Problem 4.11 (6 pts)
Let $INFINITE_{PDA} = \{(M) \mid M$ is a PDA and $L(M)$ is an infinite language}. Show $INFINITE_{PDA}$ is decidable.

We must construct a TM $A$ that decides $INFINITE_{PDA}$.
$A = \text{On input } \langle M \rangle \text{ where } M \text{ is a PDA:}$
1. Construct an equivalent CFG $G$ from $M$.
2. Remove a variable $X$ (and the productions containing $X$) from $G$ if it is not the case that $S \Rightarrow^* uXv \Rightarrow^* w$, where $w$ is in $L(G)$.
3. Accept, $\langle M \rangle$ if there exists a derivation of $X \Rightarrow^+ uXv$ for some $uv \in \Sigma^+$. Otherwise, reject $\langle M \rangle$.

Since all steps in this machine will stop, $INFINITE_{PDA}$ is decidable.

Problem 4.13 (6 pts)
Let $A = \{(R,S) \mid R$ and $S$ are regular expressions and $L(R) \subseteq L(S)\}$. Show $A$ is decidable.

We must construct a TM $M$ that decides $A$.
$M = \text{On input } \langle R,S \rangle \text{ where } R$ and $S$ are regular expressions:
1. Construct a DFA $X$ such that $L(X) = L(S) \cap L(R)$
2. Run $E_{DFA}$ on input $\langle X \rangle$. Accept if it accepts and reject if it rejects.

Since $E_{DFA}$ is decidable, $M$ is decidable. This machine works because
$L(R) \subseteq L(S) \iff L(S) \cap L(R) = \emptyset$.

Problem 4.17 (6 pts)
Prove that $EQ_{DFA}$ is decidable by testing the two DFAs on all strings up to a certain size. Calculate a size that works.

Let $EQ_{DFA} = \{(A,B) \mid A, B \text{ are DFAs and } L(A) = L(B)\}$. Let $M$ be a TM that decides $EQ_{DFA}$.
$M = \text{On input } \langle A,B \rangle$:  
1. Check that input $\langle A,B \rangle$ includes 2 DFAs $A$ and $B$ with the same alphabet. If not, reject.
2. Calculate $n = |Q_A|$ and $m = |Q_B|$ (the numbers of states in the two DFAs).
3. Enumerate all strings in $\Sigma$ up to length $n \times m$, and for each such string $w$:
   a. Run $A$ on $w$.
   b. Run $B$ on $w$.
   c. If the result of the two executions is different, reject. Otherwise, continue.
4. If all length up to $n \times m$ strings explored, accept.

The reason we can check only the length up to $n \times m$ strings is because if the two DFAs do not accept the same language, then there must be a string $w$ of size $|w| \leq n \times m$ for which $A(w) \neq B(w)$.

Assume by contradiction that the shortest string that yields a different output of $A$ and $B$ is $w'$ and $|w'| = l > nm$, then there is a sequence of states $a_0, a_1, \ldots, a_l \in Q_A$ and $b_0, b_1, \ldots, b_l \in Q_B$ that describe the transitions for $w'$ in $A$ and $B$ respectively. Since $l > nm$, putting those sequences side by side, there must be some repetition of a pair of sequences $a_i, b_i$ and $a_j, b_j$ such that $a_i = a_j, b_i = b_j, i < j$.

Therefore, we can remove all subsequences in between leaving only $a_i, b_i$ and by that get a smaller string $w''$ that $A, B$ will act the same over exactly as over $w'$, thus contradicting the assumption that $w'$ is the shortest such that $A(w') 
eq B(w')$. Hence, checking all strings up to size $nm$ is sufficient.
Problem 4.21 (6 pts)
Let \( S = \{ \langle M \rangle \mid M \text{ is a DFA that accepts } \omega^R \text{ whenever it accepts } \omega \} \). Show that \( S \) is decidable.

We must construct a TM \( M \) that decides \( S \).

\( M = \) “On input \( \langle M \rangle \) where \( M \) is a DFA:

1. Construct DFA \( M^R \) that recognizes \( \{ \omega \mid \omega^R \in L(M) \} \) (see #)
2. Run \( EQ_{DFA} \) on the input \( \langle M, M^R \rangle \). If it accepts, then accept. Otherwise, reject.

This machine works by accepting only DFAs which accept a word and its reverse.

(#) We can construct the DFA \( M^R \) by constructing an NFA from \( M \) by reversing all transitions, making the previous start state the only accept state and adding a new start state with \( \epsilon \) transitions to all previous accepting states. Then a DFA can be constructed from that NFA.

Problem 4.28 (6 pts)
Let \( C = \{ \langle G, x \rangle \mid G \text{ is a CFG, } x \text{ is a substring of some } y \in L(G) \} \). Show that \( C \) is decidable.

We must construct a TM \( M \) that decides \( C \).

\( M = \) “On input \( \langle G, x \rangle \) where \( G \) is a CFG and \( x \) is a string:

1. Construct DFA \( A \) that recognizes the language of the regular expression \( \Sigma^* \cup \{x\} \cup \Sigma^* \)
2. Construct CFG \( F \) for the context-free language \( L(G) \cap L(A) \)
3. Run the TM \( EQ_{CFG} \) on \( L(F) \). If it accepts, reject. Otherwise, accept.”

This machine will work because we know that the intersection of a CFL and a regular language is a CFL; therefore, \( L(F) \) will be a CFL. Additionally, \( L(A) \) is the language of all strings with \( x \) as their substring and is a regular language. Hence, if \( G \) generates some string \( w \) with \( x \) as its substring, the intersection, \( L(F) \), should be nonempty.