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) \( L = \{ a^n b^m c^p \mid n > 0, m > 0, n > m, p = n - m \} \), \( \Sigma = \{ a, b, c \} \). \text{CFL}

(b) \( L = \{ w \in \Sigma^* \mid n_a(w) \text{ is even}, n_b(w) \text{ is divisible by } 10 \} \), \( \Sigma = \{ a, b \} \). \text{REG}

(c) \( L = \{ a^n b^{n+1} c^{n+2} \mid n > 0 \} \), \( \Sigma = \{ a, b, c \} \). \text{NOT}

(d) \( L = \{ a^n b^{2n} c^{2m} d^n \mid n > 0, m > 0 \} \), \( \Sigma = \{ a, b, c, d \} \). \text{CFL}

(e) \( L = \{ a^n b^m c^m \mid n > 0, m > n_b(w) \} \), \( \Sigma = \{ a, b, c \} \). \text{REG}

(f) \( L = \{ w \in \Sigma^* \mid n_a(w) < n_b(w), n_b(w) \text{ is odd} \} \), \( \Sigma = \{ a, b \} \). \text{CFL}

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

(a) If a CFG has some left-recursive rules, then the grammar can be transformed into an equivalent CFG that has no left-recursive rules. (TRUE or FALSE?)

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

(c) If a CFG has no left-recursion, then the LL(1) parse table for that grammar should not have a conflict in its table. (TRUE or FALSE?) Consider \( S \rightarrow aS, S \rightarrow ab, S \rightarrow a \)

(d) If a CFG is LR(1) then it should also be LL(1). (TRUE or FALSE?)

(e) If \( M_1 \) is a DFA and \( M_2 \) is an NPDA, then there exists a TM \( M_3 \) such that \( L(M_3) = L(M_1) \cap L(M_2) \). (TRUE or FALSE?)

(f) If \( G_1 \) is a CFG with only one variable, and \( G_2 \) is a CFG with only one variable, then there exists a CFG \( G_3 \) such that \( L(G_3) = L(G_1) \cap L(G_2) \). (TRUE or FALSE?)
3. (4 pts) Give two CFGs $G_1$ and $G_2$ such that $G_1 \cap G_2$ is a CFG.

$G_1: S \Rightarrow AB$
$A \Rightarrow aA \mid \lambda$
$B \Rightarrow bB \mid \lambda$

$G_2: S \Rightarrow aSb$
$S \Rightarrow cC$
$S \Rightarrow \lambda$

$G_3 = G_1 \cap G_2$

$G_3: S \Rightarrow aSb$

4. (6 pts) Consider the following CFG.

$S \rightarrow aSabb \mid B$
$B \rightarrow bbB \mid \lambda$

(2) (a) Show a derivation of the string $aabbabab$

$S \Rightarrow aSabb \Rightarrow aaSabb \Rightarrow aaBabab \Rightarrow aabbBabab \Rightarrow aabbabab$

(2) (b) Show a top-down parse tree for the string $aabbabab$

(2) (c) Show a bottom-up parse tree for the string $aabbabab$
5. (4 pts) Describe the differences between how a Turing machine and a DFA both process their tape.

DFA - tape head reads only and moves left to right once.

TM - tape head reads and writes to a cell on the tape. The tape head can move left or right.

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.

\[
S \rightarrow AbC | aACa \\
A \rightarrow aA | aAa | \lambda \\
C \rightarrow daC | b
\]

not LL(k) for any k

A → any number of a's

AbC & aACa both can generate a large number of a's.
7. (10 pts) Consider the following grammar (DO NOT change the grammar):

\[
\begin{align*}
S & \rightarrow BcSA \mid b \\
A & \rightarrow aA \mid b \\
B & \rightarrow aBA \mid c
\end{align*}
\]

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

**(4)(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</td>
<td>a, b, $</td>
</tr>
<tr>
<td>A</td>
<td>a, b</td>
<td>a, b, c, $</td>
</tr>
<tr>
<td>B</td>
<td>a, c</td>
<td>a, b, c</td>
</tr>
</tbody>
</table>

**(5)(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>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>BeSA</td>
<td>b</td>
<td>BeSA</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>aA</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>aBA</td>
<td></td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

**(1)(c)** Is this grammar an LL(1) grammar? Explain.

**Yes**

There are no conflicts in the table.
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) $S' \rightarrow S$
1) $S \rightarrow cSa$
2) $S \rightarrow B$
3) $B \rightarrow aB$
4) $B \rightarrow c$

(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>$a_1c$</td>
<td>$a_1$$</td>
</tr>
<tr>
<td>$B$</td>
<td>$a_1c$</td>
<td>$a_1$$</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>c</th>
<th>$</th>
<th>B</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>s3 s4</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>r2</td>
<td>r2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>acc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>s3 s6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>s3 s4 r4</td>
<td>1</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>r3</td>
<td>r3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>r4</td>
<td>r4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>s8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>r1</td>
<td>r1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</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>
</tr>
<tr>
<td>11</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>
</tr>
</tbody>
</table>
9. (8 pts) Consider the following L-system.

Axiom: X Y g
X → g + X
Y → Y [ - g ] g

angle 90
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.

\[ g + X Y [ - g ] g g \]

```
| |
```

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

\[ g + g + X Y [ - g ] g [ - g ] g g \]

```
| |
```
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:

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

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

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 $w = a^{m}c^{m}b^{m+1}$

(b) Prove the case when $v = a^{t_{1}}$ and $y = a^{t_{2}}$

\[ i = 2 \quad uv^{2}xy^{2}z = a^{m+t_{1}+t_{2}}c^{m}b^{m+1} \notin L \text{ since } \]
\[ n_{a}(w) \geq n_{b}(w) \]

(c) Prove the case when $v = c^{t_{1}}$ and $y = b^{t_{2}}$

\[ i = 0 \quad uv^{0}x^{0}y^{0}z = a^{m}c^{m-t_{1}}b^{m+1-t_{2}} \notin L \text{ since } \]
\[ n_{a}(w) \geq n_{b}(w) \]
11. (10 pts) Construct a one-tape Turing machine transducer (using a transition diagram) that computes the following function:

\[ w \in \{a, b\}^*, \quad |w| \geq 1, \quad f(w) = w' \text{ where } w' \text{ is only the } b\text{'s from } w. \]

For example, \( f(abaabb) = bbb \) since there are three \( b\)'s in the string. \( f(aaabaabbabaaabbaa) = bbbbbbb \) since there are six \( b\)'s in the string. \( f(aaa) = \) since there are no \( b\)'s. In this case there is no \( b\)'s 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.)

\[ |w| = n. \text{ What is the worst case running time (big-Oh) of your TM? } \mathcal{O}(n^2) \]
12. (10 pts) Construct a deterministic TM (using building blocks) that computes the following function:

\( w \in \{a, b\}^*, |w| \geq 2, f(w) = w' \) where \( w' \) is the middle of the string added to both ends of the string. If the string is of even length, then the middle is two characters, the first one is added to the start of the string and the second character is added to the end of the string. If the string is of odd length, then the single middle character is added to both the beginning of the string and the end of the string.

For example, \( f(ababa) = aababaab \) since the middle of the string is \( ab \), \( a \) is added to the start and \( b \) is added to the end. Another, \( f(aabab) = baababb \) since the middle of the string is \( b \), it is added to the start and the end.

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?

\( \mathcal{O}(n^2) \)
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. \( R_{ab} \) - move right until you see an \( a \) or \( b \)
8. \( L_{ab} \) - 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. \( \frac{a,b}{\rightarrow} \)
   
   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.