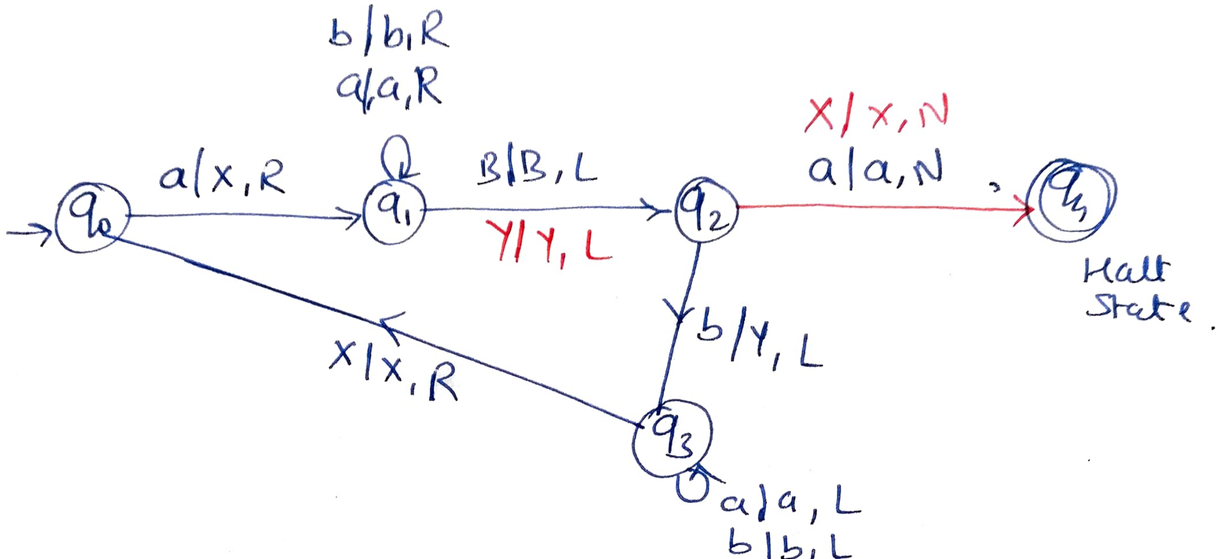
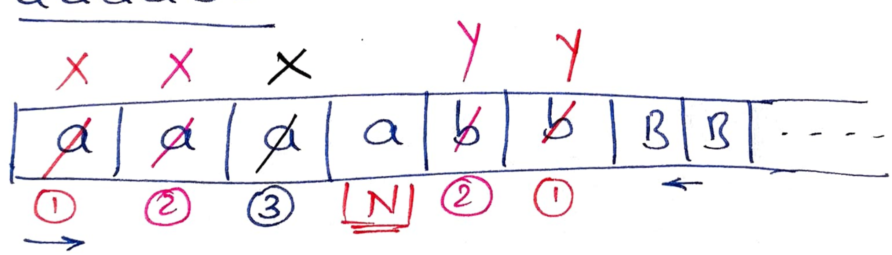
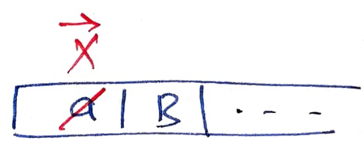
(5) $L = \{a^n b^m \mid n > m\}$



