1. Prove that the language \( \{0^n1^{2n}\} = \{\lambda, 011, 001111, 00011111, \ldots\} \) is not regular.

\[
L = \{0^n1^{2n}\} \text{ is not regular}
\]

**Proof by contradiction**

Let's assume that \( L \) is regular. Then by Pumping Lemma, there exists \( p \) such that every \( w \in L \) whose length is at least \( p \) can be represented as \( x \gamma z \), so that \( \text{len}(x) > 0 \), \( \text{len}(x \gamma) \leq p \), and for every \( i, xy^i z \in L \).

Let us take \( w = 0^p1^{2p} = 0^{\frac{p}{2}} 1^{\frac{2p}{2}} \), so \( w \in L \).

\[
\text{len}(w) = 3p \geq p, \text{ and since } \text{len}(x \gamma) \leq p, x \text{ and } \gamma \text{ are among the first } p \text{ characters of } w, \text{ and these characters are '0', so } \gamma \text{ consists of '0' only. By Pumping Lemma } xy^i z \text{ is in } L \text{, but when we go from } xyz \text{ to } x\gamma z, \text{ we add } \gamma \text{ i.e. we add '0's but we do not add any '1's, so } xy^i z \in L. \text{ A contradiction, so our assumption that } L \text{ is regular is false, thus } L \text{ is not regular.}
\]
2. Design a pushdown automaton that would recognize the words of the type $a^{n+1}b^n$, i.e., the language $L = \{a, aab, aaabb, \ldots\}$. Show, step by step, how your pushdown automaton will recognize the word $aab$. Hint: use a pushdown automaton for recognizing the words of the type $a^n b^n$ as a sample.
3. Design a context-free grammar that would generate all the words of the type $a^{n+1}b^n$, i.e., the language $L = \{a, aab, aaabb, \ldots \}$. Show, step by step, how your grammar will generate the word aab.

*Hint:* use a context-free grammar for generating the words of the type $a^nb^n$ as a sample. There, we had rules $S \rightarrow \varepsilon$ and $S \rightarrow aSb$, now a slight modification is needed.

![Diagram of a context-free grammar](file://Q:/cs3350.18/test2.html)

- **CFG**
  1. $S \rightarrow A_{sf}$
  2. $A_{sf} \rightarrow A_{AB}$
  3. $A_{AB} \rightarrow aA_{AB}b$
  4. $A_{AB} \rightarrow aA_{AA} \varepsilon$
  5. $A_{AA} \rightarrow \varepsilon$

- **Recognize:** $aa \ b$

![parse tree](file://Q:/cs3350.18/test2.html)
4. Use a general algorithm that we had in class to generate a context-free grammar that corresponds to the following finite automaton for recognizing valid Java identifiers. For simplicity, assume that we only allow letters a and b and digits 0 and 1. This automaton has three states: the starting state s, the final state f, and the sink state k.

- From s, any letter (a or b) lead to f, while any digit leads to k.
- From f, any symbol leads to f.

Show, step by step, how your grammar generates a word b1a.
5. Transform, step by step, the grammar with rules $S \rightarrow \epsilon$, $S \rightarrow 0S0$, and $S \rightarrow 1S1$ to Chomsky normal form. Show how the word 1001 will be generated in the resulting Chomsky-normal-form grammar.

**Step 0**

- $S_0 \rightarrow \epsilon$ (1)
- $S \rightarrow 00$ (2)
- $S \rightarrow 11$ (2)

**Step 1**

- $S_0 \rightarrow 0S0$ (2)
- $S_0 \rightarrow 1S1$ (2)
- $S_0 \rightarrow 00$ (2)
- $S_0 \rightarrow 11$ (2)

**Step 2**

- $V_0 \rightarrow 0$
- $V_1 \rightarrow 1$
- $S \rightarrow V_0SV_0$ (3)
- $S \rightarrow V_1SV_1$ (3)
- $S \rightarrow V_0V_0$
- $S \rightarrow V_1V_1$
- $S_0 \rightarrow V_0SV_0$ (3)
- $S_0 \rightarrow V_1SV_1$ (3)
- $S_0 \rightarrow V_0V_0$
- $S_0 \rightarrow V_1V_1$

**Step 3**

- $S \rightarrow V_{05}V_0$
- $S \rightarrow V_{15}V_1$
- $V_{05} \rightarrow V_0S$
- $V_{15} \rightarrow V_1S$
- $S_0 \rightarrow V_{05}V_0$
- $S_0 \rightarrow V_{15}V_1$

All rules

1) $S_0 \rightarrow \epsilon$
2) $V_0 \rightarrow 0$
3) $V_1 \rightarrow 1$
4) $S \rightarrow V_1V_1$
5) $S \rightarrow V_0V_0$
6) $S_0 \rightarrow V_1V_1$
7) $S_0 \rightarrow V_0V_0$
8) $S \rightarrow V_{05}V_0$
9) $S \rightarrow V_{15}V_1$
10) $S_0 \rightarrow V_{05}V_0$
11) $S_0 \rightarrow V_{15}V_1$

Recognize: 1001
6. Use a general algorithm for transforming CFG into PDA to design a pushdown automaton which is equivalent to the grammar with rules $S \rightarrow \varepsilon$, $S \rightarrow 0S0$, and $S \rightarrow 1S1$. Show, step by step, how the word 1001 will be accepted by the resulting pushdown automaton.

(file://Q:/cs3350.18/test2.html)
7. Use a general algorithm for transforming PDA into CFG to design a CFG that corresponds to the following pushdown automaton. This automaton has two states: the starting a-state \( s_a \) and the b-state \( s_b \). Both states are final. The transitions are:

- From \( s_a \) to \( s_a \), the transition is \( a, \varepsilon \rightarrow x \).
- From \( s_a \) to \( s_b \), the transition is \( b, x \rightarrow \varepsilon \).
- From \( s_b \) to \( s_b \), the transition is \( b, x \rightarrow \varepsilon \).

Show, step by step, how the word \( ab \) will be generated by the resulting grammar.

\[
\begin{align*}
S & \rightarrow A_{s_a}s_b \\
A_{s_a}s_b & \rightarrow aA_{s_a}s_b b \\
A_{s_b}s_b & \rightarrow a_{s_a}s_a b \\
A_{s_a}s_a & \rightarrow \varepsilon
\end{align*}
\]
8. (for extra credit) Use the general stack-based algorithms to show:

- how the compiler will transform the expression \((10 - 1) / (11 - 8)\) into postfix form, and
- how it will compute the value of the resulting postfix expression.