1. Use the algorithm that we had in class to transform the following context-free grammar into Chomsky normal form: $S \rightarrow BB$, $S \rightarrow BSb$, $S \rightarrow bA$, $A \rightarrow aB$, $A \rightarrow \epsilon$, $B \rightarrow \epsilon$.

$S \rightarrow BB$  
$S \rightarrow BSb$  
$S \rightarrow bA$  
$A \rightarrow aB$  
$A \rightarrow \epsilon$  
$B \rightarrow \epsilon$

**STEP 1**

$S \rightarrow bSb$  
$S \rightarrow bA$  
$A \rightarrow aB$  
$A \rightarrow \epsilon$

**STEP 2**

$S \rightarrow bSb$  
$S \rightarrow bA$  
$A \rightarrow aB$  
$A \rightarrow \epsilon$

**STEP 3**

$S \rightarrow bSb$  
$S \rightarrow bA$  
$A \rightarrow aB$  
$A \rightarrow \epsilon$
2. Use the general algorithm to design a (non-deterministic) pushdown automaton that recognizes exactly the context-free language described in Problem 1.
3. In the context-free grammar described in Problem 1, we can derive the sequence bab as follows:

- first, we use the rule $S \rightarrow BSb$;
- then, we use the rules $B \rightarrow \varepsilon$ and $S \rightarrow bA$;
- after that, we use the rule $A \rightarrow ab$;
- 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 $uv^ixy^iz$ 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 $a^{3n}b^{2n}c^n$ cannot be recognized by a pushdown automaton.

$$L = \{a^{3n}b^{2n}c^n : n \geq 0, 1, 2, \ldots\}$$

**Proof** Assume $L$ is a context free grammar then by pumping lemma there exists a $p$ such that for $s \in L$ and $|s| \geq p$ there exist $UVX'YZ = s$ where $|VX'| > 0$, $|VX| \leq p$ and for $i = 0, 1, \ldots$ $UV^iX'Y^iZ \in L$.

Let $s = a^{3p}b^{2p}c^p$ then by pumping lemma $UVX'YZ = s$ since $|s| = 6p \geq p$.

Now take $UUVX'Y^iZ$ for $i = 2$ then $UU'VX'Y^iZ \notin L$.

- If $U$ is in $a$'s and we pump only $a$'s are added disrupting balance. The case is the same for $V$ in $b$'s or $c$'s.
- If $U$ is in $a$'s and $b$'s then only $a$'s and $b$'s will be added when we pump and balance will disrupted.
- If $U$ is in $b$'s and $c$'s then only $b$'s and $c$'s will be added when we pump and balance will be disrupted since $|VX| \leq p$ it cannot be in all three letter groups and since in all other cases balance was disrupted $UUVX'Y^iZ$ is not in $L$ which is contradiction.

So $L$ is not 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:

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

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

$S \rightarrow Q_1$
$Q_1 \rightarrow 1Q_2$
$Q_2 \rightarrow 1Q_2$
$Q_1 \rightarrow 0Q_1$
$Q_2 \rightarrow 0Q_1$
$Q_1 \rightarrow \epsilon$
6. Use the general algorithm to design a Turing machine that is equivalent to a finite automaton from Problem 5.

\[
\begin{align*}
(s, c, c+\#) &\rightarrow (Q_1, R) \\
(Q_1, 1) &\rightarrow (Q_2, R) \\
(Q_1, 0) &\rightarrow (Q_1, R) \\
(Q_2, 1) &\rightarrow (Q_2, R) \\
(Q_2, 0) &\rightarrow (Q_1, R) \\
(Q_2, \#) &\rightarrow \text{reset} \\
(Q_1, \#) &\rightarrow \text{accept}
\end{align*}
\]
7. Use the general stack-based algorithms to show:

- how the compiler will transform the expression $2 \times (3 \times 4 - 5)$ into inverse Polish notation, and
- how it will compute the value of this expression.
8-9. Design a Turing machine that computes the function $n + 2$: 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.
10. Computability and feasibility:

- Formulate Church-Turing thesis.
- Is it a mathematical theorem? A statement about the physical world?
- Formulate the current definition of a feasible algorithm.
- For extra credit: Explain why this definition is not perfect.

Church-Turing thesis

Anything that is computable on a physical device can be computed on a Turing machine.

Is a statement about a physical world.

Feasible algorithm

An algorithm is feasible if there exist a polynomial \( p(n) \) such that \( f_A(x) \leq p(\text{len}(x)) \).

Not a perfect definition because there are algorithms that are not feasible in practice such as \( f_A(n) = 10^{10^{10^n}} \) and there are algorithms that are not feasible by definition but are feasible in practice such as \( f_A(n) = e^{10^{10^n}} \).