1. (a) False, e.g., $A_{TM}$ is not Turing-recognizable.
(b) False, e.g., if $A = \{00, 11, 111\}$ and $B = \{00, 11\}$, then $A \cap B = \emptyset$, but $A \neq B$.
For $A$ and $B$ to be equal, we instead need $(A \cap B) \cup (A \cap \overline{B}) = \emptyset$.
(c) False. A TM $M$ may loop on input $w$.
(d) True, by Theorem 4.9.
(e) True, by slide 4-38.
(f) False, by Theorem 4.8.
(g) False, by Theorem 4.11.
(h) True, by Theorem 4.5.
(i) False, by Homework 9, problem 1.
(j) False, by Corollary 4.23.

2. (a) No, because $f(x) = f(z) = 2$.
(b) Yes, because $f(y) = 1$ and $f(x) = 2$, so all members of $B$ are hit by $f$.
(c) No, because $f$ is not one-to-one.
(d) An algorithm is a Turing machine that always halts.
(e) A language $L_1$ that is Turing-recognizable has a Turing machine $M_1$ such that $M_1$ accepts each $w \in L_1$, and $M_1$ loops or rejects every $w \notin L_1$. A language $L_2$ that is Turing-decidable has a Turing machine $M_2$ such that $M_2$ accepts each $w \in L_2$, and $M_2$ rejects every $w \notin L_2$; i.e., $M_2$ never loops.
It is important to note that Turing-recognizable and Turing-decidable are properties of languages and not Turing machines.

3. (a) $q_1 10\#0 1 x q_3 10 \#0 1 x 1 q_3 0 \#0 1 x 10 q_3 \#0 1 x 10 \#0 q_{reject} 1$
(b) $q_1 0 \#0 x q_2 \#0 x \# q_4 0 x q_6 \# x q_7 x \# x q_1 x x \# q_8 x x \# x q_8$
$\quad x \# x \downarrow q_{accept}$

4. [This is from slides 4-39 and 4-40.] Let $\mathcal{L}$ be the set of all languages over an alphabet $\Sigma$. Let $\mathcal{B}$ be the set of all infinite binary sequences, and we know that $\mathcal{B}$ is uncountable from class (this can be shown by using a diagonalization argument).
We will construct a mapping $\chi : \mathcal{L} \to \mathcal{B}$ such that $\chi$ is a correspondence, which will establish that $\mathcal{L}$ and $\mathcal{B}$ are of the same size. Then since $\mathcal{B}$ is uncountable, we will have that $\mathcal{L}$ is also uncountable.
We now describe how to construct the mapping $\chi$. First let $s_1, s_2, s_3, \ldots$ be a lexicographic listing of the strings in $\Sigma^*$. For any language $A \subseteq \Sigma^*$, define $\chi(A) = b_1 b_2 b_3 \cdots$, where $b_i = 1$ if $s_i \in A$, and $b_i = 0$ if $s_i \notin A$. Thus, the $i$th bit in the infinite binary sequence $\chi(A)$ is 1 if and only if the language $A$ contains the $i$th
string \( s_i \). We call \( \chi(A) \) the characteristic sequence of the language \( A \). For example, if \( \Sigma = \{0, 1\} \) and \( A = \{0, 00, 01, 000, \ldots\} \), then
\[
\begin{align*}
\Sigma^* &= \{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots \} \\
A &= \{ 0, 00, 01, 000, \ldots \} \\
\chi(A) &= 0 1 0 1 1 0 0 1 \ldots
\end{align*}
\]

Now we show that \( \chi : \mathcal{L} \rightarrow \mathcal{B} \) is a correspondence.

- To show that \( \chi \) is one-to-one, note that if languages \( A_1 \) and \( A_2 \) such that \( A_1 \neq A_2 \), then they differ in at least one string \( s_i \); i.e., one of the languages includes \( s_i \) and the other does not. Then \( \chi(A_1) \) and \( \chi(A_2) \) differ in the \( i \)th bit, so \( \chi(A_1) \neq \chi(A_2) \). Hence, \( A_1 \neq A_2 \) implies \( \chi(A_1) \neq \chi(A_2) \), so \( \chi \) is one-to-one.

- To show that \( \chi \) is onto, note that given any infinite binary sequence \( b_1b_2b_3\ldots \in \mathcal{B} \), the language \( A \) defined such that it includes string \( s_i \) if and only if \( b_i = 1 \) has \( \chi(A) = b_1b_2b_3\ldots \). Thus, for every element \( b \in \mathcal{B} \), there is an element in \( \mathcal{L} \) that \( \chi \) maps to \( b \). Hence, \( \chi \) is onto.

Since \( \chi \) is one-to-one and onto, it is a correspondence.

Hence, \( \mathcal{L} \) and \( \mathcal{B} \) are of the same size. Since we know that \( \mathcal{B} \) is uncountable, that must mean that \( \mathcal{L} \) is also uncountable.

5. Define the language as
\[
C = \{ \langle D, R \rangle \mid D \text{ is a DFA and } R \text{ is a regular expression with } L(D) = L(R) \}.
\]

Recall that the proof of Theorem 4.5 defines a Turing machine \( F \) that decides the language \( EQ_{\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \). Then the following Turing machine \( T \) decides \( C \):

\[
T = \text{ "On input } \langle D, R \rangle \text{, where } D \text{ is a DFA and } R \text{ is a regular expression:}
\]

1. Convert \( R \) into an equivalent DFA \( D' \)
   using the algorithm in the proof of Kleene’s Theorem.

2. Run TM \( F \) for \( EQ_{\text{DFA}} \) on input \( \langle D, D' \rangle \).

3. If \( F \) accepts, accept. If \( F \) rejects, reject."

6. This is Homework 8, problem 4. We need to show there is a Turing machine that recognizes \( \overline{E_{\text{TM}}} \), the complement of \( E_{\text{TM}} \). Let \( s_1, s_2, s_3, \ldots \) be a list of all strings in \( \Sigma^* \). For a given Turing machine \( M \), we want to determine if any of the strings \( s_1, s_2, s_3, \ldots \) is accepted by \( M \). If \( M \) accepts at least one string \( s_i \), then \( L(M) \neq \emptyset \), so \( \langle M \rangle \in \overline{E_{\text{TM}}} \); if \( M \) accepts none of the strings, then \( L(M) = \emptyset \), so \( \langle M \rangle \notin \overline{E_{\text{TM}}} \). However, we cannot just run \( M \) sequentially on the strings \( s_1, s_2, s_3, \ldots \). For example, suppose \( M \) accepts \( s_2 \) but loops on \( s_1 \). Since \( M \) accepts \( s_2 \), we have that \( \langle M \rangle \in \overline{E_{\text{TM}}} \). But if we run \( M \) sequentially on \( s_1, s_2, s_3, \ldots \), we never get past
the first string. The following Turing machine avoids this problem and recognizes $E_{\text{TM}}$:

$$R = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a Turing machine:}$$

1. Repeat the following for $i = 1, 2, 3, \ldots$
2. Run $M$ for $i$ steps on each input $s_1, s_2, \ldots, s_i$.
3. If any computation accepts, accept.