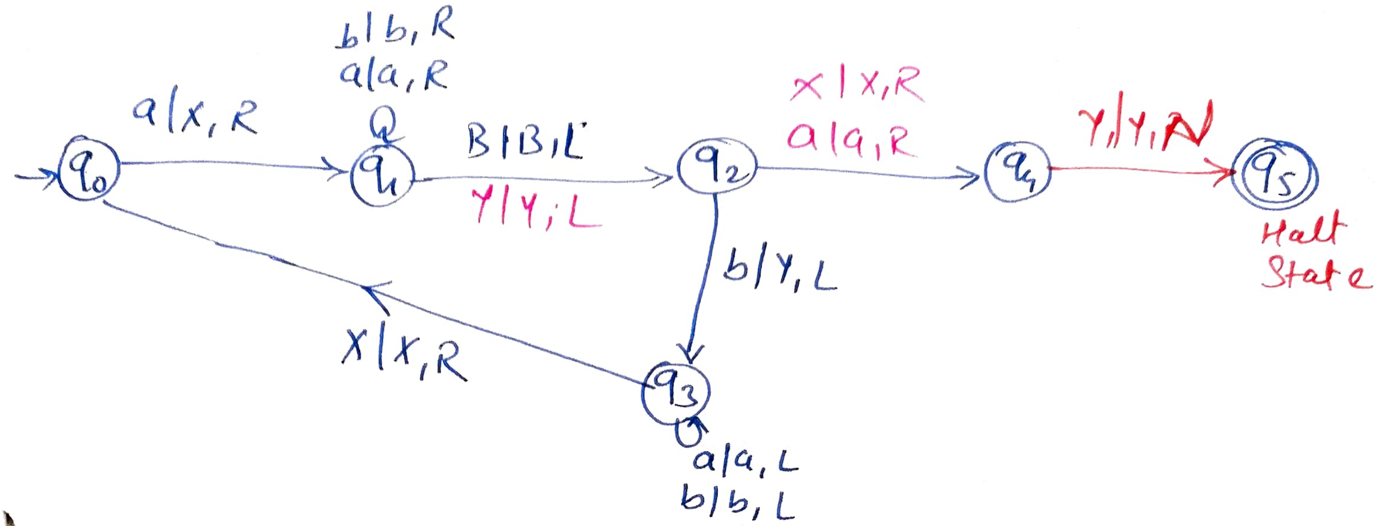
(1) w = aaaabb



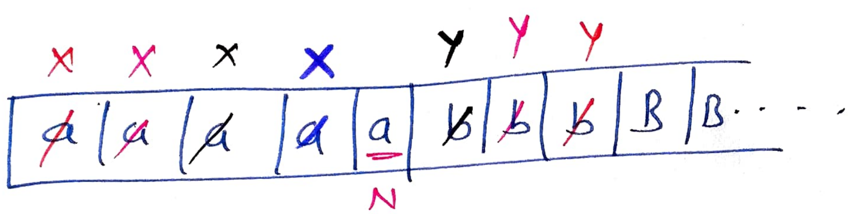
(2) w = a



(6) $L = \{a^n b^m \mid n > m ;\ n, m \geq 1\}$

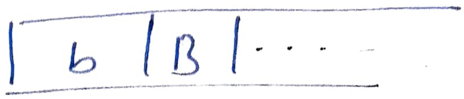


w = aaaaabbb

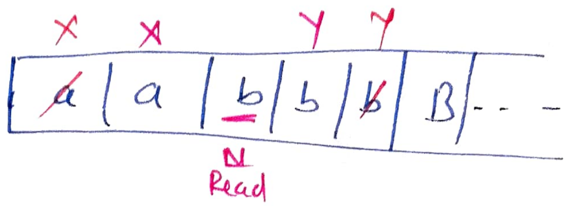


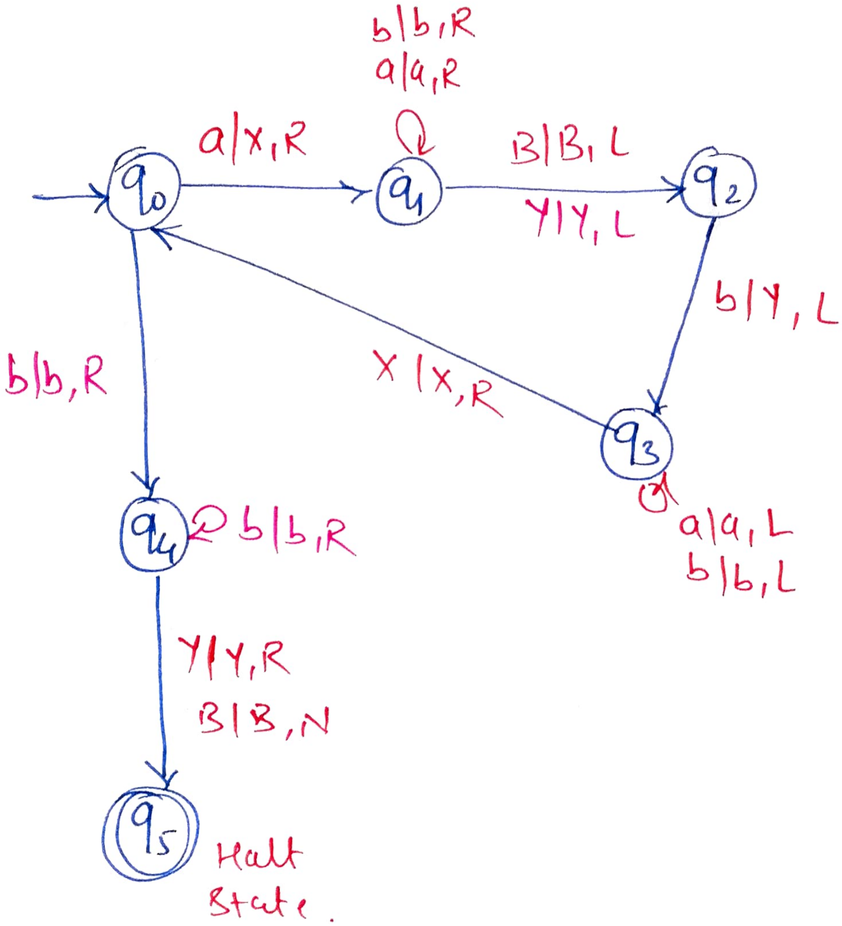
⑦ $L = \{a^n b^m \mid n < m\}$

$\Rightarrow$ $a = 0, b = 1$



$a = 2, b = 3$





$M = \{Q, \Sigma, \delta, q_0, F, \Gamma, B\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$; $F = \{q_5\}$

$\Sigma = \{a, b\}$; $\Gamma = \{a, b, X, Y, B\}$

Processing Sequence for w = aabbb

