1. (**15 points**) Let $A_1 = \{M\#w \mid M \text{ halts after at most } |w|^{|w|} \text{ steps when run with input } w\}$. Show that language $A$ is Turing decidable.

   **Solution:** I'll describe a TM, $M_{A1}$ that decides $A$. It is convenient to use a multi-tape $TM$ for $M_{A1}$. Here's what $M_{A1}$ does on input $M\#w$.

   1. $M_{A1}$ uses it's second tape to determine the length of $w$ and calculate $|w|^{|w|}$.
   2. $M_{A1}$ simulates $M$ running on input $w$. $M_{A1}$ counts the number of steps of the simulation.
      2.a. If $M$ halts (accepting or rejecting) after at most $|w|^{|w|}$ steps, $M_{A1}$ halts and accepts.
      2.b. Otherwise ($M$ is still running after $|w|^{|w|}$ steps), $M_{A1}$ rejects.

2. (**30 points**) Let $A_2 = \{M \mid \exists w.\ M\#w \in A_1\}$.

   (a) (**15 points**) Show that language $A_2$ is not Turing decidable.

   **Solution:** I'll reduce $A_{TM}$ to $A_2$. Given a string $M\#w$ that describes a TM, $M$, and an input string, $w$, the reduction constructs the description of a Turing machine $M'$ that on input $x$ does the following:

   1. Erases its tape.
   2. Writes $w$ on its tape.
   3. Moves its head back to the beginning of the tape.
   4. Runs $M$ on its tape.

   Constructing the description of $M'$ from the description of $M$ is clearly Turing computable, and $M'$ accepts $x$ iff $M$ accepts $w$. Let $N_{123}$ be the number of moves required to perform the first three steps described above. There are simple implementations with $N_{123} = 2(\max(|w|, |x|+1)+1)$. For large $|x|$, this is $N_{123} = 2|x| + 4 \ll |x|^{|x|}$.
   Now note that, $M'$ accepts $x$ in at most $|x|^{|x|}$ moves iff $M$ accepts $w$ in at most $|x|^{|x|} - N_{123}$ moves. If $M$ accepts $w$, we can find an $x$ that is long enough that $M'$ will accept. This shows that if $M\#x \in A_{TM}$ then $M' \in A_2$. If $M$ does not accept $w$ then $L(M') = \emptyset$; in other words, $M'$ rejects $x$ no matter what $x$ is. Thus, the description of $M'$ is in $A_2$ iff $M$ accepts $w$. This shows that $A_{TM} \leq_m A_2$. We know that $A_{TM}$ is not Turind decidable. Therefore, $A_{TM}$ is not Turing decidable either.

   (b) (**15 points**) Show that language $A_2$ is Turing recognizable.

   **Solution:** I'll reduce $A_2$ to $A_{TM}$. Let $M_{A2}$ be a TM that does the following on input $M$:

   1. If $M$ is not a valid Turing machine description, then $M_{A2}$ rejects immediately.
   2. Otherwise, $M_{A2}$ construct the description of a TM, $M'$ that does the following:

      for(each string $w \in \Sigma^*$) {
        run $M$ on input $w$ for at most $|w|^{|w|}$ moves.
        if($M$ halts after at most $|w|^{|w|}$ moves)
          accept;
      }

      Note that $M'$ accepts $M$ iff there is some string $w$ such that $M$ accepts $w$ after at most $|w|^{|w|}$ moves.
   3. If $M'\#M \in A_{TM}$, then $M_{A2}$ accepts. Otherwise, $M_{A2}$ rejects.

   Checking that $M$ is a valid Turing machine description is Turing computable. Furthermore, the construction of the description of $M'$ from the description of $M$ is Turing computable. Thus, this is a reduction from $A_2$ to $A_{TM}$. The language $A_{TM}$ is Turing-recognizable; therefore, $A_2$ is Turing-recognizable as well.

3. (**40 points**)

(a) (**10 points**) Show that the class of Turing-decidable languages is closed under complement.

**Solution:** Let $A$ be a Turing-decidable language. Because $A$ is Turing-decidable, there is some TM that decides $A$, let

$$M \quad = \quad (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

be TM that decides $A$. Because $M$ either accepts or rejects for any given input (i.e. it never loops), we can exchange the accept and reject states to obtain a TM that decides $\overline{A}$. Let

$$\overline{M} \quad = \quad (Q, \Sigma, \Gamma, \delta, q_0, q_{reject}, q_{accept})$$

Because $M$ never loops, $\overline{M}$ never loops. Thus, $L(\overline{M}) = \overline{L(M)} = \overline{A}$ and $\overline{M}$ decides $\overline{A}$. This shows that $\overline{A}$ is Turing-decidable. Because $A$ is an arbitrary, Turing-decidable language, this shows that the class of Turing-decidable languages is closed under complement.

(b) (**10 points**) Show that the class of Turing-decidable languages is closed under star.

**Solution:** Let $A$ be a Turing-decidable language, and let $M$ be a TM that decides $A$. We showed in class (and Sipser section 3.2) that non-deterministic TMs (NTMs) are equivalent to deterministic ones. I'll describe a NTM, $M_{A^*}$ that decides $A^*$. With input $w$, $M_{A^*}$ does the following:

1. If $w = \epsilon$, then $M_{A^*}$ accepts.
2. Otherwise, $M_{A^*}$ divides $w$ into strings $w_1, w_2, \ldots w_k$ such that $w_1 \cdot w_2 \cdots w_k = w$, and for each $1 \leq i \leq k$, $|w_i| > 0$.
   2.a. For each $1 \leq i \leq k$, $M_{A^*}$ runs $M$ on $w_i$.
   2.b. If $M$ accepts all of the $w_i$'s, then $M_{A^*}$ accepts $w$.
   2.c. Otherwise, $M$ rejects $w$.

