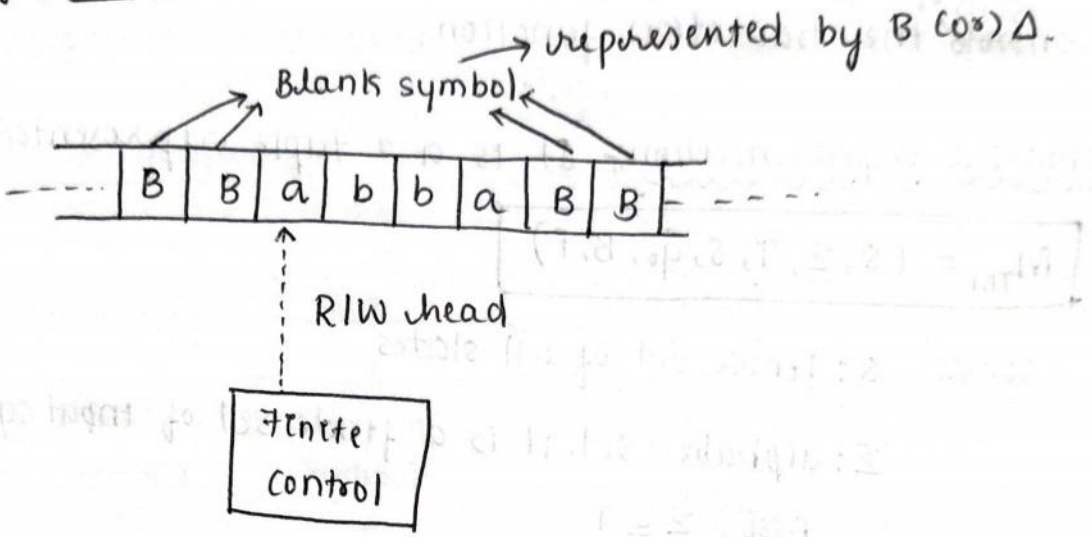
Imp***

Turing machine:- It is the powerful automata.

Working principle:-



Turing machine consists of 3 components:

i) Input tape

ii) Read/Write head

iii) Finite control.

Input tape:-

*) It is an infinite length tape.

*) Divided into cells.

*) Each cell can contain only one symbol.

*) Here in turing machine, cells can have ~~sy~~blank symbol (can be represented by B) on either side of string.

Read/Write head:-

*) It always points to the left most side (symbol) of the given string.

*) It can only move one cell at a time.

*) It can replace the existing symbol with a new symbol.

*) It can move/traverse either from left to right (or) right

to left.

Finite control:-

*) It controls the read/write head.

*) It consists of the transition function.

Mathematical representation:- It is a 7 tuple representation.

$$M_{TM} = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

where Q: finite set of all states

Σ: alphabet set, it is a finite set of input symbols.

Also, $\Sigma \subseteq \Gamma$

Γ: Tape symbol, it is a finite set

q_0: Initial state, $q_0 \in Q$

B: Blank symbol, $B \in \Gamma$

F: Finite set of final states, $F \in Q$

δ: Transition function defined as

$$(Q \times \Gamma) = (Q \times \Gamma \times D)$$

where D: direction ie., R (or) L

↑ (move right) ↑ (move left)

Imp *** (10m).

