1. If $A$ is a language, then $SUFFIX(A)$ is the language 
\{
u | vu \in A \text{ for some string } v\}\)

(a) Prove that if $A$ is decidable, then $SUFFIX(A)$ is Turing recognizable.

(b) Prove that if $A$ is Turing recognizable, then $SUFFIX(A)$ is Turing recognizable.

[Since every decidable language is Turing recognizable, this part implies the first part, but since the proof is harder, I made it a separate part.]

Solution:

a) Let $A$ be a language over an alphabet $\Sigma$ and let $M$ be a Turing machine that decides $A$. Since $\Sigma^*$ is countably infinite, we may list the strings in $\Sigma^*$ as $s_1, s_2, s_3, \ldots$. A Turing machine $N$ that recognizes $SUFFIX(A)$ is given by

\[
N = \text{"On input } w \\
1. \text{ For } i = 1, 2, 3, \ldots \\
2. \text{ Run } M \text{ on } s_iw. \text{ If } M \text{ accepts, accept. If } M \text{ rejects, next } i." 
\]

b) Now assume that $M$ only recognizes $A$. Then, $M$ may go into an infinite loop on some inputs, and we have to modify our definition of $N$ to take account of this fact. Instead of just running $M$ on $s_iw$ and waiting for $M$ to halt, which may not happen, $N$ runs $M$ in parallel on several inputs, for a fixed number of steps.

\[
N = \text{"On input } w \\
1. \text{ For } i = 1, 2, 3, \ldots \\
2. \text{ Run } M \text{ on } s_1w, s_2w, \ldots, s_iw \text{ for } i \text{ steps each. If } M \text{ accepts any of the strings in } i \text{ steps accept, else, next } i." 
\]

2. Apply the method from class that decides $E_{DF\tilde{A}}$ to the following DFA and answer the questions below.
(a) List the states you mark in the order they get marked.
   \( p, q, r, u \)

(b) Does the DFA belong to \( E_{DFA} \)? Yes

(c) How does your answer to (b) follow from your answer to (a)?
   No accept state is marked.

3. Apply the method from class that decides \( E_{CFG} \) to the following CFG and answer the questions below.

\[
\begin{align*}
S & \rightarrow aTbU | aSTb \\
T & \rightarrow YaU | bT \\
U & \rightarrow aYbY | VW \\
V & \rightarrow aV | bW \\
W & \rightarrow aW | bV \\
X & \rightarrow bX | \varepsilon \\
Y & \rightarrow aY | aTU
\end{align*}
\]
(a) List the terminals and variables you mark in the order they get marked. (List each terminal and variable only the first time you mark it. There is more than one possible order.)
\[ a, b | X, Y | U | T | S \]

(b) Does the CFG belong to $E_{CFG}$? No

(c) How does your answer to (b) follow from your answer to (a)?

S is marked

4. The language $EQ_{REX}$ is defined as $\{\langle R, S \rangle | R, S \text{ are regular expressions and } L(R) = L(S)\}$. Prove that $EQ_{REX}$ is decidable.

Solution:
A Turing machine $M$ that decides the language $EQ_{REX}$ is given by

\[ M = \text{"On input } \langle R, S \rangle \text{ where } R \text{ and } S \text{ are regular expressions,} \]

1. Combining the methods of Lemma 1.55 and Theorem 1.39, produce DFAs $B$ and $C$ with $L(B) = L(R)$ and $L(C) = L(S)$.
2. Run the TM $F$ that decides $EQ_{DFA}$ on $\langle B, C \rangle$.
3. If $F$ accepts, then accept. If $F$ rejects, then reject.”

5. Let $ALL_{NFA} = \{\langle A \rangle | A \text{ is an NFA and } L(A) = \Sigma^*\}$. Show that $ALL_{NFA}$ is decidable.

Solution:
$ALL_{NFA}$ is decided by the following Turing machine $M$.

\[ M = \text{"On input } \langle A \rangle \text{ where } A \text{ is an NFA,} \]

1. Using the method of Theorem 1.39, construct a DFA $B$ that is equivalent to $A$.
2. Obtain a DFA $C$ from $B$ by reversing accept and reject states.
3. Run the TM $T$ that decides $E_{DFA}$ on $\langle C \rangle$.
4. If $T$ accepts, then accept. If $T$ rejects, then reject.”

Note that you have to transform $A$ into a DFA because the complementation construction does not always work for NFAs.

6. Let $F = \{\langle A \rangle | A \text{ is a DFA with input alphabet } \{0, 1\} \text{ and every string in } L(A) \text{ has an even number of } 0\text{'s}\}$. Prove that $F$ is decidable.

