1. (12 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) \( L = \{a^n b^p c^q d^r \mid n < p, q < r, n > 0, q > 0\}, \Sigma = \{a, b, c, d\}. \) **CFL**

(b) \( L = \{w \in \Sigma^* \mid n_a(w) < n_b(w) \text{ and } n_c(w) < n_d(w)\}, \Sigma = \{a, b, c, d\}. \) **NOT**

(c) \( L = \{w \in \Sigma^* \mid n_a(w) \text{ is even, } 0 < n_a(w) < 500 \text{ and } n_b(w) = 2 \times n_d(w)\}, \Sigma = \{a, b\}. \) **REG**

(d) \( L = \{a^n b^p c^q d^r \mid n > 0, p > 0, q > 0, r > 0\}, \Sigma = \{a, b, c, d\}. \) **REG**

(e) \( L = \{a^n b^{2n} c^p \mid n > 0, p > 0\}, \Sigma = \{a, b, c\}. \) **CFL**

(f) \( L = \{a^n b^{2n} c^p \mid p > 2n, n > 0\}, \Sigma = \{a, b, c\}. \) **NOT**

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

(a) If \( G \) is a grammar in Greibach Normal Form, then \( G \) is a context-free grammar. **(TRUE or FALSE?)**

(b) Suppose \( G \) is a context-free grammar with start variable \( S \) and \( S \) has only the three rules: \( S \to ABCS\text{CBA}, S \to a, \) and \( S \to bA, \) and the other variable's rules are not shown. With appropriate rules from the other variables in the grammar, it could be possible that \( S \) could derive \( \lambda. \) **(TRUE or FALSE?)**

(c) If \( M \) is a Turing machine that halts on all inputs, then there exists a CFG \( G \) such that \( L(M) = L(G). \) **(TRUE or FALSE?)**

(d) If \( G \) is a CFG and \( M \) is an NPDA, then there exists an NPDA \( M' \) such that \( L(M') = L(G) \cap L(M). \) **(TRUE or FALSE?)**

(e) In LR parsing, where \( S \) denotes the start variable of the grammar, there can only be one entry in the \( S \) column of the LR(1) parse table if the grammar is LR(1). **(TRUE or FALSE?)**

(f) Every CFG is LL(k) for some \( k. \) **(TRUE or FALSE?)**
3. (4 pts) List two differences between a Turing machine and a DFA.

A T.M. can write on the tape and move left or right, a DFA cannot write or move left.

4. (3 pts) Consider the following derivation of a string from some CFG with \( \Sigma = \{a, b, c, d\} \).

\[ S \to AbCc \to aBBcC \to abbCc \to abbc \]

From this derivation, list the elements that you can determine are in FOLLOW sets for the following variables.

(a) \( \text{FOLLOW}(A) = \{b, d\} \)
(b) \( \text{FOLLOW}(B) = \{b, c, \} \)
(c) \( \text{FOLLOW}(C) = \{\} \)

5. (6 pts) Consider the following CFG.

\[
S \to A \mid B \mid AcB \mid C \\
A \to aa \mid \lambda \\
B \to A \mid b \\
C \to cC \mid AC \\
D \to Dd \mid d
\]

(a) List all the variables that can derive \( \lambda \)

\( A, S, B \)

(b) List all the unit productions

\( S \to A, S \to B, S \to C, B \to A \)

(c) List the useless productions

\( D \to DD \mid d, C \to cC \mid AC, S \to C \)

6. (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.

\[
S \to ACD \mid CBCA \\
A \to abA \mid \lambda \\
B \to bB \mid b \\
C \to a \mid b \mid c \\
D \to dD \mid d
\]
7. (10 pts) Consider the following grammar (DO NOT change the grammar):

\[
S \rightarrow aSB \mid CBd \\
B \rightarrow bB \mid c \\
C \rightarrow Cd \mid cc \mid \lambda
\]

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>$a, b, c, d$</td>
<td>$1, b, c</td>
</tr>
<tr>
<td>C</td>
<td>$c, d, \lambda$</td>
<td>$b, c, d$</td>
</tr>
<tr>
<td>B</td>
<td>$b, c$</td>
<td>$1, b, c, d</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>aSB</td>
<td>CBD</td>
<td>CBD</td>
<td>CBD</td>
<td>$</td>
</tr>
<tr>
<td>C</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>bB</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Is this grammar an LL(1) grammar? Explain. **NO**  
The table has entries with conflicts.
8. (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.

0) S' → S
3) A → a

1) S → BAcB
4) B → b
5) B → λ

(a) Calculate the FIRST and FOLLOW sets of variables.

<table>
<thead>
<tr>
<th></th>
<th>FIRST</th>
<th>FOLLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>b, a</td>
<td>$$</td>
</tr>
<tr>
<td>A</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>B</td>
<td>b, λ</td>
<td>a, $$</td>
</tr>
</tbody>
</table>

(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.**)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>#</th>
<th>S</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>r5</td>
<td>s3</td>
<td>r5</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>acc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>s4</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>r4</td>
<td>r4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>s4</td>
<td>r3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>s7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>r2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>r5</td>
<td>s3</td>
<td>r5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>r1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
9. (8 pts) Consider the following L-system.

