Attempt any **three** of the **six** problems below. The homework is graded on a scale of 100 points, even though you can attempt fewer or more points than that. Your recorded grade will be the total score on the problems that you attempt.

1. (**30 points**) For each language below, determine whether or not it is Turing decidable. If it is Turing decidable, describe a Turing machine that decides it. If it is not decidable, show this using a reduction from a problem shown to be undecidable in *Sipser*, from lectures. or earlier homework or midterm 2.

   (a) (**10 points**) $\{M \mid M$ describes a TM that halts when run with the empty string for input. $\}$

   Solution: Let A_ϵ denote the language for this problem. I'll reduce A_{TM} to A_ϵ. Construct a TM M_{ATM} does the following when run with input $M\#w$:

   Step 1. (M_{ATM}): Compute the description of a TM, M' that does the following when run with input s:

   Step 1. (M'): Erase it's input tape (i.e. overwrite every symbol in s with a blank.

   Step 2. (M'): Write w on its tape.

   Step 3. (M'): Run M on input w.

   Step 2.a. (M'): If M accepts w, M accepts.

   Step 2.b. (M'): If M rejects w, M rejects.

   Step 2.c. (M'): If M loops on w, M loops.

   Step 2. (M_{ATM}): Check if $M' \in A_\epsilon$ (i.e. run a decider for A_ϵ on the description of M'):

   Step 2.a. (M_{ATM}): If $M' \in A_\epsilon$, then M accepts $M\#w$.

   Step 2.b. (M_{ATM}): If $M' \notin A_\epsilon$, then M rejects $M\#w$.

   Thus, we've reduced deciding whether or not M accepts w to whether or not M' accepts the empty string, and our construction works for any M and w. This shows that $A_{TM} \leq_m A_\epsilon$; therefore X is undecidable (because we have already shown that A_{TM} is undecidable).

   (b) (**10 points**) $\{M \mid M$ describes a TM with exactly 42 states. $\}$

   Solution: Let $A_{42-states}$ denote the language described above. $A_{42-states}$ is Turing decidable. We can construct a TM, $M_{42-states}$ that reads the description of M. If M is not a valid description of a TM, then $M_{42-states}$ rejects. Otherwise, $M_{42-states}$ simply checks the number of states in the the description of M and accepts if there are 42 such states.

   For example, the format for describing TMs that presented in the Oct. 27 lecture has the binary encoding of the number of states of the machine as the first element of the description. Using this this format, $M_{42-states}$ just needs to make sure that its input is a valid TM description and then makes sure that this description starts with the substring:

   `101010,`

   (c) (**10 points**) $\{M \mid M$ describes a TM that accepts exactly 42 strings. $\}$

   Solution: Let $A_{42-strings}$ denote the language described above. $A_{42-strings}$ is Turing not decidable. I'll reduce A_{TM} to A_ϵ. For the same of contradiction, assume that there is a TM, $M_{42-strings}$ that decides $A_{42-strings}$. Construct a TM M_{ATM} does the following when run with input $M\#w$:

   Step 1. (M_{ATM}): Compute the description of a TM, M' that does the following when run with input s:

   Step 1. (M'): If $s = w$, run M on w.

   Step 1.a. (M'): If M accepts w, M accepts.

   Step 1.b. (M'): If M rejects w, M rejects.

   Step 1.c. (M'): If M loops on w, M loops.

   Step 2. (M'): If $s \in \{w+1, \ldots w+41\}$ accept. Here, $w+1$ is the lexigraphical successor to w (i.e. the first string, in lexigraphical ordering, that is greater than w).

Step 3. (M'): Otherwise, reject.

Step 2. (M_{ATM}): Check if $M' \in A_{42-strings}$ (i.e. run M_ϵ on the description of M'):

Step 2.a. (M_{ATM}): If $M' \in A_\epsilon$, then M accepts $M\#w$.

Step 2.b. (M_{ATM}): If $M' \notin A_\epsilon$, then M rejects $M\#w$.

