1. (a) True, since every regular language is context-free, every context-free language is decidable, and every decidable language is Turing-recognizable.

(b) False, by Theorem 4.11.

(c) False, by Corollary 4.23.

(d) False. Can decide this by the following TM:

\[ M = \text{"On input } \langle N, R \rangle \text{, where } N \text{ is an NFA and } R \text{ is a regular expression:} \]

1. Check if \( \langle N, R \rangle \) is a proper encoding of NFA \( N \) and regular expression \( R \); if not, reject.
2. Convert \( N \) into equivalent DFA \( D_1 \) using algorithm in Theorem 1.39.
3. Convert \( R \) into equivalent DFA \( D_2 \) using algorithms in Lemma 1.55 and Theorem 1.39.
4. Run TM \( S \) for \( EQ_{DFA} \) on input \( \langle D_1, D_2 \rangle \). If \( S \) accepts, then accept; else, reject."

(e) True, by Theorem 4.5.

(f) False, by Theorem 3.13.

(g) False, by Corollary 3.15.

(h) False, e.g., the set of positive integers is infinite and countable.

(i) 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 \( (A \cap B) \cup (A \cap \overline{B}) = \emptyset \).

(j) False. TM \( M \) may loop on input \( w \).

2. (a) No, because \( f(x) = f(y) = 1 \).

(b) No, because nothing in \( A \) maps to \( 3 \in B \).

(c) No, because \( f \) is not one-to-one nor onto.

(d) A language \( L_1 \) that is Turing-recognizable has a Turing machine \( M_1 \) that may loop forever on a string \( w \notin L_1 \). A language \( L_2 \) that is Turing-decidable has a Turing machine \( M_2 \) that always halts.

(e) An algorithm is a Turing machine that always halts.

3. (a) \( q_1010\#1 \ xq_210\#1 \ x1q_20\#1 \ x10q_2\#1 \ x10\#q_41 \ x10\#q_{\text{reject}} \)

(b) \( q_11\#1 \ xq_3\#1 \ x\#q_51 \ xq_6\#x \ xq_7\#x \ xq_1\#x \ x\#q_8x \ x\#xq_8 \)

\( x\#x \uplus q_{\text{accept}} \)

4. This is Theorem 4.22. First we show that if \( A \) is decidable then it is both Turing-recognizable and co-Turing recognizable. Suppose that \( A \) is decidable. Then it must also be Turing-recognizable. Also, since \( A \) is decidable, there is a TM \( M \) that decides \( A \). Now define another TM \( M' \) to be the same as \( M \) except that we swap the accept and reject states. Then \( M' \) decides \( \overline{A} \), so \( \overline{A} \) is decidable. Hence, \( \overline{A} \)
is also Turing-recognizable, so \( A \) is co-Turing-recognizable. Thus, we proved that \( A \) is both Turing-recognizable and co-Turing-recognizable.

Now we prove the converse: if \( A \) is both Turing-recognizable and co-Turing-recognizable, then \( A \) is decidable. Since \( A \) is Turing-recognizable, there is a TM \( M \) with \( L(M) = A \). Since \( A \) is co-Turing-recognizable, \( \overline{A} \) is Turing-recognizable, so there is a TM \( M' \) with \( L(M') = \overline{A} \). Any string \( w \in \Sigma^* \) is either in \( A \) or \( \overline{A} \) but not both, so either \( M \) or \( M' \) (but not both) must accept \( w \). Now build another TM \( D \) as follows:

\[
D = \text{"On input string } w:\n1. \text{ Run } M \text{ and } M' \text{ alternatively on } w \text{ step by step.}
2. \text{ If } M \text{ accepts } w, \text{ accept. If } M' \text{ accepts } w, \text{ reject.}\n\]

Then \( D \) decides \( A \), so \( A \) is decidable.

