1. Honor code (2 pts)

Type your name to acknowledge the Duke Community Standard

2. (12 pts) Answer TRUE or FALSE to each of the statements below.

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

(b) If G is a CFG, then there exists a regular expression R such that L(G) = L(R). (TRUE or FALSE?)
   **ANSWER: FALSE**

(c) \( L = \{a^mb^n a^n \mid m > 0, p > 0, n > 0\} ; \Sigma = \{a,b\} \). L is a regular language. (TRUE or FALSE?)
   **ANSWER: TRUE**

(d) \( L = \{a^mb^n \mid n > m, m > 10\} ; \Sigma = \{a,b\} \). L is a regular language. (TRUE or FALSE?)
   **ANSWER: FALSE**

(e) \( L = \{w \in \Sigma^* \mid n_a(w) \text{ is even and } n_b(w) = n_a(w) + 1\} ; \Sigma = \{a,b\} \). L is a regular language. (TRUE or FALSE?)
   **ANSWER: FALSE**

(f) \( L = \{a^mb^p c^{2m} \mid p > 100, m < 100\} ; \Sigma = \{a,b,c\} \). L is a regular language. (TRUE or FALSE?)
   **ANSWER: TRUE**
3. (6 pts) Consider the following grammar G with start variable S.

\[
\begin{align*}
S & \rightarrow aSBCd \mid c \\
B & \rightarrow bB \mid b \\
C & \rightarrow d
\end{align*}
\]

A) Give a RIGHTMOST derivation for the string acbbdd.

\textbf{ANSWER:}

\[
\begin{align*}
S & \rightarrow aSBCd \\
S & \rightarrow aSBdd \\
S & \rightarrow aSbbdd \\
S & \rightarrow acbbdd
\end{align*}
\]

B) Draw the parse tree for the same string acbbdd.

\textbf{ANSWER:}
4. (6 pts) Write a CFG G for the following language:

\[ L = \{ b^m a^n c^n d^p \mid n > 0, m > 0, p > 0 \}, \Sigma = \{ a, b, c, d \}. \]

For example, \( bbbaaccd \in L \).

**ANSWER:**

\[
\begin{align*}
S & \rightarrow bSd \mid bBd \\
B & \rightarrow bBc \mid bAc \\
A & \rightarrow aA \mid a
\end{align*}
\]
5. (10 pts) Consider \( L = \{a^n b^m c^n d^p \mid n > 0, m > 0, p > 0\} \), \( \Sigma = \{a, b, c, d\} \). 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 \( L \).

**ANSWER:**

\( \text{abbcd, abbbccd, aabbbccd} \)

(b) Now draw the transition diagram.

**ANSWER:**

![Transition Diagram](attachment:image.png)
6. (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

$$
|xy| \leq m \\
|y| \geq 1 \\
xy^iz \in L \text{ for all } i \geq 0
$$

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

$L = \{ w \in \Sigma^* | n_a(w) + n_b(w) < 2 \times n_c(w) \}, \Sigma = \{ a, b, c \}$

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

Assume $L$ is regular.

Choose $w = a^mb^mc^{m+1}$ where $m$ is the constant in the pumping lemma.

The only way to partition $w$ into three parts, $w = xyz$, is such that $x$ contains 0 or more $a$’s, $y$ contains 1 or more $a$’s, and $z$ contains 0 or more $a$’s concatenated with the rest of the string $b^mc^{m+1}$. This is because of the restrictions $|xy| \leq m$ and $|y| > 0$. So the partition is:

$x = a^k$ \quad $y = a^j$ \quad $z = a^{m-k-j}b^mc^{m+1}$

where $k \geq 0, j > 0$, and $k + j \leq m$ for some constants $k$ and $j$.

It should be true that $xy^iz \in L$ for all $i \geq 0$.

Choose $i=3$. Note that $i=2$ does not give a contradiction.

$xy^3z = a^{m+2j}b^mc^{m+1} \notin L$, since $2m + 2j > 2 \times (m + 1)$

$\Rightarrow L$ is not regular!. QED.

NOTE: other strings will work. For example, choose $w = a^mb^{m+1}c^{m+1}$ where $m$ is the constant in the pumping lemma. Then $i=2$ gives a contradiction.
7. (9 pts) Consider the following grammar G with start variable S.

\[
S \rightarrow aSBd \mid cS \mid a \\
B \rightarrow bBb \mid b
\]

In the additions below you may add a new variable or two if you want.

