CS 341, Fall 2014
Solutions for Midterm 2

1. (a) False, by Theorem 5.4.
   (b) True, by Theorem 4.5.
   (c) False, by Corollary 4.23.
   (d) True, by Theorem 4.9.
   (e) True, by Theorem 4.9.
   (f) False, e.g., $\overline{A_{\text{TM}}}$ is not Turing-recognizable.
   (g) False. A TM M may loop on input w.
   (h) True. List the strings in string order.
   (i) False. Homework 9, problem 1.
   (j) True, by Theorems 3.13 and 3.16.

2. (a) No, because $f(x) = f(z) = 2$.
   (b) Yes, because all elements in $B = \{1, 2\}$ are hit: $f(y) = 1$ and $f(x) = 2$.
   (c) No, because f is not one-to-one.
   (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. $q_1 0\#0 \quad xq_2\#0 \quad x\#q_4 0 \quad xq_6\#x \quad q_7 x\#x \quad xq_1\#x \quad x\#q_8 x \quad x\#xq_8$ $x\#x\sqcup q_{\text{accept}}$

4. This is HW 7, problem 2a. For any two decidable languages L_1 and L_2, let M_1 and M_2, respectively be the TMs that decide them. We construct a TM M' that decides the union of L_1 and L_2:

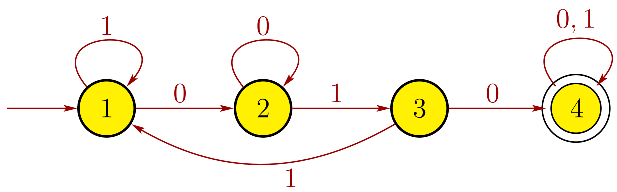
   M' = "On input string w:
   1. Run M_1 on w. If it accepts, *accept.*
   2. Run M_2 on w. If it accepts, *accept.* Otherwise, *reject.*

   M' accepts w if either M_1 or M_2 accepts it. If both reject, M' rejects.

5. This is a slight modification of HW 8, problem 3. Let $\Sigma = \{0, 1\}$, and the language of the decision problem is

$$A = \{ \langle R \rangle \mid R \text{ is a regular expression describing a language over } \Sigma \text{ containing at least one string } w \text{ that has 010 as a substring (i.e., } w = x010y \text{ for some } x \text{ and } y) \}.$$

Define the language $C = \{\, w \in \Sigma^* \mid w \text{ has } 010 \text{ as a substring} \,\}$. Note that C is a regular language with regular expression $(0 \cup 1)^* 010 (0 \cup 1)^*$ and is recognized by the following DFA D_C:



Now consider any regular expression R with alphabet Σ. If $L(R) \cap C \neq \emptyset$, then R generates a string having 010 as a substring, so $\langle R \rangle \in A$. Conversely, if $L(R) \cap C = \emptyset$, then R does not generate any string having 010 as a substring, so $\langle R \rangle \notin A$. By Kleene's Theorem, since $L(R)$ is described by regular expression R, the language $L(R)$ must be a regular language. Since C and $L(R)$ are regular languages, $C \cap L(R)$ is regular since the class of regular languages is closed under intersection, as was shown in Chapter 1. Thus, $C \cap L(R)$ has some DFA $D_{C \cap L(R)}$. Theorem 4.4 shows that $E_{\text{DFA}} = \{\, \langle B \rangle \mid B \text{ is a DFA with } L(B) = \emptyset \,\}$ is decidable, so there is a Turing machine H that decides E_{DFA}. We apply TM H to $\langle D_{C \cap L(R)} \rangle$ to determine if $C \cap L(R) = \emptyset$. Putting this all together gives us the following Turing machine T to decide A:

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

1. Convert R into a DFA D_R using the algorithm in the proof of Kleene's Theorem.

2. Construct a DFA $D_{C \cap L(R)}$ for language $C \cap L(R)$ from the DFAs D_C and D_R.

3. Run TM H that decides E_{DFA} on input $\langle D_{C \cap L(R)} \rangle$.

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

6. 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.

7. This is Theorem 5.1, whose proof is given on slide 5-8. Suppose that $HALT_{\mathrm{TM}}$ is decidable and that it is decided by a TM R. Define the following TM S, which will decide A_{TM} using R as a subroutine:

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

1. Run R on input $\langle M, w \rangle$.

2. If R rejects, then *reject*.

3. If R accepts, then run M on input w until it halts."

4. If M accepts w, *accept*; otherwise, *reject*."

Note that stage 1 checks if it is safe to run M on w. If not, then M loops on w, so S rejects $\langle M, w \rangle \notin A_{\mathrm{TM}}$, which is stage 2. If stage 1 determines it is safe to run M on w, then stage 3 runs M on w, and then stage 4 gives the same output. In particular, if M accepts w, then S accepts $\langle M, w \rangle$; if M rejects w, then S rejects $\langle M, w \rangle$.

Thus, we have shown that A_{TM} reduces to $HALT_{\mathrm{TM}}$. But since A_{TM} is undecidable, we must have that $HALT_{\mathrm{TM}}$ is also undecidable.