CSE 103	Computational Models	
Fall 2023	Prof. Fremont	HW 6

Due November 21 at 11:59pm

(3 questions, 280 points total)

1. **(100 pts.) Working with a TM**
Consider the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- $Q = \{\textsc{Start}, \textsc{SeenB}, \textsc{Seen\$}, \textsc{Matched}, \textsc{Accept}, \textsc{Reject}\}$
- $\Sigma = \{b, \$\}$
- $\Gamma = \{b, \hat{b}, \$, \sqcup\}$
- $\delta(q,s) = \begin{cases} (\textsc{Accept}, s, R) & q = \textsc{Start} \text{ and } s = \$ \\ (\textsc{SeenB}, \hat{b}, R) & q = \textsc{Start} \text{ and } s = b \\ (\textsc{Seen\$}, s, R) & q = \textsc{SeenB} \text{ and } s = \$ \\ (q, s, R) & q = \textsc{SeenB} \text{ and } s = b \\ (\textsc{Matched}, \hat{b}, L) & q = \textsc{Seen\$} \text{ and } s = b \\ (\textsc{Start}, s, R) & q = \textsc{Matched} \text{ and } s = \sqcup \\ (q, s, L) & q = \textsc{Matched} \text{ and } s \in \{b, \hat{b}, \$\} \\ (q, s, R) & s = \hat{b} \\ (\textsc{Reject}, s, R) & \text{otherwise} \end{cases}$
- $q_0 = \textsc{Start}$
- $q_{\text{accept}} = \textsc{Accept}$
- $q_{\text{reject}} = \textsc{Reject}$

(a) (*45 pts.*) Draw M as a graph, omitting the reject state and all transitions leading to it (following the convention used in class, we will say that all missing transitions lead to the reject state).

(b) (*15 pts.*) List the sequence of configurations of M (the computation history) when run on input $b\$bb$. Please put each configuration on a separate line.

(c) (*20 pts.*) For each of the following input strings, state whether M accepts, rejects, or does not halt.

- ε
- $bb\$b$
- $bb\$bbb$
- $\$b$

(d) (*20 pts.*) What is the language of M?

2. (80 pts.) Fleshing out a TM description

Consider the following language over $\Sigma = \{0, 1, \#, \$\}$, which represents a simple form of array lookup, namely checking if the kth element of the array $(e_0, e_1, \ldots, e_n)$ is equal to a given binary string v:

$$L = \{1^k \,\#\, v \,\$\, e_0 \,\$\, e_1 \,\$\, \ldots \$\, e_n \,\$ \mid 0 \le k \le n,\ \ v, e_0, \ldots, e_n \in \{0,1\}^*, \text{ and } e_k = v\}.$$

Here are some example strings (putting extra space between the sections of the string for readability, and bolding the sections that have to match):

$$11 \,\#\, \mathbf{011} \,\$\, 0 \,\$\, 100 \,\$\, \mathbf{011} \,\$\, 11 \,\$\ \in L$$
$$11 \,\#\, \mathbf{011} \,\$\, 0 \,\$\, 100 \,\$\, \mathbf{010} \,\$\, 11 \,\$\ \notin L$$
$$111 \,\#\, \mathbf{11} \,\$\, 0 \,\$\, 100 \,\$\, 010 \,\$\, \mathbf{11} \,\$\ \in L$$

Consider the following medium-level description (what the textbook calls an "implementation description") of a TM deciding L, which uses the tape alphabet $\Gamma = \{0, 1, \#, \$, X, \sqcup\}$ where X is a new symbol representing a "crossed-off" part of the tape:

1. Go through each of the 1s at the start of the input; for each one, cross it off and scan right until you find a \$, and cross that off as well, rejecting if there are no \$s left. (After this step, the leftmost remaining \$ will be the one just before e_k.)
2. Go through each of the symbols of v; for each one, remember whether it is a 0 or a 1, then scan right until you find a \$. Scan right until you find a 0 or 1, and reject if it doesn't agree with the symbol of v we saw earlier.
3. Go back to the #, then move right to the first \$. Continue moving right, and if you see another \$ without encountering any 0s or 1s along the way, then accept.

Let's flesh out this description into a more detailed one listing all the states and what should be done at each one. Use English, like the low-level description given in the 11/13 lecture (on the last page of the notes): *do not write out the TM as a graph.*

(a) (*20* pts.) Describe how to implement step (1) above. You should only need 3 states, but don't worry about having exactly this number.

(b) (*40* pts.) Describe how to implement step (2) above, assuming the head of the TM is already at the first symbol of v. It's possible to do this with 6 states.

(*Hint:* If you're having trouble figuring this out, Example 3.9 in the textbook may be helpful.)

(c) (*20* pts.) Describe how to implement step (3) above. It's possible to do this with 4 states.

3. (100 pts.) Designing a TM

Design a TM that can compare two integers represented in binary; more precisely, that recognizes the language L of strings $x\$y$ where $x, y \in \{0,1\}^*$, $|x| = |y|$ (for simplicity), and $x \leq y$ when interpreted as integers. For example, 010\$100 and 001\$010 are in L but 1\$0 and 0\$00 are not. Give a medium-level description of your machine, like the 3-step description in the statement of Problem 2 above (do *not* give a low-level description talking about each state).