$\underset{\uparrow q_0}{a}abbbB \vdash X\underset{\uparrow q_1}{a}bbbB \vdash Xa\underset{\uparrow q_1}{b}bbB \vdash Xab\underset{\uparrow q_1}{b}bB$

$\vdash Xabb\underset{\uparrow q_1}{b}B \vdash Xabbb\underset{\uparrow q_1}{B} \vdash Xabb\underset{\uparrow q_2}{b}B \vdash Xab\underset{\uparrow q_3}{b}YB \vdash$

$Xa\underset{\uparrow q_3}{b}bYB \vdash X\underset{\uparrow q_3}{a}bbYB \vdash \underset{\uparrow q_3}{X}abbYB \vdash X\underset{\uparrow q_0}{a}bbYB \vdash$

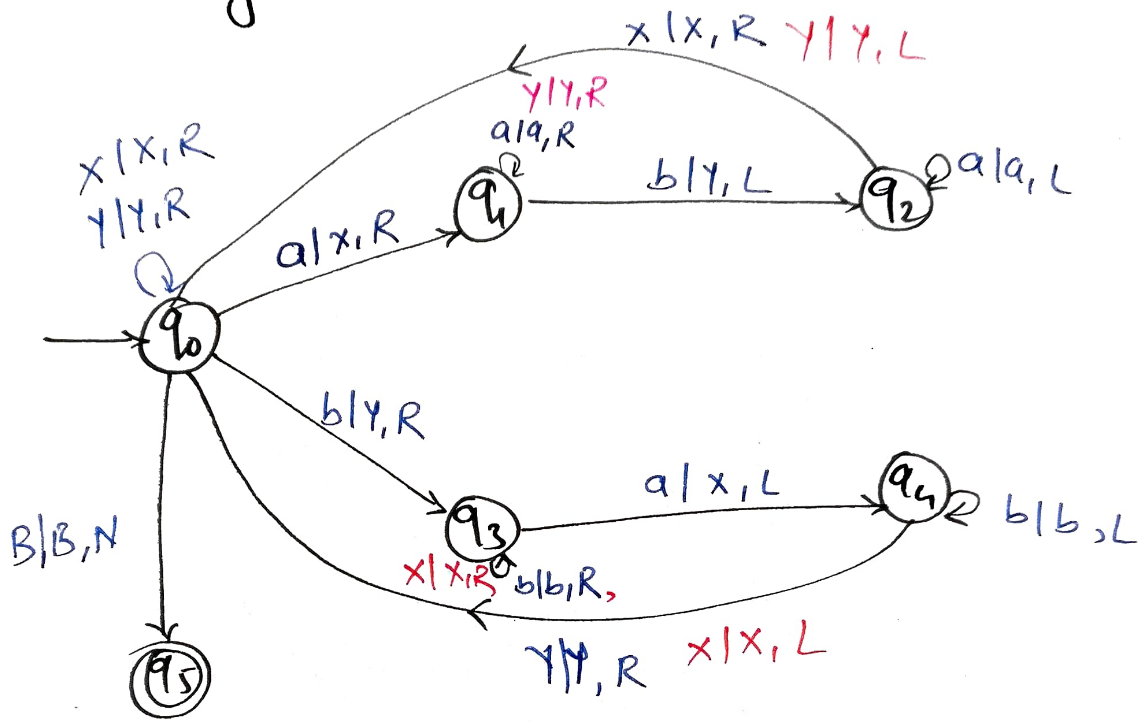
$XX\underset{\uparrow}{b}bYB \vdash XXb\underset{\uparrow}{b}YB \vdash XXbb\underset{\uparrow q_1}{Y}B \vdash XXb\underset{\uparrow q_2}{b}YB \vdash$

