CS 341-006, Hybrid section, Spring 2022
Solutions for Midterm 2

1. (a) True. For example, the universal Turing machine recognizes A_{TM}, which is undecidable.
   (b) False, by Theorems 3.13 and 3.16.
   (c) False. A TM M may loop on input w.
   (d) True, by Theorem 4.9.
   (e) True, by slide 4-38.
   (f) False, by Theorem 4.8.
   (g) False, e.g., $\overline{A_{\text{TM}}}$ is not Turing-recognizable.
   (h) True. $A \subseteq B$ means that every element of A also belongs to B, which is equivalent to saying that there are no elements in A that do not belong to B, i.e., $A \cap \overline{B} = \emptyset$.
   (i) True, by Theorem 4.5.
   (j) False, by Theorem 4.11.

2. (a) Yes, because $f(x) \neq f(y)$ whenever $x \neq y$.
   (b) Yes, because everything in R is "hit" by f.
   (c) Yes, because f is one-to-one and onto.
   (d) A language L_1 that is Turing-recognizable is recognized by a Turing machine M_1 that may loop forever on a string $w \notin L_1$. A language L_2 that is Turing-decidable is recognized by a Turing machine M_2 that always halts.
   (e) An algorithm is a Turing machine that always halts.

3. $q_1 0001\#10 \quad x q_2 001\#10 \quad x0 q_2 01\#10 \quad x00 q_2 1\#10 \quad x001 q_2 \#10 \quad x001\# q_4 10$ $x001\#1 q_{\text{reject}}$

4. (This is a slight modification of Theorem 4.17.) For a proof by contradiction, suppose that $A = \{ x \in \Re \mid 6 \leq x < 7 \}$ is countable. The set A is clearly infinite, so the assumption that A is countable means that we can define a correspondence $f : \mathcal{N} \to A$, where $\mathcal{N} = \{1, 2, 3, \ldots\}$ is the set of natural numbers, and let $a_n = f(n)$. In other words, we can enumerate the elements of A as a list $a_1, a_2, a_3, \ldots$, where

n	$f(n) = a_n$
1	$6.d_{11}d_{12}d_{13}\ldots$
2	$6.d_{21}d_{22}d_{23}\ldots$
3	$6.d_{31}d_{32}d_{33}\ldots$
$\vdots$	$\ddots$

For the nth number a_n in the list, its ith digit after the decimal point is d_{ni}. Now we construct a number $y \in A$ as $y = 6.b_1 b_2 b_3 \ldots$, where for each $n = 1, 2, 3, \ldots$, the nth digit in y after the decimal point is $b_n = 3$ if $d_{nn} = 1$, and $b_n = 1$ if $d_{nn} \neq 1$. The number y belongs to the set A, but for each $n = 1, 2, 3, \ldots$, the number y but does not equal the nth number in the list because they differ in the nth digit, i.e., $b_n \neq d_{nn}$. Therefore, we get a contradiction because the list was supposed to contain all elements of A, but the list does not include $y \in A$. We thus conclude that A is uncountable.

5. (This is HW 7, problem 2b.) For any two Turing-recognizable languages L_1 and L_2, let M_1 and M_2, respectively, be TMs that recognize them. We construct a TM M' that recognizes the union $L_1 \cup L_2$:

   M' = "On input string w:

   1. Run M_1 and M_2 alternately on w, one step at a time.
   If either accepts, *accept*. If both halt and reject, *reject*.

   To see why M' recognizes $L_1 \cup L_2$, first consider $w \in L_1 \cup L_2$. Then w is in L_1 or in L_2 (or both). If $w \in L_1$, then M_1 accepts w, so M' will eventually accept w. Similarly, if $w \in L_2$, then M_2 accepts w, so M' will eventually accept w. On the other hand, if $w \notin L_1 \cup L_2$, then $w \notin L_1$ and $w \notin L_2$. Thus, neither M_1 nor M_2 accepts w, so M' will also not accept w. Hence, M' recognizes $L_1 \cup L_2$. Note that if neither M_1 nor M_2 accepts w and one of them does so by looping, then M' will loop, but this is fine because we only needed M' to *recognize* and not *decide* $L_1 \cup L_2$.

6. (This is a modification of Theorem 4.5, which shows that EQ_{DFA} is decidable.) The language of the decision problem is

   $$A = \{ \langle D_1, D_2 \rangle \mid D_1 \text{ and } D_2 \text{ are DFAs with } L(D_1) \subseteq L(D_2) \ \}.$$

   To simplify notation, let $L_1 = L(D_1)$ and $L_2 = L(D_2)$, and define $L_3 = L_1 \cap \overline{L_2}$. Note that $L_1 \subseteq L_2$ if and only if $L_3 = \emptyset$, so we will show that A is decidable through a TM S that

   - first constructs a DFA D_3 for L_3 (using that the class of regular languages is closed under complementation (HW 2, problem 3) and intersection (HW 2, problem 5)), and
   - then checks if D_3 recognizes the empty language (the emptiness problem for DFAs is decidable by Theorem 4.4).

Specifically, here are the details of a decider S for A:

S = "On input $\langle D_1, D_2\rangle$, where D_1 and D_2 are DFAs:

0. If $\langle D_1, D_2\rangle$ is not a proper encoding of two DFAs, then *reject*.

1. Construct a DFA D_3 for language $L_3 = L(D_1) \cap \overline{L(D_2)}$ using the algorithms for DFA complementation and intersection.

