CS 341, Spring 2013
Solutions for Midterm 2

1. (a) True, since the definition of Turing-decidable is more restrictive than the definition of Turing-recognizable.
   (b) True, by Theorem 3.13.
   (c) True, by slide 4-25.
   (d) False, e.g., if $A = \{00, 11\}$ and $B = \{00, 11, 111\}$, then $A \cap \overline{B} = \emptyset$ but $A \neq B$. For A and B to be equal, we instead need $(\overline{A} \cap B) \cup (A \cap \overline{B}) = \emptyset$.
   (e) False, since the set $\mathcal{N} = \{1, 2, 3, \ldots\}$ is countable.
   (f) True, since every regular language is context-free by Corollary 2.32, and every context-free language is decidable by Theorem 4.9.
   (g) True. This is just the definition of co-Turing-recognizable.
   (h) False, by Theorem 3.16.
   (i) False. A TM M may loop on input w.
   (j) False. $\overline{A_{\mathrm{TM}}}$ is not Turing-recognizable by Corollary 4.23.

2. (a) Yes, because each element in A maps to a different element in B.
   (b) No, because there is no element in A that maps to $4 \in B$.
   (c) No, because f is not onto.
   (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.

3. (a) $q_1 110\#01 \quad x q_3 10\#01 \quad x1q_3 0\#01 \quad x10q_3\#01 \quad x10\#q_5 01 \quad x10\#0q_{\mathrm{reject}}1$
   (b) $q_1 0\#0 \quad xq_2\#0 \quad x\#q_4 0 \quad xq_6\#x \quad q_7 x\#x \quad xq_1\#x \quad x\#q_8 x \quad x\#xq_8 \quad x\#x\sqcup q_{\mathrm{accept}}$

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 = "On input string w:

1. Run M and M' alternatively on w step by step.

2. If M accepts w, *accept*. If M' accepts w, *reject*.

Then D decides A, so A is decidable.

5. Define the language as

$$C = \{\langle N, R\rangle \mid N \text{ is an NFA and } R \text{ is a regular expression with } L(N) = L(R)\,\}.$$

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

T = "On input $\langle N, R\rangle$, where N is an NFA and R is a regular expression:

1. Convert N into an equivalent DFA D using the algorithm in the proof of Kleene's Theorem.

2. Convert R into an equivalent DFA D' using the algorithm in the proof of Kleene's Theorem.

3. Run TM F from Theorem 4.5 on input $\langle D, D'\rangle$.

4. If F accepts, *accept*. If F rejects, *reject*."

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

S = "On input $\langle M\rangle$, where M is a TM:

1. Run R on input $\langle M, M_\emptyset\rangle$, where $M_\emptyset$ is a TM such that $L(M_\emptyset) = \emptyset$.

2. If R accepts, *accept*; if R rejects, *reject*.

The TM S just checks if the inputted TM M is equivalent to the empty TM $M_\emptyset$, so S decides E_{TM}. But E_{TM} is undecidable, so that must mean the decider R for EQ_{TM} cannot exist, so EQ_{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 = "On input $\langle M, N\rangle$, where M and N 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 $\langle M, N \rangle \in EQ_{\text{TM}}$, 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.