1. Use the algorithm that we had in class to transform the following context-free grammar into Chomsky normal form. This grammar describes simple arithmetic expressions (E), combining numbers (N) 0 and 1 with + and -. The starting variable is E, the rules are: E --> N, E --> N+S, N --> 0, N --> 1, E --> N-E.

\[
\begin{align*}
E &\rightarrow N \\
E &\rightarrow N+E \\
N &\rightarrow 0 \\
N &\rightarrow 1 \\
E &\rightarrow N-E,
\end{align*}
\]

**Step 0**

**Step 1**

\[
\begin{align*}
E &\rightarrow 0 \\
E &\rightarrow 1 \\
S_0 &\rightarrow N+E \\
S_0 &\rightarrow N-E
\end{align*}
\]

**Step 2**

\[
\begin{align*}
V_0 &\rightarrow 0 \\
V_1 &\rightarrow 1 \\
V_0 &\rightarrow + \\
V_0 &\rightarrow - \\
V_0 &\rightarrow E \\
V_0 &\rightarrow N+E \\
V_0 &\rightarrow N-E \\
S_0 &\rightarrow 0 \\
S_0 &\rightarrow 1
\end{align*}
\]

Rules:

\[
\begin{align*}
N &\rightarrow 0 \\
N &\rightarrow 1 \\
E &\rightarrow 0 \\
E &\rightarrow 1 \\
E &\rightarrow N+E \\
E &\rightarrow N-E \\
V_0 &\rightarrow 0 \\
V_1 &\rightarrow 1 \\
V_0 &\rightarrow + \\
V_0 &\rightarrow - \\
V_0 &\rightarrow E \\
V_0 &\rightarrow N+E \\
V_0 &\rightarrow N-E \\
S_0 &\rightarrow 0 \\
S_0 &\rightarrow 1 \\
S_0 &\rightarrow V_{N-E} E \\
S_0 &\rightarrow V_{N-E} \\
S_0 &\rightarrow V_{N+E} E \\
S_0 &\rightarrow V_{N+E} \\
S_0 &\rightarrow V_{N-E} E \\
S_0 &\rightarrow V_{N-E} \\
V_0 &\rightarrow + \\
V_0 &\rightarrow -
\end{align*}
\]
2. Use the general algorithm to design a (non-deterministic) pushdown automaton that recognizes exactly the context-free language described in Problem 1. Show, step-by-step, how a word 1+0 will be accepted by this automaton.
3. Apply, to pushdown automaton that you designed in Problem 2, the general algorithm of transforming a pushdown automaton into a context-free grammar, and show how this new grammar will generate the word 1+1. (If you are running out of time, just a few steps of this algorithm will be OK.)
4. In the context-free grammar described in Problem 1, we can derive the sequence 0+1+1 as follows:

- first, we use the rule $E \rightarrow N+E$;
- then, we use the rules $N \rightarrow 0$ and $E \rightarrow N+E$;
- after that, we use the rules $N \rightarrow 1$ and $E \rightarrow N$;
- finally, we use the rule $N \rightarrow 1$.

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 $i = 2$, draw a tree explaining how the corresponding sequence $uv^ixy^iz$ can be derived in this grammar.
5. Use pumping lemma for pushdown automata to prove that the language consisting of all the words of the type $1^n0^n1^n0^n$ cannot be recognized by a pushdown automaton. Can you use the same lemma to prove that the language $1^n0^n$ cannot be generated by a context-free grammar?

Proof: Let's assume that $L$ can be recognized by a CFG.

Then, by pumping lemma, there exists a $p$ such that every word $s$ from $L$ whose length is at least $p$ can be represented as $s = uvxyz$, where $|uvy| < p$, $|uxyz| \leq p$, and for any $i$, $uv^ixy^iz \in L$.

Let's take $s = 1^p0^p1^p0^p \in L$, here $|s| = 4p \geq p$. So it can be represented as $uvxyz$ - by pumping lemma, let's consider all possible cases that hold $|vx| \leq p$.

We take $s = uvuvxyyz \in L$.

But,

- If $vx$ is in first $1$'s, first $0$'s, second $1$'s or second $0$'s, means that when we pump we will add more of that symbol, disrupting the balance.
- If $vx$ is in first $1$'s and first $0$'s, then if we pump we will have more first $1$'s and first $0$'s than second $1$'s and second $0$'s. Similarly, if $vx$ is in first $0$'s and second $1$'s and second $0$'s.

Any combination gives us a contradiction, since the word is not part of $L$. So, cannot be recognized by PDA. It is not a context-free.
$1^0 \cdot 0^n$

I think that $1^0 \cdot 0^n$ can be recognized by a CFG since $vxy$ can be between $1^3$ and $0^3$. 
6. Use the general algorithm to design a Turing machine that is equivalent to the following finite automaton. This automaton helps to check whether a person is a saint. The alphabet consists of two symbols: + ("good deed") and − ("bad deed"). A person is a saint if he or she performed at least one good deed and none bad deeds. The automaton has three states: b ("born", starting state), p ("perfect so far", final state), and i ("imperfect").

- From each state, − leads to the state i.
- The state i is a sink (once in i, we stay in i).
- From p, + stays in p.
- From b, + leads to p.

Show, step-by-step, how this Turing machine will reject a sequence consisting of a minus; and a plus as not corresponding to a saint.
7. 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.

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

\[ 2 \quad + \quad 23 \quad 234 \quad 2345 \quad 2345- \quad 2345+ \]

\[ 3 - 20 = -17 \]

\[ 2 \quad 3 \quad 4 \quad 5 \quad * \quad - \quad + \]

\[ 2 \quad 3 \quad 2 \quad 3 \quad 4 \quad 20 \quad -17 \quad -15 \quad 2 \quad 2 \quad 3 \quad 2 \quad 2 \quad \]

Result $= -15$
8-9. Design a Turing machine that computes the function \( n + 2 = n + 10_2 \) in binary. 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.

```
<table>
<thead>
<tr>
<th>Case 0</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1</td>
<td>1 1 1</td>
<td>1 1 0 1</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td></td>
<td>+ 1 0</td>
<td>+ 1 0</td>
<td>+ 1 0</td>
</tr>
<tr>
<td></td>
<td>1 0 0 0 1</td>
<td>1 0 0 0 0</td>
<td>1 1 1 0</td>
</tr>
</tbody>
</table>
```

we have

```
we want start
```

```
start, — → check0, R
check0, — → 0, add2, R
add2, — → 1, back, L
check0, 0 → add2, R
add2, 0 → 1, back, L
add2, 1 → 0, replace, R
replace, 1 → 0, replace, R
replace, 0 → 1, back, L
replace, 1 → 1, back, L
back, 0 → back, L
back, 1 → back, L
back, 1 → halt

file:///Q:/cs3350.16a/test2.html
10. Computability:

- Formulate Church-Turing thesis.
- Is it a mathematical theorem? A statement about the physical world?
- Formulate the halting problem. Is it computable?

1. Church–Turing Thesis

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

2. Church–Turing thesis is not a mathematical theorem. It is a statement about the physical world.

3. Halting Problem

Th. No algorithm is possible that given a problem P and data d, checks whether P halts on d.

No, it is not computable.