2. Run TM R that decides E_{DFA} on input $\langle D_3\rangle$.

3. If R accepts, *accept*. If H rejects, *reject*."

Below are additional details, which are not required in an answer. To start, we are given two DFAs $\langle D_1, D_2\rangle$ as input. Then $L_1 = L(D_1)$ and $L_2 = L(D_2)$ are regular because they are the languages recognized by DFAs $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, respectively. The proof that the class of regular languages is closed under complementation (HW 2, problem 3) shows how to construct a DFA D_2' for $\overline{L_2}$ by swapping accepting and non-accepting states of the DFA D_2 for L_2; i.e., $D_2' = (Q_2, \Sigma, \delta_2, q_2, Q_2 - F_2)$ recognizes $\overline{L_2}$. Because the class of regular languages is also closed under intersections (HW 2, problem 5), we then have that $L_1 \cap \overline{L_2}$ is regular, and we can construct a DFA D_3 for $L_3 = L_1 \cap \overline{L_2}$ by running D_1 and D_2' simultaneously and accepting if and only if both accept; i.e., L_3 is recognized by $D_3 = (Q_3, \Sigma, \delta_3, q_3, F_3)$, where

- $Q_3 = Q_1 \times Q_2$,
- $\delta_3((x, y), \ell) = (\delta_1(x, \ell), \delta_2(y, \ell))$ for $(x, y) \in Q_3$ and $\ell \in \Sigma$,
- $q_3 = (q_1, q_2)$, and
- $F_3 = F_1 \times (Q_2 - F_2)$.

Now let TM R be the decider for E_{DFA} in Theorem 4.4 (emptiness problem for DFAs), and we can determine if the DFA D_3 recognizes the empty language (i.e., if $L_3 = \emptyset$, or equivalently if $L_1 \subseteq L_2$) by running R on input $\langle D_3\rangle$. Putting this all together gives us the above Turing machine S to decide A.

An incorrect answer is

S' = "On input $\langle D_1, D_2\rangle$, where D_1 and D_2 are DFAs:

0. If $\langle D_1, D_2\rangle$ is not a proper encoding of two DFAs, then *reject*.

1. Run D_1 on input string w.

2. If D_1 accepts, then run D_2 on string w.
If D_2 accepts, *accept*; else, *reject*.

One problem with TM S' is that in Stage 1, what is the string w? Even if D_1 and D_2 both accept one particular string w, there may be another string w' that D_1 accepts but D_2 rejects, so then $L(D_1) \not\subseteq L(D_2)$, in which case $\langle D_1, D_2\rangle \notin A$. But S' can't determine this because it tests D_1 and D_2 on only one specific string w. The only YES instances $\langle D_1, D_2\rangle \in A$ that S' accept are when $L(D_1) = \{w\}$ and

$w \in L(D_2)$. But there are many other YES instances $\langle D_1, D_2 \rangle \in A$ that S' does not accept, so S' does not even recognize A.

We could try to fix TM S' by constructing another TM S'' that tests D_1 and D_2 on all possible strings $w \in \Sigma^*$, but this modification also does not lead to a decider for A. Specifically, let $w_1, w_2, w_3, \ldots$ be an enumeration of the strings in Σ^* (e.g., in string order), and consider the following TM S'':

S'' = "On input $\langle D_1, D_2 \rangle$, where D_1 and D_2 are DFAs:

0. If $\langle D_1, D_2 \rangle$ is not a proper encoding of two DFAs, then *reject.*
1. For $i = 1, 2, 3, \ldots$
2. Run D_1 on input string w_i.
3. If D_1 accepts w_i, then run D_2 on string w_i.
If D_2 rejects w_i, *reject.*
4. If D_2 accepts every string w_i that D_1 accepts, *accept.*

A problem with TM S'' is that S'' does not even recognize the language A. To see why, first note that Σ^* is infinite, so the loop starting in Stage 1 could go on forever. Now suppose that $\langle D_1, D_2 \rangle$ is a NO instance; i.e., $\langle D_1, D_2 \rangle \notin A$, so there is at least one string w_k (depending on both D_1 and D_2) that D_1 accepts but D_2 rejects. The TM S'' will eventually find this string w_k, so S'' correctly will reject the NO instance. But when $\langle D_1, D_2 \rangle$ is a YES instance (i.e., $\langle D_1, D_2 \rangle \in A$), then the TM S'' will loop forever because it will never find a string w_i that D_1 accepts and D_2 rejects, so S'' never rejects in Stage 3 and never reaches Stage 4. Thus, TM S'' does not even recognize A because S'' does not accept every YES instance, and in fact, S'' loops on each YES instance.

7. (This is HW 8, problem 4, worded slightly differently.) 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^*. 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 $\langle M \rangle \in \overline{E_{\text{TM}}}$; if M accepts none of the strings, then $L(M) = \emptyset$, so $\langle M \rangle \notin \overline{E_{\text{TM}}}$. 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. Since M accepts s_2, we have that $\langle M \rangle \in \overline{E_{\text{TM}}}$. 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 $\overline{E_{\text{TM}}}$:

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.*