Because $M$ never loops, TM $M_{A^*}$ never loops, and $L(M_{A^*}) = A^*$. Thus, $A^*$ is Turing-decidable which shows that the class of Turing-decidable languages is closed under star.

(c) (**10 points**) Show that the class of Turing-recognizable languages is not closed under complement.

**Solution:** For the sake of contradiction, assume that the Turing-recognizable langaugues are closed under complement. I will use this assumption to construct a TM that decides $A_{TM}$, a contradiction.

The language $A_{TM}$ is Turing-recognizable. This means that we can construct a TM, $M_{A_{TM}}$ that when run with input $M\#w$ accepts if $M$ is the description of a TM that accepts when run with input $w$. $M_{A_{TM}}$ may either reject or loop if $M$ does not accept $w$. If the class of Turing-recognizable languages were closed under complement, then $\overline{A_{TM}}$ would be Turing-recognizable. Let $M_{\overline{A_{TM}}}$ be a TM that recognizes $\overline{A_{TM}}$.

Now, I'll construct $D_{A_{TM}}$, a TM that *decides* $A_{TM}$. On input $w$ $D_{A_{TM}}$ simulates both $M_{A_{TM}}$ and $M_{\overline{A_{TM}}}$ running with input $w$. In particular, $D_{A_{TM}}$ alternates between simulating a step for $M_{A_{TM}}$ and simulating a step for $M_{\overline{A_{TM}}}$. Note that either $x \in L(M_{A_{TM}})$ or $x \in L(M_{\overline{A_{TM}}})$. Thus, $D_{A_{TM}}$ will eventually simulate a step where one of these machines halts. If the halting step is that $M_{A_{TM}}$ accepts $x$ (or $M_{\overline{A_{TM}}}$ rejects $x$), then $D_{A_{TM}}$ accepts. If the halting step is that $M_{\overline{A_{TM}}}$ accepts $x$ (or $M_{A_{TM}}$ accepts $x$), then $D_{A_{TM}}$ rejects. Thus, $D_{A_{TM}}$ is a decider for $A_{TM}$. We know that $A_{TM}$ is undecidable. Therefore, $D_{A_{TM}}$ cannot exist, which refutes our assumption that the Turing-recognizable languages are closed under complement.

This shows that the Turing-recognizable languages are not closed under complement.

(d) (**10 points**) Show that the class of Turing-recognizable languages is closed under star.

**Solution:** My solution is essentially the same as for showing that the Turing-decidable langauges are closed under star. In this case, if the input string $w$ is in $A^*$, then each substring will be accepted by $M$. Because $M$ recognizes $A$, it will halt for each substring. Thus, we can construct a recognizer from $A^*$ given a recognizer for $A$.

4. (**30 points**) A *linear bounded automaton* (LBA) is a Turing Machine with a bounded tape; it cannot move its head past *either* end of the input string. For example, you can assume that the input string has the form $\vdash v \dashv$ where $\vdash$ is a special left endmarker (that appears nowhere in $v$) and $\dashv$ is a special right endmarker (that appears nowhere in $v$). All transitions from $\vdash$ preserve the $\vdash$ and move the head to the right. All transitions from $\dashv$ preserve the $\dashv$ and move the head to the left. Let

$$A_{LBA} \quad = \quad \{M\#w \mid M \text{ is an LBA that accepts } w\}.$$

$A_{LBA}$ is Turing decidable (see Sipser Lemma 5.8). Thus, the halting problem for LBA's is Turing decidable as well.

Prove that there is some language, $B$ such that $B$ is Turing decidable but $B$ is not accepted by any LBA. (**Hint:** use diagonalization.)

**Solution:** Let

$$B \quad = \quad \{[M] \mid [M] \text{ describes an LBA that does not accept when run with } [M] \text{ as its input} \qquad \}$$

$B$ *is not accepted by any LBA.*

For the sake of contradiction, assume otherwise. Let $M_B$ be an LBA that accepts $B$. Run $M_B$ with its own description, $[M_B]$ as its input. If $M_B$ accepts, then $[M_B] \notin B$. On the other hand, if $M_B$ rejects or loops, then $[M_B] \in B$. Both cases lead to a contradiction. This shows that there is no LBA that accepts $B$.

$B$ *is Turing-decidable* As noted above, $A_{LBA}$ is Turing-decidable. As shown in problem 3a, the Turing-decidable languages are closed under complement. Thus, $\overline{A_{LBA}}$ is Turing-decidable. Let $M_{\overline{A_{LBA}}}$ be a TM that decides $\overline{A_{LBA}}$.

I'll now construct at TM, $T_B$ that decides $B$. On input $x$, $T_B$ constructs the string $x\#x$. $T_B$ then runs $M_{\overline{A_{LBA}}}$ on $x\#x$. If $M_{\overline{A_{LBA}}}$ accepts $x\#x$, then $T_B$ accepts $x$. Otherwise $T_B$ rejects. Because $M_{\overline{A_{LBA}}}$ is a decider, $M_{\overline{A_{LBA}}}$ never loops. Thus, $T_B$ never loops. $T_B$ is a TM that decides $B$. This shows that $B$ is Turing decidable.