1. For each of the following regular expressions, describe or draw two finite-state machines:
   - An NFA that accepts the same language, obtained using Thompson’s recursive algorithm
   - An equivalent DFA, obtained using the incremental subset construction. For each state in your DFA, identify the corresponding subset of states in your NFA. Your DFA should have no unreachable states.

   (a) \((00 + 11)^*(\emptyset + 1 + \epsilon)\)

   **Solution:** Here is Thompson’s NFA for this regular expression. All unlabeled edges are \(\epsilon\)-transitions.

   ![Thompson’s NFA](image)

   The incremental subset construction builds a 6-state DFA as follows:

<table>
<thead>
<tr>
<th>(q')</th>
<th>(\epsilon)-reach((q'))</th>
<th>(q' \in A'?)</th>
<th>(\delta'(q',0))</th>
<th>(\delta'(q',1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s)</td>
<td>sabfklmnptuv</td>
<td>✓</td>
<td>co</td>
<td>gq</td>
</tr>
<tr>
<td>(co)</td>
<td>cdorv</td>
<td>✓</td>
<td>e</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>(gq)</td>
<td>ghqrv</td>
<td>✓</td>
<td>(\emptyset)</td>
<td>i</td>
</tr>
<tr>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
<td>✓</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>(e)</td>
<td>sabfklmnptuv</td>
<td>✓</td>
<td>co</td>
<td>gq</td>
</tr>
<tr>
<td>(i)</td>
<td>sabfklmnptuv</td>
<td>✓</td>
<td>co</td>
<td>gq</td>
</tr>
</tbody>
</table>

   We can shrink this DFA to just four states by merging \(s\), \(e\), and \(i\), which all have identical outgoing transitions. This improvement gives us the smallest DFA for \((01 + 10)^*(\emptyset + 1 + \epsilon)\).
(b) $1^* + (01)^* + (001)^*$

**Solution:** Here is Thompson’s NFA for this regular expression. All unlabeled edges are $\epsilon$-transitions.

![NFA Diagram]

The incremental subset construction builds a 9-state DFA as follows:

<table>
<thead>
<tr>
<th>$q'$</th>
<th>$\epsilon$-reach($q'$)</th>
<th>$q' \in A'$?</th>
<th>$\delta'(q', 0)$</th>
<th>$\delta'(q', 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>sabcefgklmnuv</td>
<td>✓</td>
<td>ho</td>
<td>d</td>
</tr>
<tr>
<td>$ho$</td>
<td>hiop</td>
<td></td>
<td>q</td>
<td>j</td>
</tr>
<tr>
<td>$d$</td>
<td>bcdelv</td>
<td>✓</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>$q$</td>
<td>qr</td>
<td></td>
<td>Ø</td>
<td>t</td>
</tr>
<tr>
<td>$j$</td>
<td>fgiklv</td>
<td>✓</td>
<td>$h$</td>
<td>Ø</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$t$</td>
<td>mntuv</td>
<td>✓</td>
<td>o</td>
<td>Ø</td>
</tr>
<tr>
<td>$h$</td>
<td>hi</td>
<td></td>
<td>$\emptyset$</td>
<td>$j$</td>
</tr>
<tr>
<td>$o$</td>
<td>op</td>
<td></td>
<td>q</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

All missing transitions in this drawing go to the reject state $\emptyset$. This is actually the smallest DFA for $1^* + (01)^* + (001)^*$.

**Rubric:** These solutions follow Thompson’s algorithm mechanically, interpreting any three-way sum $x + y + z$ as $(x + y) + z$; interpreting $x + y + z$ as $x + (y + z)$ yields slightly different but also correct results. No penalty for obvious small improvements to Thompson’s algorithm, like three-way branches for $x + y + z$ or simple paths without $\epsilon$-transitions for strings, as long as these improvements are applied consistently. These improvements will not change the final DFAs.

It is not necessary to include the table for the incremental subset construction, although it may help us give you partial credit. Comments about the minimal DFA (here in gray) are not required.
2. Give context-free grammars for the following languages, and clearly explain how they work and the role of each nonterminal. Grammars can be very difficult to understand; if the grader does not understand how your construction is intended to generate the language, then you will receive no credit.

