CS 341, Fall 2018
Solutions for Midterm 2

1. (a) True, by Theorem 4.5.

   (b) False, by Theorem 3.13.

   (c) False, by Corollary 3.15.

   (d) False. For example, consider the language with regular expression $(0 \cup 1)^*$. The language regular by Kleene's Theorem, so it is also context-free (Corollary 2.32). This implies it is further decidable (Theorem 4.9). But $(0 \cup 1)^*$ generates an infinite language.

   (e) False, by Theorem 4.11.

   (f) False, by Corollary 4.23.

   (g) False. We can decide the problem that an NFA and regular expression are equivalent by reducing the problem to EQ_{DFA}, which Theorem 4.5 shows is decidable. Here is a decider for the problem:
   $M =$ "On input $\langle N, R\rangle$, where N is an NFA and R is a regular expression:
   0. Check if $\langle N, R\rangle$ is a proper encoding of NFA N and regular expression R; if not, *reject*.
   1. Convert N into equivalent DFA D_1 using algorithm in Theorem 1.39.
   2. Convert R into equivalent DFA D_2 using algorithms in Lemma 1.55 and Theorem 1.39.
   3. Run TM S for EQ_{DFA} (Theorem 4.5) on input $\langle D_1, D_2\rangle$. If S accepts, then *accept*; else, *reject*."

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

   (i) False. TM M may loop on input w.

   (j) False, e.g., the set $\mathcal{N}$ of positive integers is infinite and countable.

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. $q_1 100\#0 \quad x q_3 00\#0 \quad x0q_3 0\#0 \quad x00q_3\#0 \quad x00\#q_5 0 \quad x00\#0q_{\text{reject}}$

4. This is HW 8, problem 4. We need to show there is a Turing machine that recognizes $\overline{E_{\text{TM}}}$, the complement of E_{TM}. Let $s_1, s_2, s_3, \ldots$ be a list of all strings in Σ^*, e.g., in string order. 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 $\langle M \rangle \in \overline{E_{\mathrm{TM}}}$; if M accepts none of the strings, then $L(M) = \emptyset$, so $\langle M \rangle \notin \overline{E_{\mathrm{TM}}}$. 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. Because M accepts s_2, we have that $\langle M \rangle \in \overline{E_{\mathrm{TM}}}$. 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 $\overline{E_{\mathrm{TM}}}$:

R = "On input $\langle M \rangle$, where M 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*.

5. (From slides 4-39 and 4-40). Let $\mathcal{L}$ be the collection of languages over an alphabet Σ, and let $\mathcal{B}$ be the set of infinite binary strings, which we know is uncountable (by a diagonalization argument, on slide 4-39). We will show that there is a correspondence between $\mathcal{L}$ and $\mathcal{B}$, so they have the same size. Let $s_1, s_2, s_3, \ldots$ be an enumeration of the strings in Σ^*, e.g., the enumeration can list the strings in string order. Define mapping $\chi : \mathcal{L} \to \mathcal{B}$ such that for a language $A \in \mathcal{L}$, the nth bit of $\chi(A)$ is 1 if and only if the nth string $s_n \in A$. We now show χ is a correspondence.

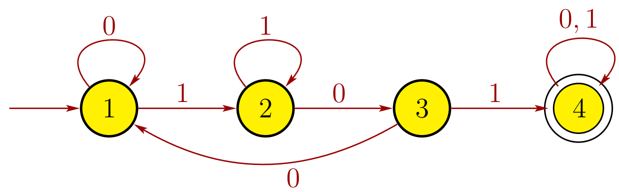
   - To show that χ is one-to-one, suppose that $A_1, A_2 \in \mathcal{L}$ with $A_1 \neq A_2$. Then there is some string s_i such that s_i is in one of the languages but not the other. Then $\chi(A_1)$ and $\chi(A_2)$ differ in the ith bit, so χ is one-to-one.
   - To show that χ is onto, consider any infinite binary sequence $b = b_1 b_2 b_3 \ldots \in \mathcal{B}$. Consider the language A that includes all strings s_i for which $b_i = 1$ and does not include any string b_j for which $b_j = 0$. Then $\chi(A) = b$, so χ is onto.

   Since χ is one-to-one and onto, it is a correspondence. Thus, $\mathcal{L}$ and $\mathcal{B}$ have the same size, so $\mathcal{L}$ is uncountable because $\mathcal{B}$ is uncountable.

6. This is a slight modification of HW 8, problem 3. The language of the decision problem is

$$A = \{ \langle R \rangle \mid R \text{ is a regular expression describing a language over } \Sigma \text{ containing at least one string } w \text{ that has 101 as a substring (i.e., } w \in L(R) \text{ and } w = x101y \text{ for some } x \in \Sigma^* \text{ and } y \in \Sigma^*) \}.$$

   Define the language $C = \{ w \in \Sigma^* \mid w \text{ has 101 as a substring} \}$. Note that C is a regular language with regular expression $(0 \cup 1)^* 101 (0 \cup 1)^*$ 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 101 as a substring, so $\langle R \rangle \in A$. Also, if $L(R) \cap C = \emptyset$, then R does not generate any string having 101 as a substring, so $\langle R \rangle \notin A$. By Kleene's Theorem, because $L(R)$ is described by regular expression R, $L(R)$ must be a regular language. Because C and $L(R)$ are regular languages, $C \cap L(R)$ is regular as the class of regular languages is closed under intersection, as was shown in Homework 2, problem 5. 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, where D_C is given above.

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*."

7. 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_{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 make 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.