$XX\underset{\uparrow q_3}{b}YYB \vdash X\underset{\uparrow q_3}{X}bYYB \vdash XX\underset{\uparrow q_4}{b}YYB \vdash XXbY\underset{\uparrow q_5}{Y}B$

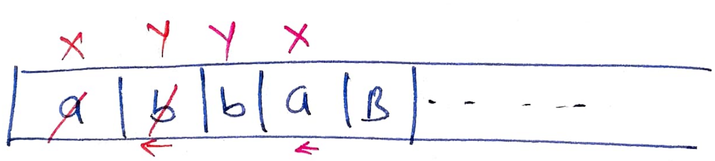
$\vdash XXbYY\underset{\uparrow q_5}{B}$ Accept

(8) Design a TM to check whether a string over {a, b}, contains equal no. of a's & b's.

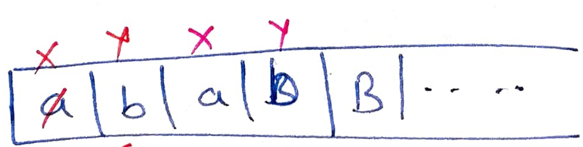
⇒ Initially a (or) b can be there.



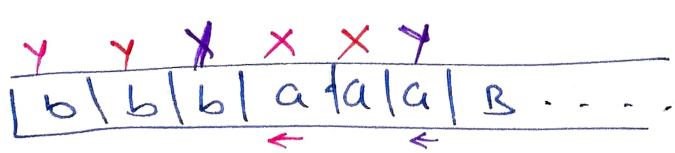
(1)



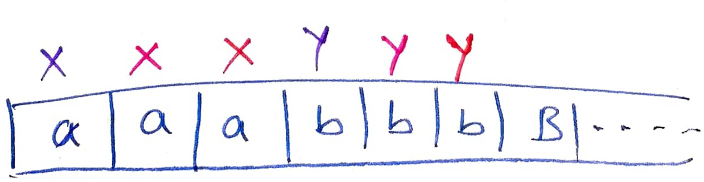
(2)



(3)

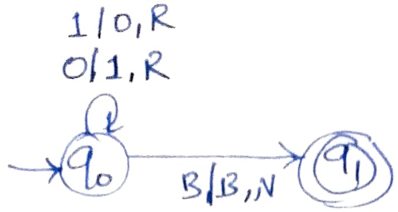


(4)



