1. Prove that the language \( \{0^n1^2n\} = \{\lambda, 011, 001111, 000111111, \ldots\} \) is not regular.

Proof by contradiction:

Let's assume that \( L \) is regular.

By pumping lemma, there exist \( p \) such that every word \( w \in L \) whose length is at least \( p \) can be represented as \( xy^iz \), so that \( |y| \geq 0 \), \( |xy| \leq p \) and for every \( xy^iz \in L \).

Let's take:

\[
w = 0^p1^2p = \underbrace{0\ldots01\ldots1}_{2p} \in L
\]

\( |w| = 3p > p \), \( w = xy^iz \)

Since \( |xy| \leq p \), \( xy \) is among first \( p \) characters of \( w \) and these characters are 0s only, so \( y \) has 0s only.

\( xy^iz = xyyz \Rightarrow \) we will be adding only 0s, and not 1s, therefore \( xyyz \in L \).

While pumping lemma states that \( xyyz \in L \).

This is a contradiction to our assumption that \( L \) is regular. Therefore, \( L \) 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 $a^n b^n$ as a sample. There, we had rules $S \rightarrow \varepsilon$ and $S \rightarrow aSb$, now a slight modification is needed.

\[
S \rightarrow a \\
S \rightarrow aSb
\]
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.

\[ S \rightarrow aF \\
S \rightarrow bF \\
S \rightarrow 0k \\
S \rightarrow 1k \\
F \rightarrow aF \\
F \rightarrow bF \\
F \rightarrow 0F \\
F \rightarrow 1F \\
F \rightarrow \epsilon \\
\]

Every final state leads to $\epsilon$. 
5. Transform, step by step, the grammar with rules $S \rightarrow \varepsilon$, $S \rightarrow 0S0$, 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:**
- $S \rightarrow 00$
- $S \rightarrow 11$
- $S_0 \rightarrow \varepsilon$

**Step 1:**
- $S_0 \rightarrow 0S0$
- $S_0 \rightarrow 1S1$
- $S_0 \rightarrow 00$
- $S_0 \rightarrow 11$

**Step 2:**
- $V_0 \rightarrow 0$
- $V_1 \rightarrow 1$
- $S \rightarrow V_0 V_0$
- $S \rightarrow V_1 V_1$
- $S_0 \rightarrow V_0 V_0$
- $S_0 \rightarrow V_1 V_1$

**Step 3:**
- $V_{05} \rightarrow V_0 S$
- $S \rightarrow V_{03} V_0$
- $V_{15} \rightarrow V_1 S$
- $S \rightarrow V_{13} V_1$
- $S_0 \rightarrow V_{05} V_0$
- $S_0 \rightarrow V_{15} V_1$

(file://Q:/cs3350.18/test2.html 2/28/2018)
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.

In the intermediary state we pop $S$ and push $1S1$.

Both intermediary states push 1 by 1, but since I ran out of time, I had to put the short version.
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.

```
S \rightarrow A_{s_a s_b}
S \rightarrow A_{s_b s_b}

A_{s_a s_a} \rightarrow \varepsilon
A_{s_b s_b} \rightarrow \varepsilon

S_a, \varepsilon \rightarrow x
S_b, b, \varepsilon \rightarrow \varepsilon

A_{s_a s_b} \rightarrow a A_{s_a s_a}
A_{s_b s_b} \rightarrow a A_{s_b s_b}
```

file:///Q:/cs3350.18/test2.html
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.