Simrun Heir and Michael Sanchez
I pledge my honor that I have abided by the Stevens Honor System.
Problem 1: Show that the following language is decidable by giving a high-level description of a TM that decides the language. {<M >: *M is a PDA $L(M)$ is an infinite language* }
Notes:

- Thm 4.7: A_{CFG} is a decidable language
  - Proven by decider S (from textbook)
  - S = "On input ⟨G, w⟩, where G is a CFG and w is a string:
    - 1. Convert G to an equivalent grammar in Chomsky normal form.
    - 2. List all derivations with 2n−1 steps, where n is the length of w; except if n = 0, then instead list all derivations with one step.
    - 3. If any of these derivations generate w, accept; if not, reject."
- Thm 4.8: E_{CFG} is a decidable language
  - Proven by decider R (from textbook
  - R = "On input ⟨G⟩, where G is a CFG:
    - 1. Mark all terminal symbols in G.
    - 2. Repeat until no new variables get marked:
    - 3. Mark any variable A where G has a rule A → U1U2 ···Uk and each symbol U1,...,Uk has already been marked.
    - 4. If the start variable is not marked, accept; otherwise, reject."
- Thm 4.9: Every context-free language is decidable
- Thm 2.20: A language is context free if and only if some pushdown automaton recognizes it.
- Theorem 2.9: Any context-free language is generated by a context-free grammar in Chomsky normal form.
- Thm 2.34: Pumping Lemma for context-free languages
- When describing strings produced by a grammar, if there is an obvious enough pattern it can be described with a regular expression

Answer:
The following TM I decides the language described above
I = "On input <A>, where A is a PDA:

1. Convert A into a CFG G
2. Compute G's pumping length, p
3. Construct a regular expression D that contains all strings of length p or more
4. Construct a CFG M such that L(M) = L(A)∩L(D)
5. Test L(M) = ϕ using the decider R from Thm 4.8
   a. If R halts and accepts, REJECT,
   b. If R halts and rejects, ACCEPT."

I is a decider that tests to see if the intersection of the language of the PDA A and the language of the strings generated by the CFG A are the same, if they are then the PDA is accepted, if not it is rejected.

Problem 2: Let *G* be a context-free grammar that generates strings over the alphabet $\Sigma = \{a, b\}$. Show that the problem of determining if *G* generates a string in $a*$ is decidable. In other words, show that the following language is decidable:
$\{<G>: G$ is a CFG over $\{a,b\}$ and $a* \cap L(G) \neq \phi\}$
Notes:
- CFLs are closed under intersection
- CFLS are closed under union, concatenation, and star operations
- Thm 4.7: A_{CFG} is a decidable language
- Thm 4.8: E_{CFG} is a decidable language

Answer:
Construct a Turing Machine T that decides the language of G.
T = "On input <G>:
1. Construct a CFG H such that L(H) = $a* \cap L(G)$
2. Using the decider R from E_{CFG} test L(H) = ϕ
3. If R halts and accepts, REJECT; if R halts and rejects, ACCEPT."

Problem 3: Let A be a TM-recognizable language of strings that encode TMs that are deciders. Prove that there is a decidable language which is not decided by any TM in A. (Hint: start with an enumerator for A.)
Notes:
- Regular ⊂ Context-free ⊂ Decidable ⊂ Turing-recognizable
- Thm 4.11: A_{TM} is undecidable
    - It is turing-recognizable but not decidable
        - TM U for recognizing A_{TM}
            - U = "On input ⟨M,w⟩, where M is a TM and w is a string:
                - 1. Simulate M on input w.
                - 2. If M ever enters its accept state, accept; if M ever enters its reject state, reject."
- Thm 3.16: Every nondeterministic Turing machine has an equivalent deterministic Turing machine
    - Corollary 3.18: A language is Turing-recognizable if and only if some nondeterministic Turing machine recognizes it
    - Corollary 3.19: A language is decidable if and only if some nondeterministic Turing machine decides it.
- Thm 3.21: A language is Turing-recognizable if and only if some enumerator enumerates it
- Thm: If S is any infinite set then no function from s to P(S), the power set of S, is bijective, and therefore, |S| < |P(S)|
- Use diagonalization to show TM constructed isn't bijective
    - Domain, all TMs and range is all decidable languages
    - Show that there is an extra language
- Print out language strings for language of TM then mark off as a pair

    - Go down until all of domain is accounted for, if there is a language in the range that is not pointed to by an element in the domain reject
    - Use this general idea but with diagonalization
        - Doing something to every turing machine
- Make a new language not decidable in by any TM in a it must differ from every TM in A at at least one spot
    - Diagonalization for keeping of track of all the different TMs
    - Way to construct a new language that frankenstein's all the TMs in A, this language won't be in A

Answer:
Let TM D be the TM that decides if there is a decidable language that is not decided by any TM encoded by A. It takes A as input.
D = "On input A

1. Split A into components <A1>, <A2>,... ,<An> such that each string in An for all n>0 encodes a specific decider for that language.
    a. Ex: A = {<A1>, <A2>,... ,<An>}
2. Iterate through A and using an enumerator E print out each An for all n>0.
    a. E prints out an An, and continue until all n pieces of A are printed
3. Once An is printed, construct a decider B encoded as follows
    a. It's description differs from A1 in some way
    b. Inductively define the ith part of Bs encoding to differ from the encoding of Ai
        i. Note: this makes Bs encoding a combination of all the encodings comprising A, but differ at each value of i.
4. Run An for n>0 on B
    a. If any An halts and accepts, REJECT
    b. If all An in A have rejected, ACCEPT"

Problem 4: Consider the problem of determining whether a TM M on input w ever attempts to move its head left when its head is on the leftmost tape cell.
a) Formulate this problem as a decision problem for a language, and
b) Show that the language is undecidable.
Notes:

- Thm 4.11: A_{TM} is undecidable
- Thm 5.1: $HALT_{TM}$ is undecidable
- Reduce Atm or HALTtm to this problem
    - Show if you have Atm to solve this language then you can use this language to solve this language
- Find some similarity between this problem and Atm and since Atm is undecidable this language is undecidable
- Say that there is a TM for this language and run it on the language that is being reduced, and find the contradiction

Answer:

a) $LEFT_{TM}$ = {<M ,*w*>: M *is a Turing Machine and w is a string that at some point causes M to try to move left off of the leftmost cell*}

b) Proof: Show that $A_{TM} \leq LEFT_{TM}$

Assume that $LEFT_{TM}$ is decidable and let K be a TM to decide it.

Use K to construct a TM S that decides A_{TM} .

S = "On input <*M,w*>

1. Construct a TM M1
    a. M1 = "On input x
        i. If x ≠ w, REJECT
        ii. Run M(w)
            1. If M accepts, REJECT
            2. Otherwise, ACCEPT
2. Run K(M1)
    a. If K(M1) accepts, then REJECT
    b. If K(M1) rejects then, then ACCEPT"

This, however, creates a contradiction.

This is because A_{TM} is provably undecidable, so no decider can exist for it.

This implies, since $A_{TM} \leq LEFT_{TM}$, $LEFT_{TM}$ is also undecidable.