9. Construct a TM for 1's complement.

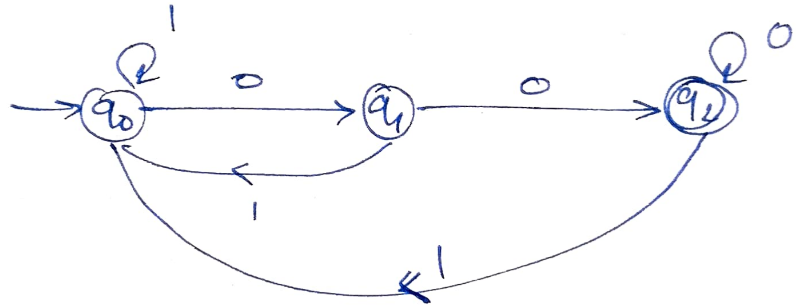
⇒



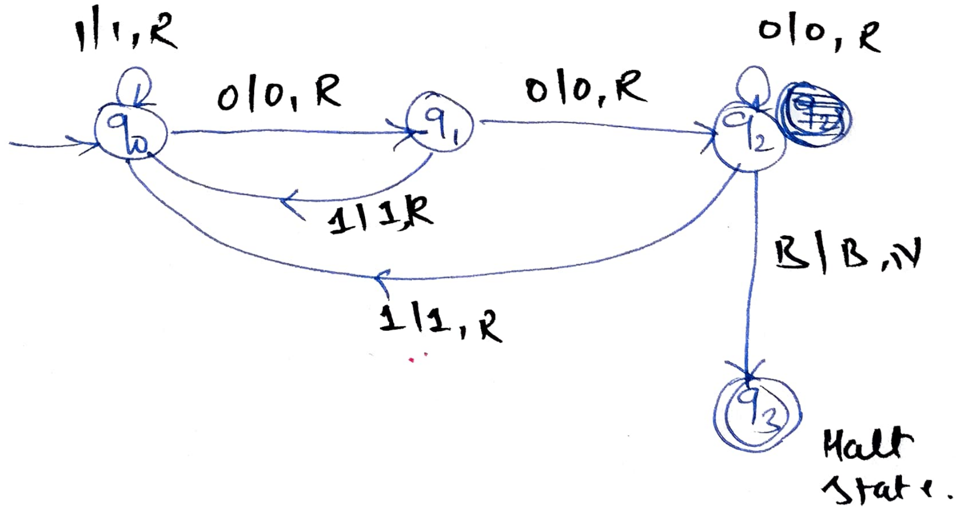
10. Construct a TM that recognizes the language

$L = \{ x \in \{0,1\}^* \mid x \text{ ends in } 00 \}$

⇒ DFA ⇒

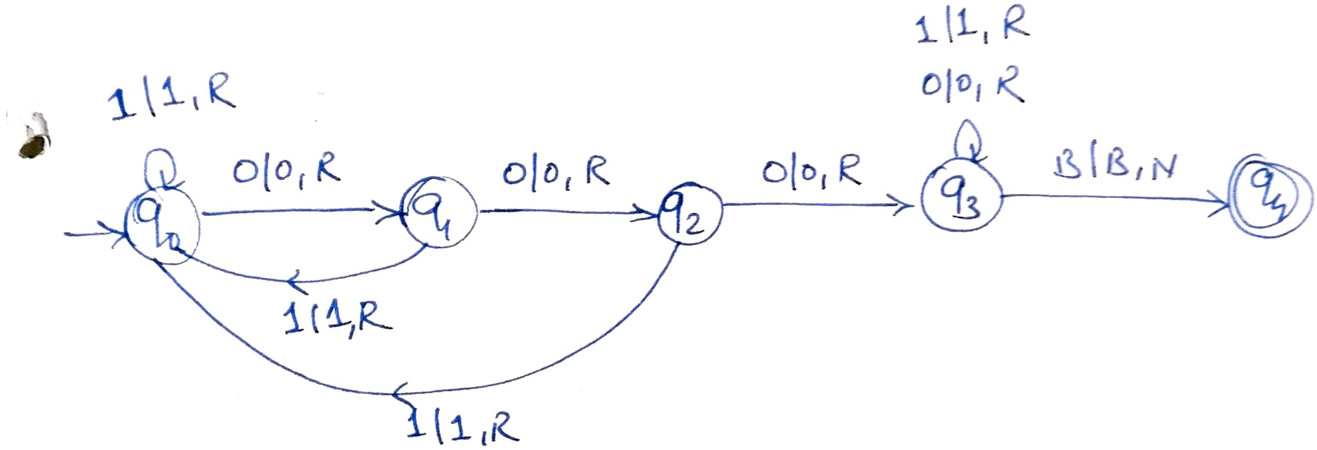


TM



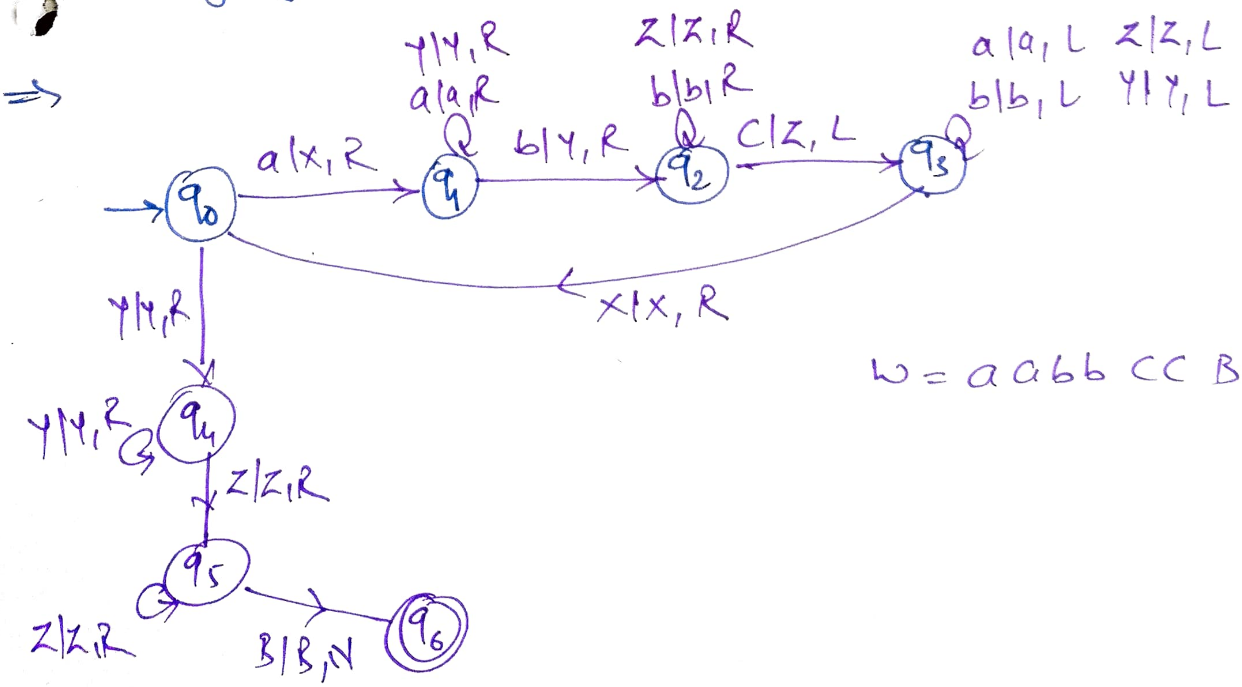
(11) Design a TM to accept strings formed with 0 & 1 that have the substring 000.

→ To accept strings with substrings 000, the TM would be similar to the FA constructed for same language.



(12) Design a TM for accepting strings of a language $L = \{a^n b^n c^n \mid n \geq 1\}$

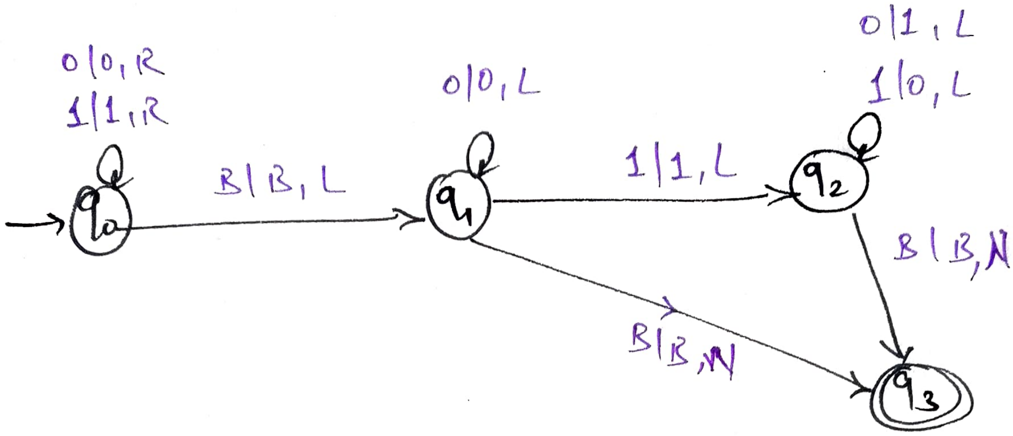
⇒



w = aabb CC B

(12) Construct TM for 2's complement of binary no.

⇒



(1) | B | 0 | B | ... ⇒ 2's ⇒ 0

(2) | B | 1 | B | ... ⇒ 2's ⇒ 1

(3) | B | 1 | 0 | 1 | 1 | B | ⇒ 2's ⇒ 0 1 0 1 B

0 1 0 1 B

1011
0100 ⇒ 1's
0101 ⇒ 2's

(13) Design a TM to make a copy of string over $\{0, 1\}$

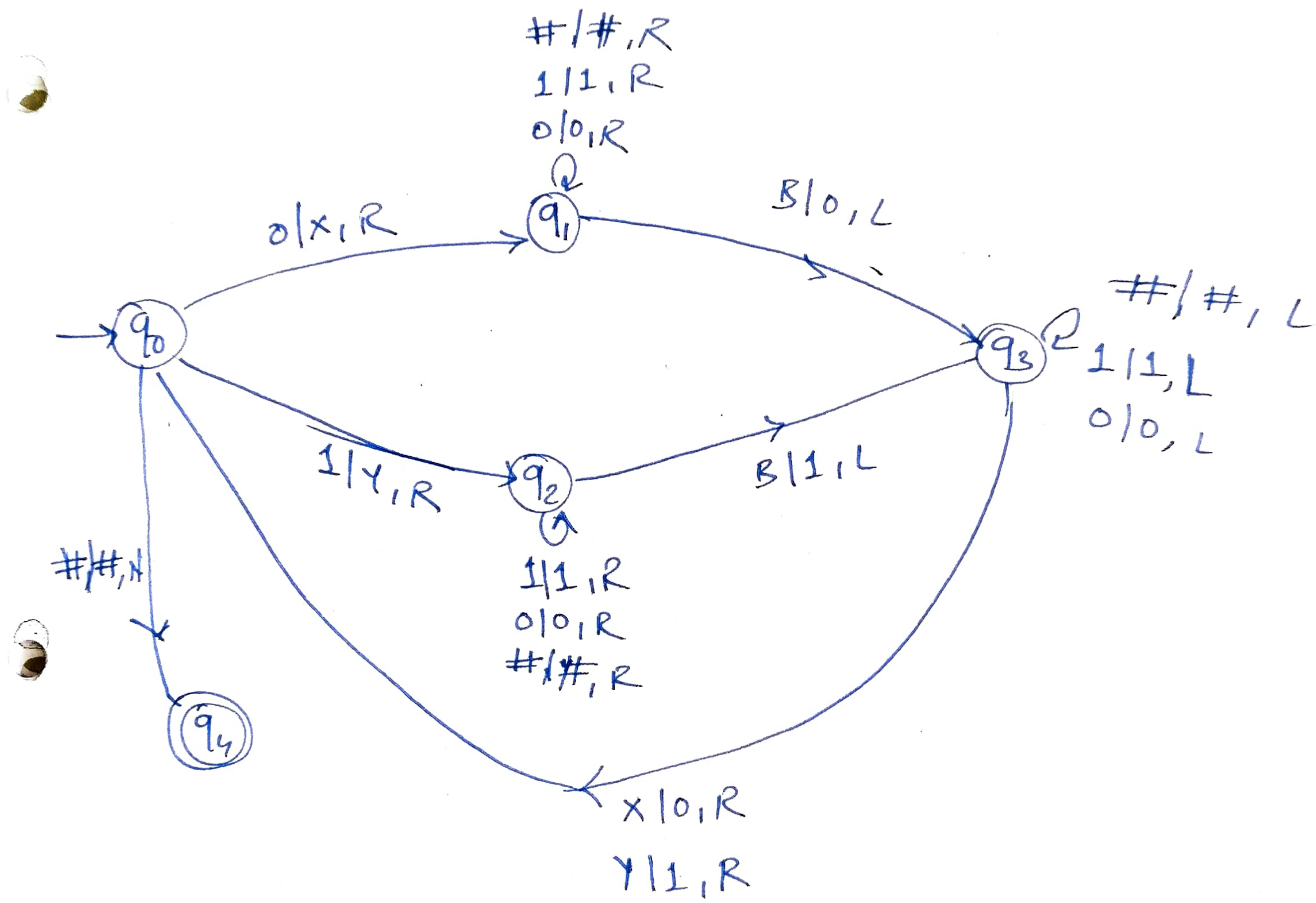
⇒ I/P:

B	1	1	0	0	#	B	...

