1. (a) Yes. [Requires counting modulo 2.]
(b) No. [Requires counting arbitrarily high.]
(c) Yes. [Yes. Simple CFG and PDA.]
(d) No.
(e) No.
(f) No.
(g) Yes.
(h) No.
(i) Yes. [The intersection of two regular languages is regular and testing whether a regular language (the intersection) is nonempty is decidable.]
(j) Yes. [Use the CYK parsing algorithm or simulate the behaviour of a PDA.]

2. (a) Apply the method described in Section 3.2.3. Simplification is permitted. Here is one possible (simplified) solution.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ q0</td>
<td>q1</td>
<td></td>
</tr>
<tr>
<td>q0</td>
<td>q5</td>
<td></td>
</tr>
<tr>
<td>q0</td>
<td></td>
<td>q6</td>
</tr>
<tr>
<td>q1</td>
<td>q2</td>
<td>q3</td>
</tr>
<tr>
<td>q2</td>
<td></td>
<td>q4</td>
</tr>
<tr>
<td>q3</td>
<td>q0</td>
<td></td>
</tr>
<tr>
<td>q4</td>
<td>q0</td>
<td></td>
</tr>
<tr>
<td>* q5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* q6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that the transition on each edge is a single symbol or $\epsilon$, but not a sequence of symbols.

(b) $G \rightarrow AC$
$A \rightarrow ab \mid aAb$
$C \rightarrow c \mid cC$

(c) $E \rightarrow E + E \Rightarrow E + b \Rightarrow E \times E + b \Rightarrow E \times b + b \Rightarrow a \times b + b$
(d) A grammar is ambiguous if some string in the language derived from the grammar has two distinct rightmost derivations (or, equivalently, two distinct parse trees). Here is a second rightmost derivation for the given string.

\[ E \rightarrow E \times E \Rightarrow E \times E + E \Rightarrow E \times E + b \Rightarrow E \times b + b \Rightarrow a \times b + b \]

3. (a) [This is the hardest question on the paper.]

The required language contains strings \(a, aa, aabbb, ababb, b, ba, \ldots\)

Here is the simplest PDA I know that accepts this language. [It happens to be deterministic.] Construct it by modifying the PDA that accepts \(L\). [But note that with PDAs you can’t just complement the accepting states to accept the complement of the language the way you can with DFAs.]

The automaton has 5 states (1 to 5), 2 stack symbols (\(A\) and \(Z\)), initial state 1, initial stack symbol \(Z\), and accepting states 2, 3 and 5. The transition function is defined as follows:

\[
\begin{align*}
\delta(1, a, Z) &= (2, Z) \\
\delta(1, b, Z) &= (5, Z) \\
\delta(2, a, Z) &= (2, AZ) \\
\delta(2, a, A) &= (2, AA) \\
\delta(2, b, A) &= (3, \epsilon) \\
\delta(2, b, Z) &= (4, Z) \\
\delta(3, b, A) &= (3, \epsilon) \\
\delta(3, b, Z) &= (4, Z) \\
\delta(4, a, Z) &= (5, Z) \\
\delta(4, b, Z) &= (5, Z) \\
\delta(5, a, Z) &= (5, Z) \\
\delta(5, b, Z) &= (5, Z)
\end{align*}
\]

[This looks simpler when drawn as a diagram.]

[Here, state 1 corresponds to the empty string, state 2 to a string in \(a^+\), state 3 to a string in \(\{a^mb^n \mid m > n > 0\}\), state 4 to a string in \(\{a^mb^n \mid m = n \geq 0\}\), and state 5 to any string in \(b\{a,b\}^+\) or \(\{a^mb^n \mid m = n \geq 1\}\{a,b\}^+.\) Note that in states 2 and 3 the stack contains one fewer \(A\)’s than the number of \(a\)’s that have been read.]

The automaton accepts by final state.

[Full marks were given for good attempts even if not completely correct.]

(b) [This question should have been easy. It merely requires you apply the general transformation method of Section 6.3.1. No problem solving is required.]

The required PDA has one state \(q\), which is (obviously) the initial state. Each variable and terminal of the grammar is a stack symbol. The initial stack symbol is \(S\). The PDA accepts by empty stack. The transition function is defined as follows:

\[
\begin{align*}
\delta(q, a, a) &= \{(q, \epsilon)\} \\
\delta(q, i, i) &= \{(q, \epsilon)\} \\
\delta(q, e, e) &= \{(q, \epsilon)\} \\
\delta(q, \epsilon, S) &= \{(q, a), (q, iS), (q, iSeS)\}
\end{align*}
\]
4. [Every proof must have the following components.]

- Suppose that $L$ is context-free.
- Let $n > 0$ be the pumping lemma constant.
- Choose $z = a^n b^n c^n d^n$. [This is the first creative step. Other similar choices also work.]
- By the pumping lemma [for CFLs], $z$ can be written in the form $z = uvwxy$, with $|vwx| \leq n$ and $|vx| > 0$.
- [Now for an analysis of $z$ and $vwx$ that leads to a contradiction. In this example, the analysis is a bit repetitive. This is the second creative step.] As $|vwx| \leq n$, $vwx$ cannot contain both $a$ and $c$ or both $b$ and $d$. Suppose $vwx$ contains $a$ but not $c$. By the pumping lemma, $uv^2wx^2y \in L$. But $v$ contains at least one $a$. So $uv^2wx^2y$ contains more $a$’s than $c$’s. Hence, $uv^2wx^2y \notin L$. This is a contradiction.

The other three cases ($c$ but not $a$, $b$ but not $d$, $d$ but not $b$) lead to contradictions similarly. [Not every similar case needs to be presented, when they are as similar as they are here.]

- Since every possible form of $vwz$ leads to a contradiction, the original assumption that $L$ is context-free must be false.