1. (a) the language is regular; it is given by the regular expression \((aa)^*\).
   (b) the language is regular; it is given by the regular expression \(a(aa)^*\).
   (c) the language is not regular, which can be show using the Pumping Lemma; in fact according to the lecture slides, it is not a context-free language.

2. (a)
   \[
   S \rightarrow \varepsilon \\
   S \rightarrow aShSa \rightarrow a\varepsilon\varepsilon a \rightarrow aba \\
   S \rightarrow aShSa \rightarrow aaShSab\varepsilon a \rightarrow aa\varepsilon\varepsilon aba \rightarrow aababa \\
   S \rightarrow aShSa \rightarrow aaShSab\varepsilon Sa \rightarrow aa\varepsilon\varepsilon aba\varepsilon\varepsilon a \rightarrow aabababa 
   \]

(b) \(bb, aa, abba, ab\)

3. 
   \[
   S \rightarrow XbY \\
   X \rightarrow aX \\
   X \rightarrow bX \\
   X \rightarrow \varepsilon \\
   Y \rightarrow abY \\
   Y \rightarrow \varepsilon 
   \]

4. (a) \(a^* \cup b^*\);
   (b) \((bb(ab))^* \cup a^*\)^*;
   (c) all words containing at least two \(a\)'s: \((a \cup b)^*a(a \cup b)^*a(a \cup b)^*\).

5. (a)
   \[
   S \rightarrow 0T \\
   S \rightarrow 1T \\
   T \rightarrow 0S \\
   T \rightarrow 1S \\
   T \rightarrow \varepsilon 
   \]

(b)
   \[
   S \rightarrow 0S0 \\
   S \rightarrow 0S1 \\
   S \rightarrow 1S0 \\
   S \rightarrow 1S1 \\
   S \rightarrow 0 
   \]

6. 
   \[
   S \rightarrow c \\
   S \rightarrow aSa \\
   S \rightarrow bSb 
   \]

7. (a)
   \[
   S \rightarrow aSb \\
   S \rightarrow aS \\
   S \rightarrow \varepsilon 
   \]

(b)
   \[
   S \rightarrow XY \\
   X \rightarrow ab \\
   X \rightarrow aXb \\
   Y \rightarrow cd \\
   Y \rightarrow cYd 
   \]

8. The automaton accepts the words of the form \(a^n bw\), where \(n \geq 0\), and \(w\) is any word of length \(n\). In particular, it rejects \(aaaba\) and accepts \(abb\) and \(aabba\).
9. (a) A PDA for the language from problem 5 is shown below:

(b) A PDA for the language \( L_1 = \{a^n b^m \mid 0 \leq m \leq n \} \):

(c) A PDA for the language
\[
L_2 = \{a^n b^n c^m d^m \mid m, n \geq 1 \}
\]

The following grammar turns \( Z \) into a sequence +1’s and -1’s adding up to zero, or into an empty word. Thus it turns A into a sequence of +1’s and -1’s adding up to 1, and B to -1. The first column ensures that the initial symbol \( S \) is transformed into \( A^n = A^n \) or \( B^n = B^n \) or to one of the first four equalities.

\[
\begin{align*}
S &\to AB = AB \\
S &\to AB = BA \\
S &\to BA = AB \\
S &\to BA = BA \\
S \to S_1 & A \to +1Z \\
S \to S_1 & A \to \epsilon \\
S_1 \to AS_1' & A \to Z + 1 \\
S_2 \to BS_2' & B \to -1Z \\
S_2 \to BS_2' & B \to Z - 1 \\
S_1' \to AS_1' & Z \to \epsilon \\
S_2' \to BS_2' & Z \to ZZ \\
S_2' \to BS_2' & Z \to +1Z - 1 \\
S_2' \to BS_2' & Z \to -1Z + 1 \\
A &\to -1AAA - 1 \\
B &\to +1BBB + 1
\end{align*}
\]

The PDA for this language is given on the following page: