I. Turing Machines

A. Design a Turing Machine that decides the language $L_a = \{0^n 1^n \mid n \geq 1\}$.

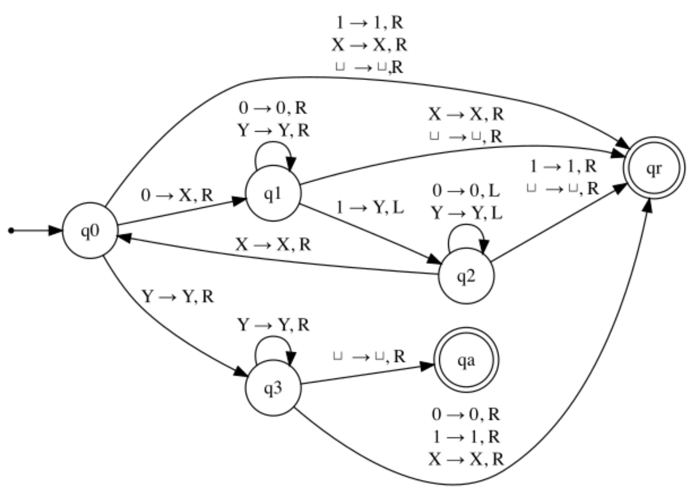
1. Alternately, the TM will change a 0 to an X and then a 1 to a Y until all 0′s and 1′s have been matched.

   In more detail, starting at the left end of the input, the TM enters a loop in which it changes a 0 to an X and moves to the right over whatever 0′s and Y′s it sees, until it comes to a 1. It changes the 1 to a Y and moves left, over Y ′s and 0′s, until it finds an X. At that point, it looks for a 0 immediately to the right, and if it finds one, changes it to X and repeats this process, changing a matching 1 to a Y.

2. $M = \{Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\}$ Where:
   - $Q = \{q_0, q_1, q_2, q_3, q_a, q_r\}$
   - $\Sigma = \{0,1\}$
   - $\Gamma = \{0,1, X, Y, \sqcup\}$
   - $\delta =$

$Q \times \Gamma$	0	1	X	Y	$\sqcup$
q_0	(q_1, X, R)	$(q_r, 1, R)$	(q_r, X, R)	(q_3, Y, R)	$(q_r, \sqcup, R)$
q_1	$(q_1, 0, R)$	(q_2, Y, L)	(q_r, X, R)	(q_1, Y, R)	$(q_r, \sqcup, R)$
q_2	$(q_2, 0, L)$	$(q_r, 1, R)$	(q_0, X, R)	(q_2, Y, L)	$(q_r, \sqcup, R)$
q_3	$(q_r, 0, R)$	$(q_r, 1, R)$	(q_r, X, R)	(q_3, Y, R)	$(q_a, \sqcup, R)$
q_a	-	-	-	-	-
q_r	-	-	-	-	-

   - $q_{accept} = q_a$
   - $q_{reject} = q_r$

3. 

4. $q_0 0011 \rightarrow Xq_1 011 \rightarrow X0q_1 11 \rightarrow Xq_2 0Y1 \rightarrow q_2 X0Y1 \rightarrow Xq_0 0Y1 \rightarrow XXq_1 Y1 \rightarrow XXYq_1 1 \rightarrow XXq_2 YY \rightarrow Xq_2 XYY \rightarrow XXq_0 YY \rightarrow XXYq_3 Y \rightarrow XXYYq_3 \sqcup \rightarrow XXYY\sqcup q_a \sqcup$
5. $q_0 0010 \rightarrow Xq_1 010 \rightarrow X0q_1 10 \rightarrow Xq_2 0Y0 \rightarrow q_2 X0Y0 \rightarrow Xq_0 0Y0 \rightarrow XXq_1 Y0 \rightarrow XXYq_1 0 \rightarrow XXY0q_1 \sqcup \rightarrow XXY0\sqcup q_r \sqcup$

B. Describe a Turing Machine that decides the language
$L_b = \{w \in \{0,1\} * \mid |w|_0 = |w|_1 \}$
where $|w|_0$ and $|w|_1$ are respectively the number of 0's and 1's in w.

