1. (10 pts) Consider the following languages. Write “REG” if it is regular, “CFL” if it is a CFL and not regular, and write “NOT” if it is not a CFL.

   (a) \( L = \{a^n b^m c^n d^m \mid n > 0, m > 0 \text{ and } m \text{ is even} \} \)

   (b) \( L = \{a^n b^m c^n d^m \mid n > 0, m > 0, n > 0, q > 0 \text{ and } p > n + m + q \} \)

   (c) \( L = \{a^n b^m c^n d^m \mid 0 < n < 100, m > 100 \} \)

   (d) \( L = \{ww^Rw \mid w \in \Sigma^* \}, \Sigma = \{a, b\} \)

   (e) \( L = \{a^n b^m c^p \mid n + p \text{ is even, } n > 0, m > 5, p > 0 \} \)

2. (10 pts) Answer TRUE or FALSE to each of the statements below.

   (a) If \( M \) is a Turing machine, then there exists an NPDA \( M' \) such that \( L(M) = L(M') \). (TRUE or FALSE?)

   (b) If \( G \) is a CFG, then there exists a TM \( M \) such that \( L(G) = L(M) \) and \( M \) halts on all inputs. (TRUE or FALSE?)

   (c) If \( M \) is an NPDA, then there exists a DPDA \( M' \) such that \( L(M) = L(M') \). (TRUE or FALSE?)

   (d) If \( M \) is an NPDA, then there exists an NPDA \( M' \) with just one final state such that \( L(M) = L(M') \). (TRUE or FALSE?)

   (e) LR parsing is a top-down parsing method. (TRUE or FALSE?)

3. (3 pts) Give two CFL’s \( L_1 \) and \( L_2 \) such that \( L_1 \cap L_2 \) is also a CFL.

4. (3 pts) Give two CFL’s \( L_1 \) and \( L_2 \) such that \( L_1 \cap L_2 \) is NOT a CFL.

5. (3 pts) Suppose a Turing machine has two tapes instead of one, and two tape heads, one for each tape as shown in the figure below.
Then the Turing machine is still defined as a 7-tuple \( M = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \) with only \( \delta \) changed. A move consists of reading both the symbols the two tape heads are pointing to, writing symbols in these two tape squares and moving each tape head one square either left or right (they could move in different directions).

Formally give the definition of \( \delta \) for this Turing machine.

6. (3 pts) The following grammar is LL(k) for what value of \( k \)?

\[
\begin{align*}
S & \rightarrow ABdab \mid aCb \\
A & \rightarrow aA \mid a \\
B & \rightarrow b \mid ba \\
C & \rightarrow abd
\end{align*}
\]

7. (10 pts) Consider \( L = \{a^n b^m c^{2n} \mid n > 0, m > 0, \text{and } m \text{ is even} \} \). 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 \( L \).

(b) Now draw the transition diagram.

8. (8 pts) Pumping Lemma for CFL’s Let \( L \) be any infinite CFL. Then there is a constant \( m \) depending only on \( L \), such that for every string \( w \) in \( L \), with \( |w| \geq m \), we may partition \( w = uvxyz \) such that:

\[
\begin{align*}
|vxy| & \leq m, \text{ (limit on size of substring)} \\
|vy| & \geq 1, \text{ (} v \text{ and } y \text{ not both empty)} \\
\text{For all } i \geq 0, \ uv^i xy^i z & \in L
\end{align*}
\]

Prove that \( L = \{w \in \Sigma^* \mid n_a(w) = 2 \ast n_b(w), n_b(w) < n_c(w)\} \), is not a context-free language.

You only have to fill in the parts below. Assume \( L \) is a context-free language.

(a) Choose \( w = \)

(b) Prove the case when \( v = a^{t_1} \) and \( y = a^{t_2} \) (both are strings of a’s)

(c) Prove the case when \( v = a^{t_1} \) and \( y = b^{t_3} \)

(d) Prove the case when \( v = b^{t_1} \) and \( y = c^{t_2} \)

9. (10 pts) Consider the following grammar (DO NOT change the grammar):
S \rightarrow ABCd  
A \rightarrow aA \mid Ba \mid \lambda 
B \rightarrow bd \mid \lambda 
C \rightarrow cCc \mid d  

For this problem you will construct the LL(1) parse table.

(a) Calculate FIRST and FOLLOW for the variables in the grammar.

<table>
<thead>
<tr>
<th></th>
<th>FIRST</th>
<th>FOLLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Calculate all entries in the LL(1) Parse Table. If there are multiple rules for an entry, write in all the rules.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10. (16 pts) Construct the LR parsing table for the following grammar (DO NOT change the grammar.) A new start symbol S' and production have already been added to the grammar.

1) S' \rightarrow S  
2) S \rightarrow ABd  
3) A \rightarrow aA  
4) A \rightarrow Ba  
5) B \rightarrow \lambda  

(a) Calculate the FIRST and FOLLOW sets of variables.
(b) Construct the transition diagram of the DFA that models the stack. Number the states, show marked productions, and identify final states by two circles.

(c) Construct the LR parse table that corresponds to the transition diagram drawn in part b. (Note: all the rows and columns given may not be needed. If there are multiple items for an entry, write all in the entry.)
S → ABd | aC | Aa
A → aAb | a
B → bEb
C → cC | cB
D → dD | d
E → Ba

Algorithm: To Remove Useless Productions:
Let G=(V,T,S,P).
I. Compute \( V_1 = \{ \text{Variables that can derive strings of terminals} \} \)
   (a) \( V_1 = \emptyset \)
   (b) Repeat until no more variables added
       • For every \( A \in V \) with \( A \rightarrow x_1 \ldots x_n, x_i \in (T^* \cup V_1) \), add \( A \) to \( V_1 \)
   (c) \( P_1 = \) all productions in \( P \) with symbols in \( (V_1 \cup T)^* \)

Then \( G_1 = (V_1,T,S,P_1) \) has no variables that can't derive strings.

II. Draw Variable Dependency Graph
For \( A \rightarrow xBy \), draw \( A \rightarrow B \).
Remove productions for \( V \) if there is no path from \( S \) to \( V \) in the dependency graph.
Resulting Grammar \( G' \) is s.t. \( L(G) = L(G') \) and \( G' \) has no useless productions.

Remove Useless Productions from the grammar above.

(a) Compute \( V_1 = \{ \)
(b) Draw the dependency graph
(c) Write the resulting grammar that has no useless productions.

12. (8 pts) Construct a TM (using a transition diagram) for the following function. \( f(w) = u0v \), where \( \Sigma = \{1\} \), \( w, u, v \in \Sigma^* \), \( w = uv \) and \( |u| = |v|, |w| > 0 \). In other words, \( w \) is a string of 1’s of even length, and the output is the same string of 1’s with a 0 inserted in the middle. You can assume the input is correct (you will not be given a string of odd length).

For example, \( f(1111) = 11011 \), and \( f(111111) = 1110111 \).

In drawing the transition diagram, remember to identify the start state by an arrow and final states by double circles. Format of labels are \( a; b, R \) where \( a \) is the symbol read on the tape, \( b \) is the symbol written to the tape and \( R \) is the direction moved (you can use \( L \) and \( R \) for directions.) Make sure the tape head is pointing to the leftmost symbol of the output.