(a) In any string, a **block** (also called a run) is a maximal non-empty substring of identical symbols. For example, the string $01100011001$ has six blocks: three blocks of 0s of lengths 1, 4, and 2, and three blocks of 1s of lengths 3, 2, and 1.

Let $L$ be the set of all strings in $\{0, 1\}^*$ that contain two blocks of 0s of equal length. For example, $L$ contains the strings $01101111$ and $01001011100010$ but does not contain the strings $000110011011$ and $00000000111$.

**Solution:**

\[
S \rightarrow ACB \quad \text{strings with two blocks of 0s of same length}
\]

\[
A \rightarrow \epsilon | X1 \quad \text{empty or ends with 1}
\]

\[
B \rightarrow \epsilon | 1X \quad \text{empty or starts with 1}
\]

\[
C \rightarrow 0C0 | 0D0 \quad 0^n y 0^n, \text{ where } y \text{ starts and ends with 1}
\]

\[
D \rightarrow 1 | 1X1 \quad \text{starts and ends with 1}
\]

\[
X \rightarrow \epsilon | 1X | 0X \quad \text{all possible strings: } (0 + 1)^n
\]

Every string in $L$ has the form $x0^ny0^nz$, where $x$ is either empty or ends with 1, $y$ starts and ends with 1, and $z$ is either empty or begins with 1. The same decomposition can be expressed more compactly as follows:

\[
S \rightarrow B | B1A | A1B | A1B1A \quad \text{strings with two blocks of 0s of same length}
\]

\[
A \rightarrow 1A | 0A | \epsilon \quad \text{all possible strings: } (0 + 1)^n
\]

\[
B \rightarrow 0B0 | 010 | 01A10 \quad 0^n y 0^n, \text{ where } y \text{ starts and ends with 1}
\]

(b) $L = \{w \in \{0, 1\}^* \mid w \text{ is not a palindrome}\}$.

**Solution:**

\[
S \rightarrow 0S0 | 0S1 | 1S0 | 1S1 | A \quad \text{non-palindromes}
\]

\[
A \rightarrow 0B1 | 1B0 \quad \text{strings that start and end with different symbols}
\]

\[
B \rightarrow 0B | 1B | \epsilon \quad \text{all strings}
\]

Every non-palindrome $w$ can be decomposed as either $w = x0^ny1z$ or $w = x1y0z$, for some substrings $x, y, z$ such that $|x| = |z|$. Non-terminal $S$ generates the prefix $x$ and matching suffix $z$; non-terminal $A$ generates the distinct symbols, and non-terminal $B$ generates the interior substring $y$.

**Solution:**

\[
S \rightarrow 0S0 | 1S1 | A \quad \text{non-palindromes}
\]

\[
A \rightarrow 0B1 | 1B0 \quad \text{strings that start and end with different symbols}
\]

\[
B \rightarrow 0B | 1B | \epsilon \quad \text{all strings}
\]
Every non-palindrome \( w \) must have a prefix \( x \) and a substring \( y \) such that either \( w = x0y1x^R \) or \( w = x1y0x^R \). Specifically, \( x \) is the longest common prefix of \( w \) and \( w^R \). In the first case, the grammar generates \( w \) as follows:

\[
S \rightarrow^* x A x^R \rightarrow^* x \epsilon B 1 x^R \rightarrow^* x \epsilon y 1 x^R = w
\]

The derivation for \( w = x1y0x^R \) is similar.

Rubric: 10 points = 5 for each part:
- 3 for a correct grammar. (These are not the only correct solutions.)
- 2 for a clear explanation of the grammar
3. Let \( L = \{ 0^i 1^j 2^{i+j} \mid i, j \geq 0 \} \).