O/P:

B	1	1	0	0	#	1	1	0	0	B	...

⇒ Two copies are seperated by #



⇒ First input & Blank 'B' should be updated then again update the X & Y inputs.

B	0	1	1	#	B

B	X	1	1	#	0

X	1	1	#	0	B

0	1	1	#	0	B

0	Y	1	#	0	1	B

0	1	Y	#	0	1	1	B

0	1	1	#	0	1	1	B

(14) Construct a TM for checking well formness of parenthesis.

⇒ To solve this, we need to match every occurence of "(" for every occurence of ")". At the end if any parenthesis is unmatched then the given string is declared not balanced.

(1) First search for the occurence of ")", for this process, in the initial state q_0 ignore all "(" until ")" is seen.

$$\delta(q_0, () = (q_0, (, R)$$

(2) On the occurence / finding ")" replace it by 'X' change to new state & travel left for the first occurence of "(". It is used to find "(" for ")" while travelling back it can see 'X'.

$$\delta(q_0,)) = (q_1, X, L)$$

$$\delta(q_1, X) = (q_1, X, L)$$

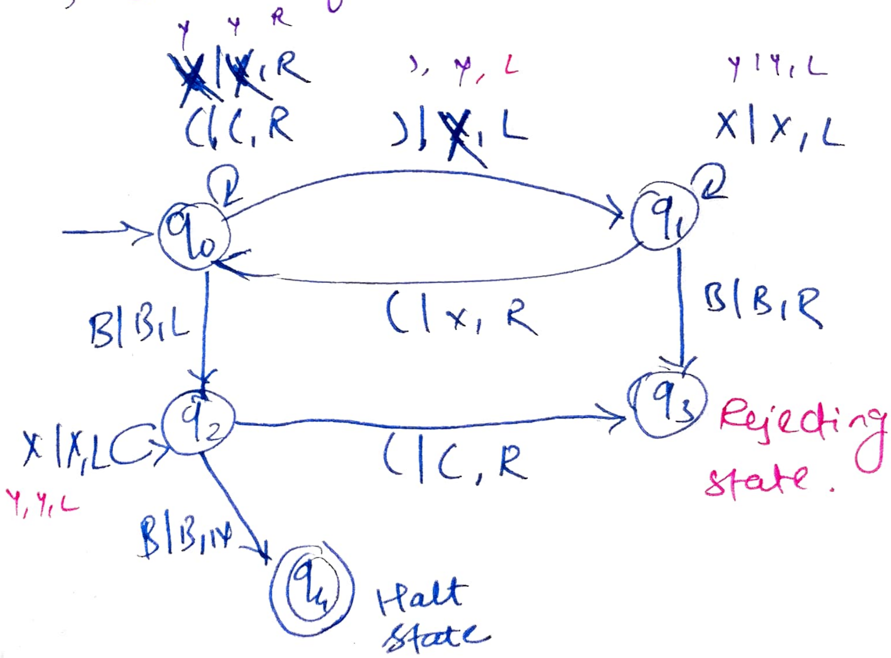
③ If "(" is found, replace it by 'X', If X is not found, enter into rejecting state. In this for ex. q_1 acts as both initial state & return state

$$\delta(q_1, ( ) = (q_0, X, R)$$
$$\delta(q_1, B) = (q_3, B, R)$$

④ Repeat step 1 & 2 until a B is encountered.

$$\delta(q_0, X) = (q_0, X, R)$$
$$\delta(q_0, B) = (q_2, B, L)$$

⑤ If B is encountered enter into a new state & check if is no "(" unbalanced.



① (B
q_0 q_2
q_3

② B) B
q_0 q_1
B X
q_3

(( )) ( ) B

q_0 q_0

(( )) ( ) B

q_0

((X) ( ) B

q_1

(X X) ( ) B

q_0

(X X) ( ) B

q_0

(X X X ( ) B

q_1 q_1 q_1

X X X X ( ) B

q_0 q_0 q_0

X X X X ( ) B

q_0 q_0

X X X X (X B

q_1 q_0

B X X X X X X B

q_0

B X X X X X X B

q_4 q_2 q_2 q_2 q_2 q_2 q_2