CS 341, Fall 2016, Face-to-Face Section
Solutions for Midterm 2

1. (a) False. A TM M may loop on input w.
   (b) False. $\overline{A_{\text{TM}}}$ is not Turing-recognizable by Corollary 4.23.
   (c) True, because the definition of Turing-decidable is more restrictive than the definition of Turing-recognizable.
   (d) True, by Theorem 3.13.
   (e) True, by slide 4-25.
   (f) 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 $(\overline{A} \cap B) \cup (A \cap \overline{B}) = \emptyset$.
   (g) False, because the set $\mathcal{N} = \{1, 2, 3, \ldots\}$ is countable.
   (h) True, because every regular language is context-free by Corollary 2.32, and every context-free language is decidable by Theorem 4.9.
   (i) True, by slide 4-38.
   (j) False, by Theorem 3.16.

2. (a) No, because $f(x) = f(z) = 1$.
   (b) Yes, because all elements in $B = \{1, 2\}$ are hit: $f(x) = 1$ and $f(y) = 2$.
   (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, i.e., a decider.

3. $q_1 0\#0$ $\quad xq_2\#0$ $\quad x\#q_4 0$ $\quad xq_6\#x$ $\quad q_7 x\#x$ $\quad xq_1\#x$ $\quad x\#q_8 x$ $\quad x\#xq_8$ $\quad x\#x\sqcup q_{\text{accept}}$

4. This is HW 9, problem 1. Each element in $\mathcal{B}$ is an infinite sequence $(b_1, b_2, b_3, \ldots)$, where each $b_i \in \{0, 1\}$. We prove that $\mathcal{B}$ is uncountable by contradiction. Suppose $\mathcal{B}$ is countable. Then we can define a correspondence f between $\mathcal{N} = \{1, 2, 3, \ldots\}$ and $\mathcal{B}$. Specifically, for $n \in \mathcal{N}$, let $f(n) = (b_{n1}, b_{n2}, b_{n3}, \ldots)$, where b_{ni} is the ith bit in the nth sequence, i.e.,

n	$f(n)$
1	$(b_{11}, b_{12}, b_{13}, b_{14}, b_{15}, \ldots)$
2	$(b_{21}, b_{22}, b_{23}, b_{24}, b_{25}, \ldots)$
3	$(b_{31}, b_{32}, b_{33}, b_{34}, b_{35}, \ldots)$
4	$(b_{41}, b_{42}, b_{43}, b_{44}, b_{45}, \ldots)$
$\vdots$	$\vdots$

Now define an infinite binary sequence $c = (c_1, c_2, c_3, c_4, c_5, \ldots) \in \mathcal{B}$, where $c_i = 1 - b_{ii}$ for each $i \in \mathcal{N}$. In other words, the ith bit in c is the opposite of the ith bit in the ith sequence. For example, if

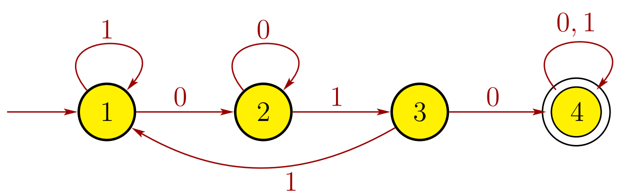
n	$f(n)$
1	$(0, 1, 1, 0, 0, \ldots)$
2	$(1, 0, 1, 0, 1, \ldots)$
3	$(1, 1, 1, 1, 1, \ldots)$
4	$(1, 0, 0, 1, 0, \ldots)$
$\vdots$	$\vdots$

then we would define $c = (1, 1, 0, 0, \ldots)$. Thus, for each $n = 1, 2, 3, \ldots$, note that $c \in \mathcal{B}$ differs from the nth sequence in the nth bit, so c does not equal $f(n)$ for any $n \in \mathcal{N}$, which is a contradiction because the enumeration was supposed to contain every infinite binary sequence. Hence, $\mathcal{B}$ is uncountable.

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 $C = \{ w \in \Sigma^* \mid w \text{ has 010 as a substring} \}$. 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 $\langle R \rangle \notin A$. Because $L(R)$ is described by regular expression R, the language $L(R)$ must be a regular language by Kleene's Theorem. Because C and $L(R)$ are regular languages, $C \cap L(R)$ is regular because 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 then run TM H on input $\langle D_{C \cap L(R)} \rangle$ 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. **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, because 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. 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. Because A is Turing-recognizable, there is a TM M with $L(M) = A$. Because 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. Alternate running one step on each of M and M', both on input w.
   2. If M accepts w, *accept*. If M' accepts w, *reject*.

   Because exactly one of M or M' will accept w, we see that D can't loop. Also, if $w \in A$, then M is the TM that will accept, so D accepts w. If $w \notin A$, then M' is the TM that will accept, so D rejects w. Hence, D decides A, so A is decidable.

7. This is Theorem 5.1, whose proof is given on slide 5-8. Suppose that $HALT_{\text{TM}}$ 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; i.e., if M doesn't loop on w. If not, then M loops on w, so S rejects $\langle M, w\rangle \notin A_{\text{TM}}$, 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 because A_{TM} is undecidable, we must have that $HALT_{\mathrm{TM}}$ is also undecidable.