Axiom: \( X g f g \)
\( X \rightarrow g \left( + g \ Y \right) X - g \)
\( Y \rightarrow + g \)

angle 45
color black
lineWidth 2
distance 10

Recall that \( g \) is for drawing a line, \( f \) is for moving forward, \( + \) means change the direction by the angle clockwise, \( - \) means change the direction by the angle counterclockwise and \( [ ] \) are used for stacking operations.

Assume a \( g \) drawn with distance 10 and lineWidth 2 is about this size |

a. Render the L-system and draw the axiom if there is a visual picture for it.

   \[ \]
   \[ \]

b. Give the first string in the language (after the axiom) and draw it.

\( g \left[ + g \ Y \right] X - g g f g \)

b. Give the second string in the language (after the axiom) and draw it.

\( g \left[ + g + g \right] g \left[ + g \ Y \right] X - g - g g f g \)
10. (6 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:

- $|vxy| \leq m$, (limit on size of substring)
- $|vy| \geq 1$, (v and y not both empty)
- For all $i \geq 0$, $uv^i xy^i z \in L$

Consider $L = \{c^n a^n b^n \mid 0 < n, 2 \ast p, p > 0\}$ $\Sigma = \{a, b, c\}$.

Prove $L$ is not a context-free language.

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

(a) Choose $w = c^m a^{m-1} b^m$

(b) Prove the case when $v = c^i$ and $y = c^j$

\[ i = 2 \quad u^i v^i x^i y^i z^i = c^{m+i+j} a^i b^i \not\in L \]

because $n_c(w) \neq n_b(w)$

(c) Prove the case when $v = a^i$ and $y = b^j$

Either $i > 0$ or $i = 0$ and $j > 0$

\[ i = 2 \quad u^i v^i x^i y^i z^i = c^{m+i+j} b^i \not\in L \]

because $n_a(w) \neq n_c(w)$

\[ i = 2 \quad u^i v^i x^i y^i z^i = c^{m+i+j} b^{i+j} \]

\[ n_a(w) \neq n_b(w) \]
11. (10 pts) Construct a one-tape Turing machine transducer (using a transition diagram) that computes the following function:

\[ w \in \{a, b\}^*, |w| \geq 1, f(w) = m \] where \( m \) is a unary number for the number of groups of a's in the string that have an even number of a's. A group of a's must either start or end a string, or be adjacent to a b.

For example, \( f(abaabb) = 1 \) since there are two groups of a's (a and aa) and only one of them has an even number of a's. \( f(aaabaababaaaabbaaa) = 111 \) since there are 5 groups of a's (aaa, aa, a, aaaa, and aa) and three of them have an even number of a's. \( f(bbab) = B \) (where B is the blank symbol), since there are no groups of a's with even length. In this case there is no output so the tape head would just be pointing to a blank symbol on the tape.

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.)

\[ |w| = n. \text{ What is the worst case running time (big-Oh) of your TM? } O(n^2) \]
12. (10 pts)

PART 1: Construct a TM (using building blocks) for adding 1 to a binary number. 
\( w \in \{0,1\}^*, |w| \geq 1, f(w) = w + 1 \)

For example, \( f(100) = 101 \) and \( f(1011) = 1100 \).

See the building block notation on later pages. Make sure the tape head is pointing to the leftmost symbol of the output.

Assume \( |w| = n \). What is the running time in terms of \( n \) (big-Oh) of your TM? \( O(n) \)
PART 2: Construct a TM (using building blocks) for adding a unary number to a binary number. \( w \in \{0,1\}^* \) and \( v \in \{1\}^* \), \( |w| \geq 1 \) and \( |v| \geq 1 \), \( f(v\#w) = w + v \)

For example, \( f(11\#100) = 110 \) (add the unary number 2 to the binary number 4, resulting in the binary number 6), \( f(1111\#1100) = 10000 \) (add the unary number 4 (1111) to the binary number 12 (1100), resulting in the binary number 16 (10000).

Hint: Call the machine you made in part 1 "ADD" and use that as a building block. You can assume it works correctly.

See the building block notation on later pages. Make sure the tape head is pointing to the leftmost symbol of the output.

Assume \( |w| = n \). What is the running time in terms of \( n \) (big-Oh) of your TM? \( O(n^2) \)
Notation for Simplifying Turing Machines

Suppose $\Gamma = \{a, b, c, B\}$

$z$ is any symbol in $\Gamma$

$x$ is a specific symbol from $\Gamma$

1. $s$ - start
2. $R$ - move right
3. $L$ - move left
4. $x$ - write $x$ (and don’t move)
5. $R_a$ - move right until you see an $a$ (note that this moves right at least one square before it checks for $a$).
6. $L_a$ - move left until you see an $a$
7. $R_{a|b}$ - move right until you see an $a$ or $b$
8. $L_{a|b}$ - move left until you see an $a$ or $b$
9. $R_{\neg a}$ - move right until you see anything that is not an $a$
10. $L_{\neg a}$ - move left until you see anything that is not an $a$
11. $h$ - halt in a final state
12. $\frac{a|b}{w}$

If the current symbol is $a$ or $b$, let $w$ represent the current symbol.

13. $C$ - copy a string, $F(w) = w0w$, makes a copy of the string and inserts a 0 between the original and the copy.

14. $S_L$ - Shift the string that is to the right of the tape head (up to a blank), to the left, writing over the symbol the tape head is pointing to.

15. $S_R$ - Shift the string that is to the left of the tape head (up to a blank), to the right, writing over the symbol the tape head is pointing to.