Thus, we've reduced deciding whether or not M accepts w to whether or not M' accepts the empty string, and our construction works for any M and w. This shows that $A_{TM} \leq_m A_\epsilon$; therefore X is undecidable (because we have already shown that A_{TM} is undecidable).

2. (**30 points**) Same instructions as for problem 1.

   (a) (**10 points**) $\{M \mid M$ describes a TM that decides the halting problem. $\}$

   Solution: Let $A_{decides-HALT}$ denote the language described above. There are no TMs that decide the halting problem. Thus, $A_{decides-HALT} = \emptyset$ and is decidable – just make a TM that transitions to its reject state on its first move regardless of its input.

   (b) (**10 points**) $\{M \mid M$ never writes the symbol `0` on two consecutive moves. $\}$

   Solution: Let $A_{\overline{00}}$ denote the language described above. I'll reduce A_ϵ (see my solution to question 1a to $A_{\overline{00}}$. Let M describe a TM. If M has the symbol `0` in its tape alphabet, we create a new TM, M' that is the same as M but with the symbol `0` replaced by a new symbol `0`$'$ that is not in the tape alphabet of M. If M does not have the symbol `0` in its tape alphabet, let $M' = M$. Now, we create a new machine, M" that is like M' except for the following changes:

   - Add the symbol `0` to the tape alphabet.
   - Add a new state, q" to the set of states.
   - Replace any transition to q_{accept} with a transition to q".
   - From state q" and for every tape symbol, M" writes a `0`, moves right and remains in state q".

   In other words, if M' accepts, the M" writes an infinite strings of `0`'s on its tape. Clearly, M" does not write *any* zeros on its tape at any other time. Thus, M" writes two consecutive zeros iff M' (and thus M) accepts when run with the empty string. This reduce A_ϵ to $A_{\overline{00}}$ as promised.

   (c) (**10 points**) $\{M \mid M$ describes a TM that decides every string, w, after at most $\sqrt{|w|} + 12$ moves. $\}$

   Solution (sketch): If M makes 17 or more moves, we can give it an input string that is too short to justify the run-time. Thus, it is sufficient to show that M always halts after at most 16 moves, and this only requirs looking at strings of length of up to 16.

3. (**35 points**, Sipser problems 5.22, 5.23 and 24)

   (a) (**10 points**) Show that A is Turing-recognizable iff $A \leq_m A_{TM}$.

   Solution (sketch): Follows directly from the definition of A_{TM}.

   (b) (**10 points**) Show that A is Turing-decidable iff $A \leq_m 0^*1^*$.

   Solution: If A is Turing decidable, then we can make a machine that when run with input w checks to see if $w \in A$. If so, it erases its tape, and runs a decider for 0^*1^* – because $\epsilon \in L(0^*1^*)*$, that machine accepts. Conversely, if $w \notin A$, then it writes 10 on its tape and runs the decider for 0^*1^*.

   (c) (**15 points**) Let $J = \{w \mid$ either $w = 0x$ for some $x \in A_{TM}$ or $w = 1y$ for some $y \notin A_{TM}\}$. Show that neither J nor $\overline{J}$ is Turing-recognizable.

   Solution: We can reduce $\overline{A_{TM}}$ to J – to determine if $w \in \overline{A_{TM}}$, prepend a 0 to w and check if $0w \in J$. $\overline{A_{TM}}$ is not Turing reducible to A_{TM}; therefore, J cannot be reduced (by a Turing machine computation) to A_{TM} either which means that it is not Turing recognizable (see part (a)).

   Likewise, we can reduce $\overline{A_{TM}}$ to $\overline{J}$ by prepending a 0 to the input string which shows that $\overline{J}$ is not Turing recognizable either.

4. (**40 points**, Rice's Theorem: from *Sipser* problem 5.29)

Rice's Theorem (see *Sipser* problem 5.28): Let

$$A = \{M \mid M \text{ describes a TM such that } p(M).\ \}$$

Where p satisifies the following two properties:

(1) p is non-trivial: there is at least one TM, M_1, such that $p(M_1)$ is true and at least one TM, M_2, such that $p(M_2)$ is false.

(2) p is a property of the *language* recognized by M.