   (a) Show that \( L \) is context-free by describing a grammar for \( L \).

   \[ \text{Solution:} \]
   \[
   S \rightarrow 0S2 \mid B \\
   B \rightarrow 1B2 \mid \varepsilon
   \]
   \[
   \{ 0^i 1^j 2^{i+j} \mid i, j \geq 0 \} \\
   \{ 1^j 2^i \mid i \geq 0 \}
   \]

   (b) Prove that your grammar \( G \) is correct. As usual, you need to prove both \( L \subseteq L(G) \) and \( L(G) \subseteq L \).

   \[ \text{Solution:} \]
   We will first prove a separate lemma that we will use in the solution.

   \[ \text{Lemma 1.} \ L(B) = \{ 1^i 2^i \mid i \geq 0 \}. \]

   \[ \text{Proof:} \]
   This proof is the same as the proof that \( L(C) = \{ 0^n 1^n \mid n \geq 0 \} \) which appears in page 4 of the lecture notes on context free grammars; just replace \( 0 \) and \( 1 \) with \( 1 \) and \( 2 \).

   \[ \square \]

   \[ \text{Lemma 2.} \ L \subseteq L(S) \]

   \[ \text{Proof (induction on } i) : \] Let \( w \) be an arbitrary string in \( L \). By definition, \( w = \theta^i 1^j 2^i \) for some non-negative integers \( i \) and \( j \). Assume that \( \theta^h 1^2^{i+j} \in L(S) \) for all non-negative integers \( h < i \). There are two cases to consider.

   \[ \text{• If } i = 0, \text{ then } w = 1^j 2^i. \] The previous lemma immediately implies \( S \Rightarrow B \Rightarrow \varepsilon \) \( w \).

   \[ \text{• Suppose } i > 0. \] Then \( w = \theta \cdot \theta^{i-1} 1^2^{i+j-1} \cdot 2 \). The inductive hypothesis implies \( S \Rightarrow \varepsilon \theta^{i-1} 1^2^{i+j-1} \in L(S) \). It follows that \( S \Rightarrow \theta S1 \Rightarrow \varepsilon \) \( w \).

   In both cases, we conclude that \( S \Rightarrow \varepsilon \) \( w \).

   \[ \square \]

   \[ \text{Proof (induction on } w) : \] Let \( w \) be an arbitrary string in \( L \). Assume that \( L(S) \) contains every string \( x \in L \) such that \( |x| < |w| \). There are three cases to consider.

   \[ \text{• If } w = \varepsilon, \text{ then } S \Rightarrow B \Rightarrow \varepsilon. \]

   \[ \text{• Suppose } w = \theta x \text{ for some string } x. \] Then \( w = \theta^i 1^j 2^i \) where \( i > 0 \), so \( w \) must end with \( 2 \). Thus, we have \( w = \theta y 2 \), where \( y \in L \). The induction hypothesis implies \( y \in L(S) \). We conclude that \( S \Rightarrow \theta S2 \Rightarrow \varepsilon \) \( w \).

   \[ \text{• Suppose } w = 1^i x \text{ for some string } x. \] Then \( w = 1^2^j \) for some \( j > 0 \), and therefore \( S \Rightarrow B \Rightarrow \varepsilon \) \( w \) by Lemma 1.

   In both cases, we conclude that \( S \Rightarrow \varepsilon \) \( w \).

   \[ \square \]

   \[ \text{Lemma 3.} \ L(S) \subseteq L \]

   \[ \text{Proof:} \] Let \( w \) be an arbitrary string in \( L(S) \). Assume \( L \) contains every string \( x \in L(S) \) such that \( |x| < |w| \). There are two cases to consider

   \[ \text{• Suppose } w = \theta x 2 \text{ for some } x \in L(S). \] The induction hypothesis implies that \( x = \theta^i 1^j 2^{i+j} \) for some integers \( i \) and \( j \). It follows that \( w = \theta^{i+1} 1^2^j \) and therefore \( w \in L \).

   \[ \text{• Suppose } w \in L(B). \] Lemma 1 implies that \( w = 1^2^l \) for some integer \( l \). It follows immediately that \( w = \theta^0 1^2 0^l \in L \).

   \[ \]
In both cases, we conclude that $w \in L$. □

Together, Lemmas 2 and 3 imply that $L = L(S)$. ■

Rubric: 10 points:
- part (a) = 4 points. As usual, this is not the only correct grammar.
- part (b) = 6 points = 3 points for $\subseteq$ + 3 points for $\supseteq$, each using the standard induction template (scaled).