Question 1. [5 points] Consider the following regular expression:

\[(c|b)(a|ba)^*cc\]

For the strings shown below, circle the strings that are in the language generated by this regular expression, and cross out the strings that are not in the language.

\[\begin{array}{c}
  \times \quad \times \\
  bcc & bha \\
  bacc & bacc \\
  cab & cacc \\
  abacc & bab \\
\end{array}\]

Question 2. [5 points] Write a regular expression that generates all strings of symbols from the alphabet \{a, b, c\} in which all occurrences of c are part of a consecutive group of exactly 2 or 3 c's. Note that groups of 4 or more consecutive c's are not allowed.

Examples of strings in the language: 

- $\epsilon$
- aba
- accba
- abcc
- cccbacccbcbb

Examples of strings not in the language:

- cbcc
- aaccccb
- abac
- cab
- abccccc

\[(a|b)^* (\epsilon | cc | ccc) (a|b)(a|b)^* (cc | ccc))^* (a|b)^*\]
Question 3. [10 points] Write a regular language to generate all strings of symbols from the alphabet \{a, b\} where a and b strictly alternate.

Examples of strings in the language: \[ \epsilon \]
\[ a \]
\[ babab \]
\[ abab \]
\[ bababa \]

Examples of strings not in the language:
\[ bba \]
\[ abaa \]
\[ abba \]
\[ babaa \]
\[ aa \]

\[ \epsilon | (a (ba)^* (b | \epsilon)) | (b (ab)^* (a | \epsilon)) \]

Question 4. [10 points] Consider the deterministic finite automaton (DFA) shown on the right. (Note that the arrows labeled with multiple symbols should be considered to be multiple transitions, one on each symbol).

For the strings shown below, circle the strings that are in the language recognized by this DFA, and cross out the strings that are not in the language.

\[ \epsilon \]
\[ abaa \]
\[ bac \]
\[ cabb \]
\[ ccaabb \]

\[ abphacc \]
\[ accaa \]
\[ ccaabacc \]
\[ abacab \]
\[ bacca \]
Question 5. [10 points] Specify a deterministic finite automaton which recognizes the language of strings of symbols from the alphabet \{a, b, c\} where each string contains at most 3 occurrences of b.

Be sure to indicate the start state and final state(s), to indicate the direction of each transition, and to label each transition with a symbol.

Examples of strings in the language: 
\[ \epsilon \]
\[ abc \]
\[ bacc \]
\[ babbc \]
\[ abbaacb \]

Examples of strings not in the language: 
\[ bbababc \]
\[ bacababb \]
\[ babbab \]
\[ cbabbcba \]
\[ bbbabbbb \]
Question 6. [10 points] Consider the following nondeterministic finite automaton (NFA):

Create a deterministic finite automaton (DFA) that recognizes the same language as the NFA. Extra credit: use the conversion algorithm we discussed in class, and show a table mapping the reachable sets of NFA states to the equivalent states in your DFA.

<table>
<thead>
<tr>
<th>NFA states</th>
<th>DFA state</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ {0}</td>
<td>0</td>
</tr>
<tr>
<td>✓ {1, 2, 3, 6, 9}</td>
<td>1</td>
</tr>
<tr>
<td>✓ {4, 7}</td>
<td>2</td>
</tr>
<tr>
<td>✓ {5, 9, 2, 3, 6}</td>
<td>3</td>
</tr>
<tr>
<td>✓ {5, 9, 2, 3, 6}</td>
<td>4</td>
</tr>
</tbody>
</table>
Question 7. [10 points] Specify a context-free grammar (CFG) that generates all strings of terminal symbols chosen from \([a [ ] ( )] \) where the square brackets and parentheses are properly balanced.

Examples of strings in the language:  
\[
\epsilon \\
a \\
[a] \\
(a)[a] \\
a((a))[a]
\]

Examples of strings not in the language:  
\[
[a) \\
([a] \\
)a[a] \\
[(a)] \\
a][a
\]

Hint: make sure delimiters (brackets and parentheses) are added in balanced pairs.

Be sure to specify which nonterminal symbol is the start symbol. Each production should have a single nonterminal symbol on the left hand side.

\[
\text{start symbol is } E \\
E \rightarrow a \\
E \rightarrow \epsilon \\
E \rightarrow [E] \\
E \rightarrow (E) \\
E \rightarrow EE
\]
**Question 8.** [20 points] Consider the context-free grammar (CFG) shown on the right, which generates strings of symbols chosen from the alphabet \( \{ f, g, x, y, \lambda, (, ) \} \).

E is the start symbol.

(a) Show a derivation for the string

\[ (f \times) (\lambda y \cdot y) \]

(b) Draw the parse tree for the derivation you found in part (a).
Question 9. [5 points] Briefly explain why a context-free grammar must not use left recursion if you are going to implement a recursive descent parser for that grammar.

Consider, e.g., the production \( E \rightarrow E \cdot L \).
If this production were chosen in the parse function for \( E \):

\[
\text{parse } E() \ 	ext{ if} \\
\quad \text{if ( choose } \boxed{E \rightarrow E \cdot L} \text{ ) } \{ \\
\quad\quad \text{parse } E() \leftarrow \text{problem here} \\
\quad\quad \text{parse } L() \\
\}\]

The recursive call to parse \( E() \) is made without consuming any terminal symbols from the input string, so the recursive call will make the same choice, leading to an infinite recursion.

Question 10. [5 points] If \( A \) and \( B \) are regular languages, is \( A \cup B \) also a regular language? Explain briefly.

Yes. Assume we have NFAs for recognizing languages \( A \) and \( B \):

We can construct an NFA that accepts strings that are either in \( A \) or \( B \), in other words, the language \( A \cup B \):

Because this is also an NFA, by definition the language it accepts is a regular language.
Question 11. [10 points] Specify a Turing Machine that, given an initial tape consisting of a and b symbols, will change each a to a b and each b to an a, and then halt.

Assume that the symbol $\Delta$ marks empty locations on the input tape.

Make sure to indicate which state is the start state, which state(s) are final (halt) states, and for each transition, to indicate the input symbol, output symbol, and direction (L/R/S).