Then, A is not Turing decidable.

Sipser gives a proof for this theorem in the solution to problem 5.28.

(a) (**2**0 points, *Sipser* problem 5.29) Sipser's proof shows that the two conditions stated above for p are *sufficient* to prove that A is not Turing decidable. Show that both conditions are also *necessary*.

Solution: If p is trivial, then either all TM descriptions are in the language (if p includes all TMs) or the language is empty. In the former case, we just build a TM that makes sure that M is a valid TM description and accepts. In the lattter case, we build a TM that rejects all strings (e.g. see the solution to question 2a).
If p is a property of the machine rather than the language, it *may* be decidable. For example the question of whether or not a TM has 42 states (see question 1b) is decidable.

(b) (**1**0 points) Use Rice's theorem to prove that

$$B = \{M \mid \text{Every string accepted by } M \text{ has and equal number of } \mathtt{a}\text{'s and } \mathtt{b}\text{'s.}\ \}$$

is not Turing decidable.

Solution: B is a property of the language of a TM. Let M_1 be a TM that rejects all strings: $M_1 \in B$. Let M_2 be a TM that accepts all strings: $M_2 \notin B$. Thus, B is not trivial. By Rice's theorem, B is undecidable.

(c) (**1**0 points) Use Rice's theorem to prove that

$$C = \{M \mid M \text{ recognizes a context-free language.}\ \}$$

is not Turing decidable.

Solution: C is a property of the language of a TM. Let M_1 be a TM that rejects all strings: $M_1 \in C$. Let M_2 be a TM that recognizes $\{s \mid \exists n.\ s = a^n b^n c^n\}$; $M_2 \notin C$. Thus, C is not trivial. By Rice's theorem, C is undecidable.

5. (**45 points**) Let

$$E = \{M_1\#M_2 \mid M_1 \text{ and } M_2 \text{ describe TMs such that } L(M_1) = L(M_2).\ \}$$

Prove that E is complete for class Π_2 of the arithmetic hierarchy (see the Nov. 7 slides). This means that there is a Turing computable reduction from any language in Π_2 to E, and a Turing computable reduction from E to some language in Π_2. You may use the fact that $TOTAL$ is complete for Π_2, where

$$TOTAL = \{M \mid M \text{ is a decider.}\ \}$$

Solution:

$TOTAL \leq_M E$: Let M be a description of a TM. Compute the description of a new TM, M' that is the same as M except that all transitions that go to the reject state of M are changed to go to the accept state of M'. Note that M' accepts a string, w, iff M halts when run with input w. In other words, M' recognizes Σ^* iff $M \in TOTAL$.

Let M_{Σ^*} be the description of a TM that recognizes Σ^*. Now we have:

$$M' \# M_{\Sigma^*} \in E \quad \Leftrightarrow \quad M \in TOTAL$$

Deriving $M' \# M_{\Sigma^*}$ from M is a Turing computable function. Thus, we've shown $TOTAL \leq_M E$ as claimed.

$E \leq_M TOTAL$: Let $M_1 \# M_2$ be a string where M_1 and M_2 describe TMs. We will construct M', the description of a TM that is in $TOTAL$ iff $L(M_1) = L(M_2)$.

Let $s \in L(M_1)$. This means that there is some integer, n_1 such that $accept(M_1, s, n_1)$ where $accept(M, s, n)$ means that the TM described by M accepts the string described by s after at most n moves (see slide 15 of the Nov. 7 slides). If $L(M_1) = L(M_2)$, then there must be some integer, n_2 such that $accept(M_2, s, n_2)$. We can write this with quantifiers as:

$\forall s \in$

$\equiv$ $\forall s \in$

$\forall s \in Sigma^*.\ (\forall n_1 \in \mathbb{Z}.\ \neg accept(M_1, s, n_1))\ \vee\ (\exists n_2 \in \mathbb{Z}.\ accept(M_1, s, n_1)),$ De M

$\forall s \in Sigma^*, n_1 \in \mathbb{Z}.\ \neg accept(M_1, s, n_1)\ \vee\ (\exists n_2 \in \mathbb{Z}.\ accept(M_1, s, n_1)),$ Combine like quantifiers

