There are four problems labeled A–D. There is also an optional Problem E.

When describing Turing machines, students are sometimes confused about how much detail to show — especially because we become less detailed as the course progresses. So let's be explicit about three possible levels of detail:

- At the Full Detail level, you explicitly state the tape alphabet, and you draw out the entire Turing machine as a directed graph with annotated edges. For clarity, please adopt the convention that any missing transition implicitly goes to the reject state. You organize your diagram into sections. You comment those sections, just as you would comment a computer program. You might need to do a rough draft before your final version. Our $\{a^m b^m c^m : m \geq 0\}$ example in class was done roughly at this level of detail.

- At the Moderate Detail level, you do not draw the Turing machine, but you do describe how the tape head moves across the tape, how the tape head marks or otherwise modifies the tape, and how the Turing machine decides to accept or reject. Usually there is a sequence of left-to-right or right-to-left scans involved. A fellow student, who sees your description at this detail level, should be able to produce a description at Full Detail. See below for an example.

- At the Low Detail level, you merely state an algorithm, as you would in any other CS course. A fellow student, who sees your description at this level of detail, should be able to produce a description at Moderate Detail (and then at Full Detail). Most of our course will operate at this level, but we're not there yet.

Here's an example at Moderate Detail. To understand it, I recommend that you execute it by hand on an example input such as 001001.

Let $A = \{ww : w \text{ is a string in } \{0,1\}^*\}$. Our decider for A uses an enlarged tape alphabet: In addition to the usual 0, 1, $\textvisiblespace$, $\vdash$, and $\dashv$, we have four symbols 0^L, 1^L, 0^R, 1^R. In this way, we can "mark" a 0 or 1 on the tape with an L or R. Now here's the decider.

1. Wrap the input in turnstiles.

2. Scan the entire tape left-to-right, marking the first unmarked 0 or 1 with an L.

3. Scan the entire tape right-to-left, marking the first unmarked 0 or 1 with an R.

4. Repeat steps 2 and 3 until the left half of the input is marked L and the right half of the input is marked R. During this time, you can also reject if the length of the input is odd.

5. Scan the entire tape left-to-right, blanking the first L-symbol and the first R-symbol if they agree (0^L and 0^R or 1^L and 1^R), or rejecting if they disagree.

6. Repeat step 5 until all input symbols are blanked. If all goes well, then accept. If any error happens, then reject.

A. Draw the $\{ww\} \subseteq \{0,1\}^*$ algorithm just described at Full Detail.

B. At Moderate Detail, do Exercise 3.8b (about having twice as many 0s as 1s).

C. At Full Detail, draw out your Turing machine from Exercise 3.8b.

D. Let $\Sigma = \{(,)\}$ be the alphabet consisting of just parentheses, and let A be the language over Σ consisting of valid parenthesis nests. (We have already learned that A is context-free and not regular.) At Moderate Detail, describe a Turing machine whose language is A.

Problem E below is optional. You do not need to hand it in. But you might consider doing it as part of your studying.

E. Using our current conception of Turing machines — one-tape, deterministic — explain how Problem D above generalizes to any PDA that happens to be deterministic. (Once we study non-deterministic Turing machines, your argument should further generalize, showing that for any PDA there is an equivalent NTM. Hence every context-free language is decidable.)