Solution: The set of all binary strings with an odd number of 0’s is a regular language, so there is a DFA $B$ that recognizes this language. Then, $\langle A \rangle$ is in $F$ if and only if $L(A) \cap L(B) = \emptyset$. Thus, we can define a Turing machine $M$ that decides the language $F$ by

\[ M = \text{"On input } \langle A \rangle \text{ where } A \text{ is a DFA with tape alphabet } \{0, 1\}, \]

1. Produce a DFA $C$ with $L(C) = L(A) \cap L(B)$, where $B$ is a DFA that recognizes the set of binary strings with an odd number of 0’s.
2. Run the TM $T$ that decides $E_{DFA}$ on $\langle C \rangle$.
4. If $T$ accepts, then accept. If $T$ rejects, then reject.”

7. Let $K = \{ \langle A \rangle | A$ is a DFA and $L(A)$ does not contain any string with at least as many $a$’s as $b$’s $\}$ Show that $K$ is decidable. (The solution to Problem 4.25 [4.23] is useful here.)

Solution: The language of all strings in $\{a, b\}^*$ with at least as many $a$’s as $b$’s is context-free. (You were given a context-free grammar for this language in the solutions to Homework 5.) Let $P$ be a PDA that recognizes this language.

A Turing machine $T$ that decides $K$ is given by:

$T=”$On input $\langle A \rangle$ where $A$ is a DFA:

1. Using the method of Problem 2.18a, construct a PDA $S$ that recognizes $L(A) \cap L(P)$, where $P$ is the PDA mentioned above.
2. Convert $S$ into a CFG $G$.
3. Run the TM $R$ that decides $E_{CFG}$ on $\langle G \rangle$.
4. If $R$ accepts, accept. If $R$ rejects, reject.”

8. Let $L = \{ \langle P \rangle | P$ is a PDA and $L(P)$ does not contain any even length string $\}$. Prove that $L$ is decidable.

Solution: For any alphabet, the set of even length strings over the alphabet is a regular language.

A Turing machine $T$ that decides $L$ is given by:

$T=”$On input $\langle P \rangle$ where $P$ is a PDA:

1. Let $E$ be a DFA that recognizes the set of even length strings over the alphabet of $G$.
2. Using the method of Problem 2.18a, construct a PDA $S$ that recognizes $L(E) \cap L(P)$.
3. Convert $S$ into a CFG $H$.
4. Run the TM $R$ that decides $E_{CFG}$ on $\langle H \rangle$.
4. If $R$ accepts, accept. If $R$ rejects, reject.”

9. The set of all finite sequences of natural numbers is countable.

Proof: Unlike the case of $\Sigma^*$ where $\Sigma$ is an alphabet, it is not possible to list all the finite sequences of natural numbers in order by their length, because there are infinitely many sequences of natural numbers with any given length other than 0, so this list would never get to any sequences of length 2, much less all the finite sequences.

One method that does work is to list the finite sequences of natural numbers in order according to the sum of their entries. (Because we do not
consider 0 to be a natural number, there are only finitely many finite sequences of natural numbers whose sum is any given number.) For a given sum, the sequences can be listed in lexicographic (ie, dictionary) order.

The first sequence listed would be \( \varepsilon \), then \( \langle 1 \rangle \). Next would come the two sequences whose sum is 2: \( \langle 1, 1 \rangle \) and \( \langle 2 \rangle \). The sequences whose sum is 3 come next: \( \langle 1, 1, 1 \rangle \), \( \langle 1, 2 \rangle \), \( \langle 2, 1 \rangle \) and \( \langle 3 \rangle \). Next come the sequences with sum 4, and so on.

10. An infinite sequence \( a_1a_2\cdots \) of natural numbers is called *strictly increasing* if \( a_1 < a_2 < a_3 < \cdots \). Let \( B \) be the set of all strictly increasing sequences of natural numbers. Use diagonalization to prove that \( B \) is uncountable.

**Solution:** Let \( f : \mathbb{N} \to B \) be any function. We will prove that there is an element \( d \) of \( B \) that is not in the range of \( f \), i.e., we will define \( d \) to be different from all of \( f(1), f(2), \ldots \). We will write \( f(i)_j \) for the \( j \)th entry in \( f(i) \). We define \( d_1 = f(1)_1 + 1 \) and for \( n \geq 1 \), \( d_{n+1} = f(n + 1)_{n+1} + d_n \).

Then, for all \( n \geq 1 \), \( d_{n+1} = f(n + 1)_{n+1} + d_n \geq 1 + d_n > d_n \), so \( d \) is strictly increasing, and \( d \) is different from all the \( f(n) \)'s because \( d_n \neq f(n)_n \), so \( f \) is not onto.