A) Give one or two productions that could be added to the grammar above to form the new grammar G’ such that L(G) = L(G’) and L(G) has a unit production.

**ANSWER:**

\[
S \rightarrow A \\
A \rightarrow a
\]

B) Give one or two productions that could be added to the original grammar above (without the changes from Part A) to form the new grammar G’ such that L(G) = L(G’) and L(G) has a \( \lambda \) production.

**ANSWER:**

\[
S \rightarrow Aa \\
A \rightarrow \lambda
\]

C) Give one or two productions that could be added to the original grammar above (without the changes from Part A or Part B) to form the new grammar G’ such that L(G) = L(G’) and L(G) has a useless production.

**ANSWER:**

\[
C \rightarrow cC \\
C \rightarrow \lambda
\]

8. (3 pts) The following grammar is LL(k) for what value of k? Give the value of k and an example of two strings that need that value of k to distinguish which rule to apply.

\[
S \rightarrow BCAb \mid babcBa \\
A \rightarrow aaA \mid \lambda \\
B \rightarrow baB \mid b \\
C \rightarrow cC \mid c
\]

**ANSWER:**

For \( k = 6 \).

\[
S \rightarrow BCAb \rightarrow baBCAb \rightarrow babCAb \rightarrow babcAb \rightarrow babcb \\
S \rightarrow babcBa \rightarrow babcbaBa \rightarrow babcbaba
\]

Note babcb is in common with the two strings, the next character in the first one is $ and the next character in the second one is a, so six lookaheads are needed to determine which S rule to use.
9. (12 pts) Consider the following grammar and information on creating the LR parse table, such as the corresponding First and Follow sets, DFA that models the LR(1) process, and LR(1) table where a conflict is shown. Note that the LR Parsing process has added the first rule in the grammar just for the LR parsing process.
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>c</th>
<th>d</th>
<th>$</th>
<th>A</th>
<th>B</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>r3</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>s5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>acc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>r3</td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>s7</td>
<td>s8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>r5</td>
<td>r5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>r2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>r1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>r4</td>
<td>r4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
a) Suppose you are parsing a string with the table and you are in row 4 of the table and d is the lookahead. Explain what happens for this step.

**ANSWER:**

s8 is the entry in Table[4,d]. That means to shift the d onto the stack followed by the 8 onto the stack and then move to row 8 in the table (which is also state 8 in the dfa).

b) Suppose you are parsing a string with the table and you are in row 8 of the table and d is the lookahead. Explain what happens, including what happens to the stack and which row of the table you are in next.

**ANSWER:**

Table[8, d] = r4. We reduce by the rule number 4 in the grammar which is $B \to Bd$, meaning that we pop Bd off the stack (and the numbers associated with them) and replace it with B on the stack. Now we return to the state or row we were in before state 8, which, from the DFA, we see is state 4 from the b, and then state 1 from the B. Then we can look in the Table[1, B] is 4 and we push 4 also on the stack, and move back to state 4 in the DFA.

c) In the table there are two conflicts. One is in row 0, column a. Explain what the conflict is (not all of it is shown in the table) and why there is a conflict.

**ANSWER:**

There is a shift-reduce conflict when the lookahead is a. Note state 0 is a final state (the lambda production), meaning you can reduce by the rule $A \to \lambda$ for every terminal in the the Follow of A, and 'a' is in the follow of A. But there is an 'a-arc' coming out of state 0, we can also shift 'a' onto stack and go to state 3 by: $A \to a.A$, so there is a shift-reduce conflict.

d) In the table the second conflict is in row 3, column a. Explain what the conflict is (not all of it is shown in the table) and why there is a conflict.

**ANSWER:** There is a shift-reduce conflict when the lookahead is a. Note state 3 is a final state (the lambda production), meaning you can reduce by the rule $A \to \lambda$ for every terminal in the the Follow of A, and 'a' is in the follow of A. But there is an 'a-arc' coming out of state 3, we can also shift 'a' onto stack and go to state 3 by: $A \to a.A$, so there is a shift-reduce conflict.
10. (4 pts) Consider the state below from a DFA that models the LR parsing process for some grammar not shown.

Draw and label all the arcs that would come out of this state, and draw the states those arcs would go to. Put the appropriate marked rules in each of those new states, apply the closure to them, and identify final states with two circles. You do not need to show any arcs out of these new states.

For the state given, you can draw it as a circle and you do not need to rewrite all the rules in it.

You are not given the full grammar so just apply the productions you can see that are in the given state.

---

**ANSWER:**

---