1. Use the algorithm that we had in class to transform the following context-free grammar into Chomsky normal form: 

\[ S \rightarrow AA, \quad S \rightarrow ASa, \quad S \rightarrow aB, \quad B \rightarrow bA, \quad A \rightarrow \varepsilon, \quad B \rightarrow \varepsilon. \]
2. Use the general algorithm to design a (non-deterministic) pushdown automaton that recognizes exactly the context-free language described in Problem 1.

\[
\begin{align*}
S & \rightarrow AA \\
S & \rightarrow ASa \\
S & \rightarrow aB \\
B & \rightarrow bA \\
A & \rightarrow \epsilon \\
B & \rightarrow \epsilon \\
\end{align*}
\]
3. In the context-free grammar described in Problem 1, we can derive the sequence abaa as follows:

- first, we use the rule \( S \rightarrow ASa \);
- then, we use the rules \( A \rightarrow \varepsilon \) and \( S \rightarrow aB \);
- after that, we use the rule \( A \rightarrow aB \);
- finally, we use the rule \( B \rightarrow bA \) and \( A \rightarrow \varepsilon \).

Draw a tree that describes this derivation. Use this tree to determine the subdivision of this sequence into \( u, v, x, y, \) and \( z \). Then, for \( i = 0 \) and \( i = 2 \), draw trees explaining how the corresponding sequences \( uv^i x y^i z \) can be derived in this grammar.
4. Use pumping lemma for pushdown automata to prove that the language consisting of all the words of the type $a^{2n}b^n c^n$ cannot be recognized by a pushdown automaton.

Proof: by contradiction

Let's assume that $L$ can be recognized by CFG. This means there exists a $p$ in all words of the language $L$ with $\text{len}(w) \geq p$ and there exists $u, v, x, y, z$ with $w = uvxyz$ and $\text{len}(vxy) > 0$ and $\text{len}(vxy) \leq p$ and for all $i$ in $uv^i x y^i z$ still pertains to the language. Take $w = a^pa^pb^pc^pc^pc^p$

$a\ldots a a\ldots a b b\ldots b c c\ldots c c\ldots c$

- If $vxy$ is in all as', then we add more as and not bs and cs making the new word not part of the language. The same is $vxy$ is in all bs or in all cs.

- $vxy$ can be in between as and bs', then by adding more as and bs and not cs, the new word will not be part of the language.

- $vxy$ can be in between bs and cs, then by adding more bs and cs and not as, the new word will not be part of the language.

We get a contradiction because by pumping we will get a word that is not part of the language, therefore $L$ cannot be recognized by a pushdown automaton.
5. Use the general algorithm to design a context-free grammar that is equivalent to the following finite automaton. This automaton has two states:

- a state $q_1$ which is both a start state and a final state, and
- a state $q_2$.

In the state $q_1$, 0 leads to $q_2$, and 1 leads to $q_1$. In the state $q_2$, 0 leads to $q_2$, and 1 leads to $q_1$.

```
S → Q_1
Q_1 → ε
Q_1 → 0Q_2
Q_1 → 1Q_1
Q_2 → 0Q_2
Q_2 → 1Q_1
```
6. Use the general stack-based algorithms to show:

- how the compiler will transform the expression \(2 - (3 - 4 \times 5)\) into inverse Polish notation, and
- how it will compute the value of this expression.

\[
\begin{array}{c}
2 \quad - \quad (3 \quad - \quad 4 \quad \times \quad 5) \\
\hline
2 \quad 3 \quad 4 \quad 5 \quad \times \quad \_ \quad \_ \\
\hline
2 \quad - \quad (3 \quad - \quad 20) \\
2 \quad - \quad (-17) \\
19
\end{array}
\]
7-8. Design a Turing machine that computes the following function in unary code: \( f(0) = 0 \) and \( f(n) = n - 1 \) for \( n > 0 \). Trace it on the example of \( n = 2 \). Show, step-by-step, how the tape of the Turing machine can be represented as two stacks.

\[
\begin{align*}
F(0) &= 0 \\
F(n) &= n - 1 \text{ for } n > 0 \\
\end{align*}
\]

- push \( n \) into Stack 1 (S1)
- pop once from Stack 1 (S1)
- pop what's in Stack 1 (S1)
- and push it into Stack 2 (S2)
- Until S1 is empty
- Answer will be in Stack 2 (S2)

\[
\begin{align*}
\text{Step 1} & \quad n = 1 \\
\text{Step 2} & \quad \\
\text{Step 3} & \quad \\
\end{align*}
\]

**Church-Turing thesis**

Anything that can be computed on any physical device can also be computed on a Turing Machine (or Java program).

Church-Turing thesis is not a mathematical theorem; it is a statement about the physical world.
10. Formulate the current definition of a feasible algorithm. Give two examples explaining why this definition is not perfect:

- an example of an algorithm that is feasible according to this definition but not feasible according to common sense; and
- an example of an algorithm that is feasible from the practical viewpoint but not feasible according to this definition.

An algorithm \( A \) is called feasible if there exists a polynomial \( P \) such that for every input \( x \), the run-time \( t_A(x) \) of this algorithm is bounded by \( P(len(x)) \), \( t_A(x) \leq P(len(x)) \).

Not a perfect definition because:

1) \( t_A(x) = 20^{100} \cdot len(x) \) \( \rightarrow \) Feasible in the sense of the definition, not feasible in practice

2) \( t_A(x) = 2^{10^{-15}} \cdot len(x) \) \( \rightarrow \) Feasible in practice but not feasible according to the formal definition.
11. (For extra credit) Define what it means for a problem to be NP-hard. Can every NP-hard problem be solved by a feasible algorithm?

A problem is said to be NP-hard if an algorithm for solving it can be translated into one for solving any other NP-problem, meaning that every problem from the class NP can be reduced to this problem.

It is not known if every NP-hard problem or every NP problem overall can be solved by a feasible algorithm, is still an unsolved problem.