1. (10 points) Fill in the blanks in the following proof via the Pumping Lemma that the language $L := \{0^m1^n \mid 0 \leq m \leq n \leq 2m\}$ is not regular:

Given $p > 0$,
let $s := \underline{0^p1^p}$. Clearly, $s \in L$ and $|s| \geq p$.
Given any $x, y, z$ such that $s = xyz$, $|y| > 0$, and $|xy| \leq p$,
let $i := \underline{2}$. Then $xy^iz \notin L$, because $\underline{[Explain briefly.] }$.

Answer: $s := 0^p1^p$ and $i := 2$. Given our choice of $s$, we must have $y = 0^k$ for some $k > 0$, so $xy^2z = 0^{p+k}1^p \notin L$, because $p + k > p$.

There are other correct answers, e.g., $s := 0^p1^{2p}$ and $i := p + 2$ (or bigger).

2. (20 points total)

(a) (10 points) Describe formally with a transition diagram a Turing machine $M$ with input alphabet $\Sigma := \{0, 1, \#\}$ that makes a “sorted” copy of a binary string. That is, on any input of the form “$w#$” where $w \in \{0, 1\}^*$, your machine $M$ eventually halts, leaving $w#x$ on the left end of its tape (and blanks everywhere else), where $x \in \{0, 1\}^*$ has the same number of 0’s and 1’s as $w$, but all the 0’s in $x$ precede all the 1’s in $x$. For example, we have the following nonblank tape contents:

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\underline{010010}$#</td>
<td>$\underline{010010}$#000011</td>
</tr>
<tr>
<td>$\underline{011101011}$#</td>
<td>$\underline{011101011}$#000111111</td>
</tr>
<tr>
<td>$#$</td>
<td>$#$</td>
</tr>
</tbody>
</table>

Please note that you can assume the input is $w#$ for some binary string $w$, and so the separator symbol $\#$ is already on the tape when the computation begins. If the input is not of this form, then $M$ may behave arbitrarily.

(b) (10 points) Give an implementation-level description of $M$.

Answer:

(a) Here is one possible $M$; the tape alphabet is $\{0, 1, $, $\#, B\}$, where $B$ is the blank symbol:
(b) $M$ first (from left to right) marks each 0 in $w$ with x, then append 0 to the end (and restoring the x back to 0. $M$ then does the same for the 1’s in $w$.

3. (15 points) Assume some fixed alphabet $\Sigma$ containing the symbol $a$. Give a high-level description of a decision procedure for the following language:

$$L := \{ \langle R \rangle \mid R \text{ is a regex and } a^* \subseteq R \}.$$

**Answer:** “On input $\langle R \rangle$, where $R$ is a regex:

(a) Convert $R$ to an equivalent DFA $D$.
(b) Let $n$ be the number of states of $D$.
(c) Run $D$ on input $a^i$ for all $0 \leq i \leq n$.
(d) If $D$ accepts all of these strings, then accept; else reject.”

This algorithm works because the path obtained by following $a$-transitions from the start state must loop back on itself after at most $n$ transitions.

4. (10 points) Let

$$L := \{ \langle M \rangle \mid (\exists n > 0)[M \text{ accepts exactly half the strings of length } n] \}.$$

Show that $A_{TM} \leq_m L$, and thus $L$ is undecidable. (Do not appeal to Rice’s theorem, which was not covered in class.)
**Answer:** Here is one possible solution: Fix some TM $M_0$ that does not accept any strings (and so $\langle M_0 \rangle \notin L$). Then let $f : \Sigma^* \to \Sigma^*$ be defined by the following algorithm: $f :=$ “On input string $s$:

(a) If $s$ is not of the form $\langle M, w \rangle$ for some TM $M$ and some string $w$ over $M$’s input alphabet, then output $\langle M_0 \rangle$ and halt.
(b) Otherwise, $s = \langle M, w \rangle$ as above. Construct a TM $R$ with input alphabet $\{0, 1\}$ as follows:
   $R :=$ ‘On input string $x \in \{0, 1\}^*$:
     i. If $x = 0$, then run $M$ on input $w$.
     ii. Otherwise, reject.’
(c) Output $\langle R \rangle$ and halt.”

Evidently, $f$ is computable. Given a TM $M$ and string $w$, letting $R$ be such that $\langle R \rangle = f(\langle M, w \rangle)$, we see that $R$ accepts at most one string: 0, one of the two strings of length one (the other is 1). $R$ accepts 0 if and only if $M$ accepts $w$, and so:

- $\langle M, w \rangle \in A_{TM}$ implies $M$ accepts $w$, which implies $R$ accepts exactly half the strings of length 1, which implies $\langle R \rangle \in L$.
- $\langle M, w \rangle \notin A_{TM}$ implies $M$ does not accept $w$, which implies $R$ does not accept any strings, which implies $\langle R \rangle \notin L$.

Finally, if string $s$ is not of the form $\langle M, w \rangle$ as above, then $s \notin A_{TM}$ and $f(s) = \langle M_0 \rangle \notin L$.

In any case, we always have $s \in A_{TM} \iff f(s) \in L$, and so $A_{TM} \leq_m L$ via $f$.

---

5. (15 points) Show that every Turing-recognizable language $L$ is enumerated by an enumerator that prints every string in $L$ exactly twice.

**Answer:** Given an enumerator $E$ for $L$, we modify it to the following enumerator $E'$ that behaves as required:

(a) Initialize a list of strings $D$ to be empty.
(b) Run $E$.
(c) Whenever $E$ prints a string $w$ that is not on the list $D$,
   i. Print $w$.
   ii. Print $w$ again.
   iii. Add $w$ to the list $D$.

$E'$ maintains a list $D$ of all strings printed by $E$ so far. Whenever $E$ first prints a string $w$, it is printed twice by $E'$ and added to the list. Having $w$ on the list means that $E'$ will not print $w$ again, no matter how many subsequent times $E$ prints $w$. 

3