, 1\}$. Suppose a particular state $q_2$ has four arcs leaving the state. List what the labels on those four arcs might be. (Format of labels are $a, b; cd$ where $a$ is the symbol on the tape, $b$ is the symbol on top of the stack that is popped, and $cd$ are pushed onto the stack (with $c$ on top of $d$).

\[ \begin{array}{c}
q_2 \\
\overrightarrow{a, Z; 1Z} \\
\overleftarrow{a; 1} \\
\overleftarrow{b, Z; 1Z} \\
\overrightarrow{b, Z; 1Z}
\end{array} \]

5. (5 pts) Write a CFG $G$ for the following language:

$L = \{a^n b^m c^{2n} \mid n > 0, m \geq 0\}, \Sigma = \{a, b, c\}$.

\[
\begin{align*}
S & \rightarrow a S c c \\
S & \rightarrow a B c c \\
B & \rightarrow b b B \\
B & \rightarrow \lambda
\end{align*}
\]
6. (8 pts) Draw a DFA for the following language. You do not have to show trap states (show the transition diagram, indicate the start state by a short arrow, and final states by double circles.)

$L = \{ w \in \Sigma^* \mid n_b(w) \mod 3 = 0 \text{ and } w \text{ does not have the substring } aab \}, \Sigma = \{ a, b \}.$

For example, $bbba, bababa, babbabbbaa$ and $aaqa$ are in $L.$
7. (6 pts) Consider the following DFA.

\[ S^*(q_0, baab) = q_5 \quad \text{a non-final state} \]
\[ S^*(q_2, baab) = q_2 \quad \text{a final state} \]

b) Give the states in the minimal state DFA (you do not need to show the arcs). Each state should indicate which states it represents from the original DFA. For example you could list one state as 0,1,2 if states 0,1 and 2 in the original DFA can be combined to form a state in the minimal state DFA.
8. (10 pts) Consider $L = \{a^n b^p c^q \mid n + q > p, p > 0, q > 0, n > 0, \text{ and } n \text{ is even }\}$, $\Sigma = \{a, b, c\}$. Draw the transition diagram for a nondeterministic pushdown automaton $M$ that accepts $L$ by final state. (Remember to identify the start state by an arrow and final states by double circles. Format of labels are $a, b; cd$ where $a$ is the symbol on the tape, $b$ is the symbol on top of the stack that is popped, and $cd$ are pushed onto the stack (with $c$ on top of $d$). $Z$ is on top of the stack when $M$ starts.)

(a) First list 3 strings in $L$.

$aabbcc$, $aabbccc$, $aabbcc$

(b) Now draw the transition diagram.

[Diagram of a nondeterministic pushdown automaton with transitions labeled according to the problem statement.]
9. (6 pts) **Pumping Lemma:** Let $L$ be an infinite regular language. \( \exists \) a constant \( m > 0 \) such that any \( w \in L \) with \( |w| \geq m \) can be decomposed into three parts as \( w = xyz \) with

\[
\begin{align*}
|xy| &\leq m \\
|y| &\geq 1 \\
xy^iz &\in L \quad \text{for all } i \geq 0
\end{align*}
\]

Use the Pumping Lemma to prove the language $L$ below is not regular.

$L = \{a^n b^p c^q \mid q > n > 0, p > 0\}, \Sigma = \{a, b, c\}$.

**Proof:** (SHOW ALL STEPS! Some have been started for you.)

Assume \( \underline{L \text{ is regular}} \)

Choose \( w = a^m b b c^{m+1} \)

Show there is no way to partition this string \( w=xyz \) such that the properties of the pumping lemma hold.

\[
\begin{align*}
x &= a^j \\
y &= a^k \\
z &= a^{m-j-k} b b c^{m+1}
\end{align*}
\]

\( \forall i \ x y^i z \in L \)

\( i = 2 \)

\( x y y z = a^{m+k} b b c^{m+1} \notin L \)

Since \( n_a(w) \geq n_c(w) \)

Contradiction!

Thus \( L \) is not regular.
10. (8 pts) Consider the following property, RemoveFirstAndLastA. If \( L \) is a regular language, then \( \text{RemoveFirstAndLastA}(L) \) consists of strings from \( L \) that have at least two \( a \)'s, and the first \( a \) and the last \( a \) is removed. If there is a string \( w \) in \( L \) that does not have two \( a \)'s in the string, then that does not put a string in \( \text{RemoveFirstAndLastA}(L) \).

For example, if \( ababa \) is in \( L \), then \( bab \) is in \( \text{RemoveFirstAndLastA}(L) \). If \( babaaab \) is in \( L \), then \( bbaab \) is in \( \text{RemoveFirstAndLastA}(L) \).

If \( ab \) is in \( L \), then \( ab \) does not generate a string in \( \text{RemoveFirstAndLastA}(L) \).

Start the proof to show that \( \text{RemoveFirstAndLastA}(L) \) is a regular language. You need to explain how to convert a DFA \( M \) for \( L \) into a DFA or NFA that represents the language \( \text{RemoveFirstAndLastA}(L) \).

\[ L \text{ is regular } \implies \exists \text{ DFA } M \text{ for } L. \]

Make 3 copies of \( M \) called \( M, M', M'' \)

\[ M = (Q, \Sigma, S, q_0, F), \quad M' = (Q', \Sigma, S', q'_0, F'), \quad M'' = (Q'', \Sigma, S'', q''_0, F'') \]

\[ \hat{M} = (\hat{Q}, \Sigma, \hat{S}, q_0, F) \] is the NFA we will construct.

\[ \hat{Q} = Q \cup Q' \cup Q'' \]

\[ \hat{S} = S - \{ \text{all } 'a' \text{ arcs} \} \cup S' \cup S'' - \{ \text{all } 'a' \text{ arcs} \} \]

\[ \{ S'(p, a) = q' \} \quad \text{where } S(p, a) = q \]

\[ \{ S''(p', a) = q'' \} \quad \text{where } S'(p', a) = q' \]

\( \hat{M} \) represents an NFA for \( \text{RemoveFirstAndLastA}(L) \).

Thus it is a regular language.
More detail on the construction

\[ M \text{ is 3 copies of } M \text{ called } M, M', M''. \]

The changes are:

1) Start with start state in M.
2) Remove all \( \lambda \) arcs in M.
3) Add a \( \lambda \) arc for all \( \lambda \) arcs in M going to the corresponding state in M'.
4) For all \( a \) arcs in M', add a \( \lambda \) arc to the corresponding state in M''.
5) Remove all \( \lambda \) arcs in M''.
6) All final states in M are now non-final.

To process a string, the first \( a \) gets replaced with \( \lambda \) and you move to M'. In M' at some point, for the last \( a \), you move to M'' and don't process the \( a \). You can only process bab at this point and then accept.

So for string \( w = uavax \), where \( u, x \in \{ \lambda, a \} \), \( v \in \Sigma^* \), \( w' = uvx \) is accepted in \( M' \).