5. Define the language as

\[
E_{\text{NFA}} = \{ \langle N \rangle \mid N \text{ is an NFA with } L(N) = \emptyset \}.
\]

Recall that the proof of Theorem 4.4 defines a Turing machine \( R \) that decides the language \( E_{\text{DFA}} = \{ \langle B \rangle \mid B \text{ is a DFA with } L(B) = \emptyset \} \). Then the following Turing machine \( S \) decides \( E_{\text{NFA}} \):

\[
S = \text{"On input } \langle N \rangle \text{, where } N \text{ is an NFA:}\n1. \text{ Convert } N \text{ into an equivalent DFA } D
   \text{ using the algorithm in the proof of Kleene’s Theorem.}
2. \text{ Run TM } R \text{ for } E_{\text{DFA}} \text{ on input } \langle D \rangle.
3. \text{ If } R \text{ accepts, accept. If } R \text{ rejects, reject.}"
\]

6. This is Theorem 5.4. Recall that \( E_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is a TM with } L(M) = \emptyset \} \), which we know is undecidable by Theorem 5.2. We can reduce \( E_{\text{TM}} \) to \( E_{\text{EQ}}_{\text{TM}} \) as follows. Suppose that \( E_{\text{EQ}}_{\text{TM}} \) is decidable by a TM \( R \). Then we could decide \( E_{\text{TM}} \) using the following TM \( S \) with \( R \) as a subroutine:

\[
S = \text{"On input } \langle M \rangle \text{, where } M \text{ is a TM:}\n1. \text{ Run } R \text{ on input } \langle M, M_{\emptyset} \rangle,
   \text{ where } M_{\emptyset} \text{ is a TM such that } L(M_{\emptyset}) = \emptyset.
2. \text{ If } R \text{ accepts, accept; if } R \text{ rejects, reject.}\n\]

The TM \( S \) just checks if the inputted TM \( M \) is equivalent to the empty TM \( M_{\emptyset} \), so \( S \) decides \( E_{\text{TM}} \). But \( E_{\text{TM}} \) is undecidable, so that must mean the decider \( R \) for \( E_{\text{EQ}}_{\text{TM}} \) cannot exist, so \( E_{\text{EQ}}_{\text{TM}} \) is undecidable.
A mistake that some students made is the following. Define the following TM $R_0$ to try to decide $EQ_{TM}$:

$$R_0 = \text{“On input } \langle M, N \rangle, \text{ where } M \text{ and } N \text{ are TMs:}$$

1. For a string $w$, run $M$ and $N$ on $w$.
2. If $M$ and $N$ both accept or both don’t, then $M$ and $N$ are equivalent, so accept; otherwise, reject.

There are several problems with this approach. First, in stage 1 what is the string $w$ on which to test the TMs $M$ and $N$? For $M$ and $N$ to be equivalent, $R$ would have to test every possible string $w \in \Sigma^*$, and make sure that $M$ and $N$ both accept or both don’t accept. Hence, on a YES instance (i.e., when $M$ and $N$ are equivalent), the TM $R_0$ would be stuck in an infinite loop since there are infinitely many strings $w \in \Sigma^*$ to test, and $M$ and $N$ would agree on all of them when $M$ and $N$ are equivalent. In other words, $R_0$ loops on $\langle M, N \rangle \in EQ_{TM}$, so $R_0$ doesn’t even recognize $EQ_{TM}$.

Another problem is that in stage 1 of $R_0$, it may not be safe to run $M$ and $N$ on $w$ since one or both might loop, in which case $R_0$ can’t be a decider since it doesn’t always halt. Moreover, there is no way to determine if $M$ or $N$ accept $w$ since the acceptance problem for TMs (i.e., $A_{TM}$) is undecidable. You might think that this then proves that $EQ_{TM}$ is undecidable, but this only shows that one particular way (i.e., TM $R_0$) does not decide $EQ_{TM}$, but there might be another TM that does decide $EQ_{TM}$. To prove that $EQ_{TM}$ is undecidable, you need to show that every TM will fail to decide $EQ_{TM}$, and this is accomplished via a reduction, as in the solution. If there were a decider $R$ for $EQ_{TM}$, then we could use $R$ to construct a decider $S$ for $E_{TM}$. But since $E_{TM}$ is undecidable (Theorem 5.2), it must be the case that $EQ_{TM}$ does not have a decider, i.e., $EQ_{TM}$ is undecidable.