1-2. Use a general algorithm to transform the following finite automaton into the corresponding regular expression.

- This automaton has two states: a state $s_1$ which is a start state and a final state, and another state $s_2$.
- In the state $s_1$, $a$ and $b$ lead to $s_2$, and $c$ leads to $s_1$.
- In the state $s_2$, $a$ and $c$ lead to $s_1$, and $b$ leads to $s_2$. 

\[
R_{\text{NSL}} = R_{s_1} \cup (R_{s_2} a b^* c)
\]

\[
R_{\text{NSM}} = R_{s_1} \cup (R_{s_2} a b^* (auc))
\]

\[
R_{11} = R_{11} \cup (R_{12} a b^* c)
\]

\[
R_{11} = R_{11} \cup (R_{12} a b^* (auc))
\]

\[
R_{11} = R_{11} \cup (R_{12} a b^* (auc))
\]

\[
\emptyset \cup (\emptyset \cup (a b^* c))
\]

\[
\emptyset \cup (\emptyset \cup (a b^* (auc)))
\]
$R_{	ext{DFA}} = R_{	ext{DFA}} \cup (R_{	ext{DFA}} \times R_{	ext{DFA}})^*$

$= \bigcup (a, b)^* (a, c)^*$

$= \bigcup (a, b)^* (a, c)^*$
3. Automaton from Problem 1-2 accepts the word abc.

- Trace, step-by-step, that this word is indeed accepted by the given automaton.
- Use this tracing to find the parts x, y, and z of this word corresponding to the Pumping Lemma.
- Trace that the words xy^2z and xy^3z are also accepted by this automaton.
4. Prove that the language $\{a^nb^{2n}\} = \{\lambda, abb, aabbbb, aaabbbbb, \ldots\}$ is not regular.

Let's assume $L = \{a^nb^{2n}\}$ is a LL (Recursive Language). Let's derive a contradiction from this assumption. Since $L$ is a LL by PL (Pumping Lemma), there exists an integer $p$ such that every word $w$ from $L$ whose $\text{len}(w) \geq p$ can be represented as $xyz$ where $\text{len}(y) > 0$, $\text{len}(xy) \leq p$, and $xy^iz \in L$. Let's take $w = a^pb^{2p} \in L$, $\text{len}(xy) \leq p$, $w = xyz$ satisfies with $xy$ so $y$ is in $a^*$, in $\text{xyz}$, we add $a$'s, but not $b$'s. We started with a word that has twice as many $b$'s than $a$'s, but $\text{xyz} \in L$, but by PL $\text{xyz} \in L$. We get a contradiction. So our assumption that $L$ is a LL is false. Thus, $L$ is not regular.
5. Design a pushdown automaton that would recognize the words of the type $a^n b^{n+2}$, i.e., the language $L = \{bb, abbb, aabbbb, \ldots \}$. Show, step by step, how your pushdown automaton will recognize the word $abbb$. 

*Hint:* use a pushdown automaton for recognizing the words of the type $a^b^n$ as a sample.
6. Design a context-free grammar that would generate all the words of the type $a^n b^{n+2}$, i.e., the language $L = \{bb, abbb, aabbbb, \ldots\}$. Show, step by step, how your grammar will generate the word abbb. *Hint:* use a context-free grammar for generating the words of the type $a^n b^p$ as a sample. There, we had rules $S \rightarrow \varepsilon$ and $S \rightarrow aSb$, now a slight modification is needed.
7. Use a general algorithm that we had in class -- for transforming a finite automaton into a context-free grammar -- to generate a context-free grammar that corresponds to the following finite automaton for recognizing unsigned integers. 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 k, while any digit leads to f.
- From f, any digit leads to f, and letter leads to k.

Show, step by step, how your grammar generates a word 010.
8. Use a general algorithm for transforming CFG into PDA to design a pushdown automaton which is equivalent to the grammar with rules $S \rightarrow \varepsilon$, $S \rightarrow 0S1$, and $S \rightarrow 1S0$. Show, step by step, how the word 1010 will be accepted by the resulting pushdown automaton.
9. (For extra credit) Prove that the following language is not context-free: \( \{a^{2n}b^n c^{2n}\} = \{\Lambda, aabcc, aaaabbccece, \ldots\}. \)

Let's assume \( L \) is CFG. Let's take \( w = a^p b^p c^p = a \_ b \_ \ldots \_ c \_ \) where \( \text{LEN}(w) = 3p \leq P \). Thus \( \text{J} = \{w, x, y, z, wuxyzw \} \) where \( \text{LEN}(wuxyzw) = 7p \). By PL, we cannot have \( uv^y \) covering \( a's, b's, c's \) since \( \text{LEN}(uwy) > P \). So we have the cases:

1. \( uv^y \) is in \( b's \). When we pump, we add \( b's \), but not \( a's \) or \( c's \).

2. \( uv^y \) is in some combination of \( a's \), \( b's \), \( c's \) where \( \text{LEN}(uwy) < P \). When we pump, we pump \( a's \) or \( b's \), but not \( c's \).

3. \( uv^y \) is in some combination of \( b's \), \( c's \) where \( \text{LEN}(uwy) < P \). When we pump, we pump \( b's \) or \( c's \), but not \( a's \).

4. \( uv^y \) is in \( a's \) such that \( \text{LEN}(uwy) < P \). When we pump, we pump \( a's \), but not \( b's \) or \( c's \).

5. \( uv^y \) is in \( c's \) such that \( \text{LEN}(uwy) < P \). When we pump, we pump \( c's \), but not \( a's \) or \( b's \).

In all possible cases, we get a contradiction, so \( L \) is not CFG.
10. (For extra credit) On the example of the word 1010 generated by a grammar from Problem 8, show how this word will be represented as uvxyz according to the Pumping Lemma. Show how the words uxz and uvvxyz will be generated by this grammar.