1. Use the algorithm that we had in class to transform the following context-free grammar into Chomsky normal form: $S \rightarrow AA, S \rightarrow aSA, S \rightarrow Ba, B \rightarrow Ab, A \rightarrow e, B \rightarrow e.$
2. Use the general algorithm to design a (non-deterministic) pushdown automaton that recognizes exactly the context-free language described in Problem 1.

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

- first, we use the rule $S \rightarrow aSA$;
- then, we use the rules $S \rightarrow Ba$ and $A \rightarrow \varepsilon$;
- after that, we use the rule $A \rightarrow B$; $B \rightarrow A \rightarrow \varepsilon$; $A \rightarrow \varepsilon$;
- finally, we use the rule $B \rightarrow \varepsilon$.

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 $uv'xy'z$ can be derived in this grammar.

$uv'xy'z = a^i ba$
4. Use pumping lemma for pushdown automata to prove that the language consisting of all the words of the type $a^{3n}b^n c^{2n}$ cannot be recognized by a pushdown automaton.

$$L = \{a^{3n}b^n c^{2n} : n \geq 0\}$$

$\Rightarrow$ assume $L$ is a CFL

by pumping lemma: $A \subseteq E, \exists p, \forall s \in A \ (|s| \geq p) \exists u,v,x,y,z \exists$

* $\text{len}(uv) > p$
* $\text{len}(vxy) = p$
* $\forall i \ uv^i x y^i z \in A$

$\Rightarrow s = a^{3p} b^p c^{2p} \ , \text{len}(s) \geq p$

$s = uv^1 xy^1 z , i = 1$

$s' = uv^2 xy^2 z , i = 2$

$\Rightarrow$ cases for $vxy$

1. $vxy \in a's$

$\Rightarrow$ if so, we will find $v^2xy^2 > vxy^1 + vxy^2$ so we will have more $a$'s assumption will be wrong $\Rightarrow$ #a's $\neq 3p$ so this will lead to imbalance of a's + b's + c's ($s' \not\in L$)

2. $vxy \in a's + b's$

$\Rightarrow$ again, messes up the balance between #a's, b's + c's inside $s' \not\in L$

3. $vxy \in b's$

$\Rightarrow$ same

4. $vxy \in b's + c's$

$\Rightarrow$ same

5. $vxy \in c's$

$\Rightarrow$ same

6. $vxy \notin a's + b's + c's$ because $\text{len}(vxy) \leq p$ and cannot cover more than #b's

$\Rightarrow$ our assumption is wrong in all cases $\Rightarrow s' \not\in L \Rightarrow$ cur Y is NOT a CFL represented by a PDA/CFG
5. Use the general algorithm to design a context-free grammar that is equivalent to the following finite automaton. This automaton has two states:

- the starting state q₁, and
- the final state q₂.

In the state q₁, 0 leads to q₁, and 1 leads to q₂. In the state q₂, 0 leads to q₁, and 1 leads to q₂.

\[ q₁ \rightarrow 0 q₂ \]
\[ q₁ \rightarrow 1 q₂ \]
\[ q₂ \rightarrow 0 q₁ \]
\[ q₂ \rightarrow 1 q₂ \]
\[ q₂ \rightarrow \epsilon \]
6. Use the general stack-based algorithms to show:

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

\[ 2 \times (3 - 4) - 5 \]

\[ 2 \quad 3 \quad 4 - \star 5 - \rightarrow 2 \quad 3 \quad 4 - \star 5 - \]

\[ 2 \quad 3 \quad 4 \quad \star \quad 5 \quad \star \quad 5 \quad - \quad - \]

\[ 2 \quad 3 \quad 4 \quad 5 \quad 5 \quad 7 \]
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 \( n = 1 \). Show, step-by-step, how the tape of the Turing machine can be represented as two stacks.

\[
\begin{align*}
&\text{(start, 4) } \rightarrow \text{(R, goEnd)} \\
&\text{(goEnd, 4) } \rightarrow \text{(L, back, 1)} \\
&\text{(goEnd, 1) } \rightarrow \text{R} \\
&\text{(back, 1) } \rightarrow \text{L} \\
&\text{(back, 4) } \rightarrow \text{halt}
\end{align*}
\]

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.

**Feasible algorithm:** \( t_A(x) \leq p(\text{len}(x)) \)

In other words, an algorithm whose time to run is less than a polynomial function of the length of the variable.

**Examples:**

- Feasible by definition but not in practice
  \( t_A(x) = 10^{50}x^2 \)
  \( 10^{50} \) is a magnitude too big that it wouldn't be computable in practice.

- Feasible in practice but not in definition
  \( t_A(x) = 5 \cdot 10^{-20}x \)
  The function is exponential, however \( 10^{-20} \) is a very small number meaning that if we had \( x = 10^{-20} \) the time would be 5, thus feasible.
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.