$\forall s \in Sigma^*, n_1 \in \mathbb{Z}.\ \exists n_2 \in \mathbb{Z}.\ \neg accept(M_1, s, n_1) \vee accept(M_1, s, n_1)),$ Push $accept(M_1, s, n_1)$ inside $\exists$

$\forall s \in Sigma^*, n_1 \in \mathbb{Z}.\ \exists n_2 \in \mathbb{Z}.\ accept(M_1, s, n_1) \Rightarrow accept(M_1, s, n_1)),$ $(p \Rightarrow q) \equiv (\neg p \vee q)$

Likewise, we require that any string accepted by M_2 is accepted by M_1. Combining these two requirement, we get

$$\begin{aligned} & M_1 \# M_2 \in E \\ \Leftrightarrow \quad & \forall s \in Sigma^*, n_1 \in \mathbb{Z}.\ \exists n_2 \in \mathbb{Z}.\ accept(M_1, s, n_1) \Leftrightarrow accept(M_1, s, n_1)), \end{aligned}$$

Thus, $E \in \Pi_2$. From the problem statement, $TOTAL$ is complete for Π_2. Thus, $E \leq_M \Pi_2$.

The above explanation completes a perfectly acceptable solution. I'll now finish the solution without relying on the claim from the problem statement that $TOTAL$ is complete for Π_2. Given $M_1 \# M_2$, construct M', the description of a TM that on input $w\$n$ does the following:

1. Run M_1 for n steps on string w.
   If M_1 accepts within n steps,
   - 1.a. then, M' runs M_2 on w.
     If M_2 accepts,
     - 1.a.i then, M' goes to step 2.
     - 1.a.ii otherwise (M_2 rejects or loops on w), M' loops.
   - 1.b. otherwise (M_1 rejects or is still running after n steps on input w), M' goes to step 2.
2. Run M_2 for n steps on string w.
   and take the corresponding actions as described above, exchanging the roles of M_1 and M_2.

With this construction, M' loops on input $n\$w$ iff either of M_1 or M_2 accepts w after at most n steps and the other machine rejects or loops when run with input w. M' halts for all other inputs. Thus,

$$M' \in TOTAL \quad \Leftrightarrow \quad M_1 \# M_2 \in E$$

This shows that $E \leq_M TOTAL$.

Note: it is straightforward to generalize the argument above to show that $TOTAL$ is complete for Π_2 as claimed in the problem statement.

Having shown that $TOTAL \leq_M E$ and $E \leq_M TOTAL$ and give that $TOTAL$ is complete for Π_2, we conclude that E is complete for Π_2.

6. (**50 points**, adapted from *Kozen*) Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM that never overwrites its input. Formally,

$$\begin{array}{rl} & \mathit{ImmutableInput}(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}) \\ = & \forall q, q' \in Q.\ \forall c, c' \in \Gamma.\ \forall d \in \{L, R\}.\ ((\delta(q, c) = (q', c', d)) \wedge (c \in \Sigma)) \Rightarrow (c = c') \end{array}$$

M can write anything it wants on the portion of the tape that is initially blank.

(a) (30 points) Prove that for any TM M with $\mathit{ImmutableInput}(M)$, $L(M)$ is regular.

Solution: We start by considering how much the TM M can "figure out" about its input string by making read-only passes over the string. We then show that a DFA can do the same thing. Without loss of generality, I'll assume that the input string to M is not the empty string – if the set of not empty strings that M accepts is A and A is regular, then the set of all strings that M accepts is either A or $A \cup \{\epsilon\}$ which are both regular sets as well.

Let M be a TM; and let α be a configuration of M. I'll write $\Psi_M(\alpha, k)$ to denote the state that M is in the *first* time it reaches the k^{th} tape square when starting from configuration α. If M does not reach the k^{th} tape square, then I'll define $\Psi_M(\alpha, k)$ as described below:

If M reaches the accepting state, then $\Psi_M(\alpha, k) = q_{accept}$.