1. Alternately, the TM will change a 0 to an X and then a 1 to a Y until all 0′s and 1′s have been matched.

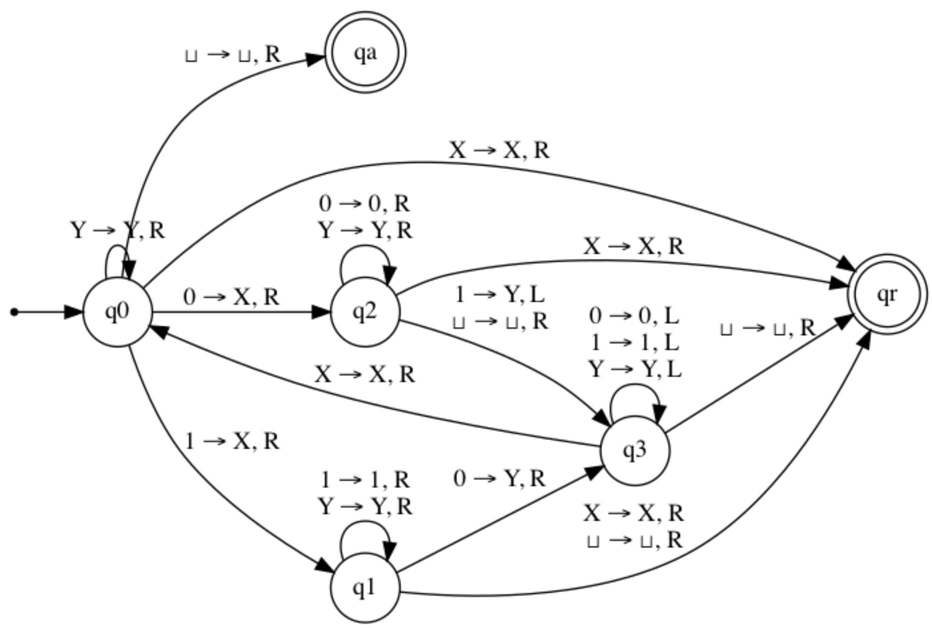
   In more detail, starting at the left end of the input, the TM enters a loop in which it changes a 0 to an X and moves to the right over whatever 0′s and Y′s it sees, until it comes to a 1. It changes the 1 to a Y and moves left, over Y ′s and 0′s, until it finds an X. At that point, it looks for a 0 immediately to the right, and if it finds one, changes it to X and repeats this process, changing a matching 1 to a Y.

2. $M = \{Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\}$ Where:
   - $Q = \{q_0, q_1, q_2, q_3, q_a, q_r\}$
   - $\Sigma = \{0,1\}$
   - $\Gamma = \{0,1, X, Y, ⊔\}$
   - $\delta =$

$Q \times \Gamma$	0	1	X	Y	⊔
q_0	(q_2, X, R)	(q_1, X, R)	(q_r, X, R)	(q_0, Y, R)	$(q_a, ⊔, R)$
q_1	(q_3, Y, R)	$(q_1, 1, R)$	(q_r, X, R)	(q_1, Y, R)	$(q_r, ⊔, R)$
q_2	$(q_2, 0, R)$	(q_3, Y, L)	(q_r, X, R)	(q_2, Y, R)	$(q_r, ⊔, R)$
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, X, R)	(q_3, Y, L)	$(q_r, ⊔, R)$
q_a	-	-	-		-
q_r	-	-	-		-

   - $q_{accept} = q_a$
   - $q_{reject} = q_r$

3.



C. How would one simulate a PDA on a Turing machine? Please do not write the Turing machine itself, but rather write the key idea in plain English.

We create the stack of the PDA as an additional section of the tape following the input. We place a delimiter character after the input to define the start of the stack. The states of the TM basically represent the states of the PDA. In addition, we need some auxiliary states (see below). The TM works as follows: Whenever the PDA would read a symbol, the TM must read the symbol under the head on the input section of the tape and moves the head one step to the right. Whenever the PDA uses the empty symbol as input symbol, the tape keeps its position. The machine then positions to the "top" of the stack section of the tape:

- If the PDA consumes a symbol from the stack but does not write a new one, the TM deletes the stack symbol under the head and moves the head one position to the left.
- If the PDA consumes a symbol from the stack and writes a new one, the TM overwrites the symbol below the head with the new one.
- If the PDA does not consume a symbol from the stack but writes a new one, the TM moves one step to the right and writes the new symbol. For this we actually need to perform two steps which can be done using some auxiliary states (we need to remember the symbol to write and the current state of the TM).
- If the PDA does not change the stack, the TM does not stack portion of the tape.

This TM accepts the input of the PDA iff it stops in an accept state (if there are several accepting states in the PDA, we use an additional transition that brings the TM from these states to its single accept state).

D. Design and explain informally, a non-deterministic Turing machine for the language $L_d = \{ww \mid w \in \{0,1\}*\}$.

Given an input string of length *n*, there are *n-1* possible ways to split this string into two substrings w_l and w_r such that w_l and $w_r \neq \emptyset$:

1. Non-deterministically guess a split point dividing the input into w_l and w_r.
2. Deterministically compare w_l and w_r.
3. If w_l and w_r are equivalent → accept, else reject