1. Let \( L \) be the language generated by the grammar \( G \) below:

\[
S \rightarrow XY | YYY \\
X \rightarrow Yb | \epsilon \\
Y \rightarrow aY | X.
\]

Transform \( G \) into an equivalent grammar in Chomsky Normal Form. Show your work.

2. Give a grammar for each language below. You must explain the role of each nonterminal, and describe how your grammar works.

- \( L_1 = \{ w \in \{0, 1\}^* \mid |w| \equiv 0 \mod 2 \text{ and } w \text{ is not a palindrome} \} \)
- \( L_2 = \{ a^ibjc^k \mid j = 2i \text{ or } k = 2j \text{ where } i, j, k \geq 0 \} \)
- \( L_3 = \{ w \in \{0, 1\}^* \mid \text{there is no } x \in \{0, 1\}^* \text{ such that } w = xx \} \)

3. A middle-pop PDA works like a PDA but the pop operation gets its stack symbol from the middle (the one closer to the top in case there are two middle symbols) of the stack instead of the top. Push operations still occur on the top of the stack. Further, the middle-pop PDA cannot view the top of the stack unless it is also the middle of the stack (that is, there are only one or two symbols in the stack). Moreover, in a middle-pop PDA, the only way to read the middle symbol is to “pop it” from the middle of the stack. Thus, in a single step, the middle-pop PDA may either (1) read an input character (or \( \epsilon \)), and based on the character read, but without regard to the unviewed middle-of-stack symbol, it pushes a string \( \gamma \) on the top of the stack, and changes states, or else it can (2) pop and view the middle stack symbol, and based in both that symbol and the input character (or \( \epsilon \)), choose to push a string on the (top of the) stack and change state. How does the class of languages accepted by middle-pop PDAs relate to the class of CFLs? As always, prove that your answer is correct.

4. Construct a PDA for the language consisting of all strings over \( \{0, 1\}^* \) that are not of the form \( \{(0^i1^j)^k \mid 1 \leq i < j, k < 2i\} \). You must explain how your PDA works.

5. A two-stack PDA is a PDA with two distinct stacks. In a single move, it reads input (or \( \epsilon \)), and, based on the current top-of-stack symbols on each of its stacks, and on its current state, changes state, pushing or popping either (or both) of the stacks independently.

- Formalize the definition of a two-stack PDA as 7-tuple, explaining what each component of the tuple denotes.
- Are the class of languages accepted by regular PDAs and two-stack PDAs the same? Prove your answer [Hint: Can you think of a language not accepted by a two-stack PDA?]
6. As seen in class, a nondeterministic recursive automaton over an alphabet \( \Sigma \) is a tuple 
\((M, main, \{(Q_m, \Sigma \cup M, \delta_m, q^0_m, F_m)\}_{m \in M})\), in which:

- \( M \) is a finite set of module names.
- \( main \in M \) is the name of the initial module.
- For each \( m \in M \), \( A_m = (Q_m, \Sigma \cup M, \delta_m, q^0_m, F_m) \) is an NFA over the alphabet \( \Sigma \cup M \).
- For any \( m, m' \in M \), \( Q_m \cap Q_{m'} \neq \emptyset \iff m = m' \).

(a) Prove that for any nondeterministic recursive automaton, there exists a PDA that can simulate it.

(b) Design a nondeterministic recursive automaton for the language \( L \) containing all strings \( w \in \{0, 1\}^* \) that have an equal number of 0s and 1s.