If M reaches the rejecting state, then $\Psi_M(\alpha, k) = q_{reject}$.

If M loops and therefore never reaches the k^{th} tape square, then $\Psi_M(\alpha, k) = q_{reject}$. Looping can be detected: the machine loops iff it visits the same square twice in the same state.

Now, consider running M on input w. By our assumption that $w \neq \epsilon$, we can write $w = uc$ for some $c \in \Sigma$. Let $Q = \{q_0, q_1, \ldots q_{n-1}\}$ be the set of states for M (for example q_{accept} could be q_1 and q_{reject} could be q_2). Shortly, I will show how we can build a machine M' that when run on input w does the following:

1. Scan across w (only moving to the left).
2. Write a string of the form

$$\Psi_M(q_0 w, |w|{+}1) \cdot \Psi_M(u q_0 c, |w|{+}1) \cdot \Psi_M(u q_1 c, |w|{+}1) \cdot \Psi_M(u q_2 c, |w|{+}1) \cdots \Psi_M(u q_{n-1} c, |w|{+}1) \dashv$$

on its tape immediately after w. If $\Psi_M(q_0 w, |w|+1) = q_{accept}$, M' immediately accepts, and if If $\Psi_M(q_0 w, |w|+1) = q_{reject}$, M' immediately rejects.

3. Move the tape head to the first blank square (immediately after the $\dashv$) and simulate M starting from state $\Psi_M(q_0 w |w| + 1)$. If M ever moves to the $\dashv$ symbol, this means that M would make another sojourn into w. If M is in state q when it moves to the $\dashv$ symbol, then M' looks up $\Psi_M(uqc, |w| + 1)$ on its tape, and moves back to the symbol after the $\dashv$ in that state. Note that this correctly simulates M.

With these moves, M' accepts w iff M accepts it. Thus, whether or not $w \in L(M)$ can be determined from the string that M' wrote, listing the values for Ψ_M. We'll complete this proof that $L(M)$ is regular by showing that this list of function values corresponds to writing down the state of a DFA.

For any non-empty, string sc let

$$\lambda(sc) \quad = \quad (\Psi(q_0 sc, |sc| + 1), \Psi(s q_0 c, |sc| + 1), \Psi(s q_1 c, |sc| + 1), \ldots \Psi(s q_{n-1} c, |sc| + 1))$$

Λ has $n + 1$ elements, each of which has n possible values. Thus there are n^{n+1} possible values for λ. Let Λ denote the set of all possible values for λ plus one extra value, λ_0. We'll note that for $c \in \Sigma$, $\lambda(c)$ is straighforward to compute, and that for $s \neq \epsilon$, $\lambda(sc)$ depends only on s and c. Let $\delta(\lambda_0, c) = \lambda(c)$, and for $\theta = \lambda(s)$, let $\delta(\theta, c) = \lambda(sc)$. Now we define a DFA $D = (\Lambda, \Sigma, \delta, \lambda_0, F)$. Note that the state of D after reading string w corresponds to the string that M' writes on its tape after reading the same string. Let F be the set of states for which M' eventually accepts. By construction: $L(D) = L(M') = L(M)$. Thus, M recognizes a regular language.

(b) (10 points) Show that the language

$$F_1 \quad = \quad \{M \mid \mathit{ImmutableInput}(M)\}$$

is Turing decidable.

Solution: $\mathit{ImmutableInput}$ is an assertion about the transition function. Just check the tuples that describe δ to make sure that there are none that modify tape squares that hold symbols from Σ.

(c) (10 points) Show that the language

$$F_2 \quad = \quad \{M\#w \mid \mathit{ImmutableInput}(M) \wedge (w \in L(M)\}$$

is not Turing decidable.

Solution: Build a TM that scans over its input, writes a "left-endmarker", writes w to the right of the endmarker, and runs some other machine, M' on w (M' never moves its head to the left of the endmarker). Now, M will recognizes Σ^* if M' accepts w and M recognizes $\emptyset$ otherwise. Thus, we've reduced A_{TM} to F_2 and conclude that F_2 is Turing undecidable.