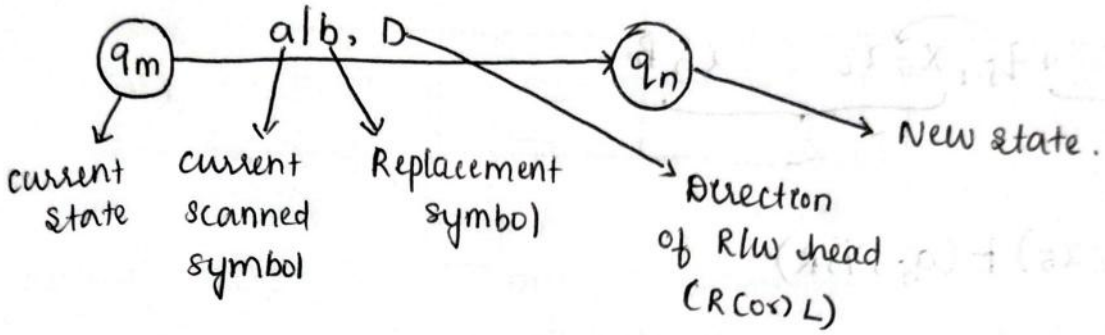
Representation of Turing machine:-

i) Transition diagram

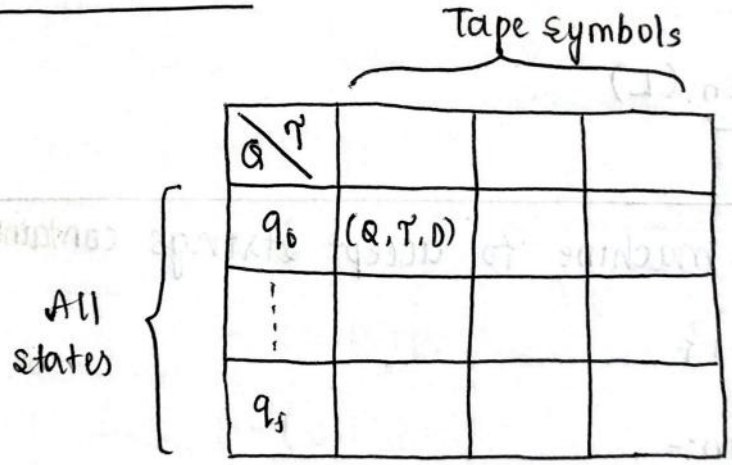
ii) Transition table

iii) Instanteneous description.

Transition diagram:-



Transition table:-



In (Q, T, D)

Q: new state

T: Replacement symbol

D: R (or) L

Instanteneous description:-

"⊢" (turnstile) <fallen "T">

$(q_0, w) \vdash^* \alpha_1 q_f \alpha_2$

part of string that is already scanned — current state (final state) — part of the string to be scanned.

$w = x_1 x_2 x_3 \ldots x_n B$

$\underbrace{x_1 x_2 x_3 x_4}_{\alpha_1} q_{f-1} \underbrace{x_5 x_6 \ldots x_n B}_{\alpha_2}$

$\delta(q_{f-1}, x_5) \vdash (q_f, Y, R)$

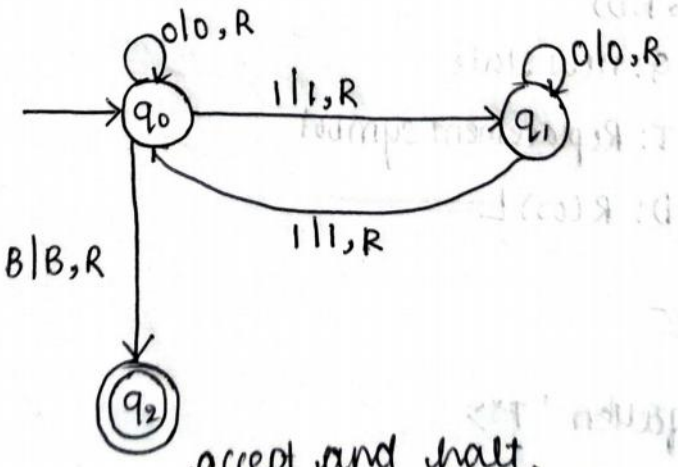
$\underbrace{x_1 x_2 x_3 x_4}_{\alpha_1} q_f \underbrace{x_6 \ldots x_n B}_{\alpha_2}$ (R)

$\underbrace{x_1 x_2 x_3 q_f x_4}_{\alpha_1} \underbrace{Y x_6 \ldots x_n}_{\alpha_2}$ (L)

---

1) Construct a turing machine to accept strings containing even no of 1's over $\{0,1\}$.

Sol:- Transition diagram:-



accept and halt.

Transition table:-

Q \ T	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	(q_2, B, R)
q_1	$(q_1, 0, R)$	$(q_0, 1, R)$	-
q_2	-	-	-

Instantaneous description:-

consider "0110"

$\delta(q_0, 0110) \vdash (0q_0110B)$

$\vdash (01q_110B)$

$\vdash (011q_00B)$

$\vdash (0110q_0B)$

$\vdash (0110Bq_2B)$

accept and halt

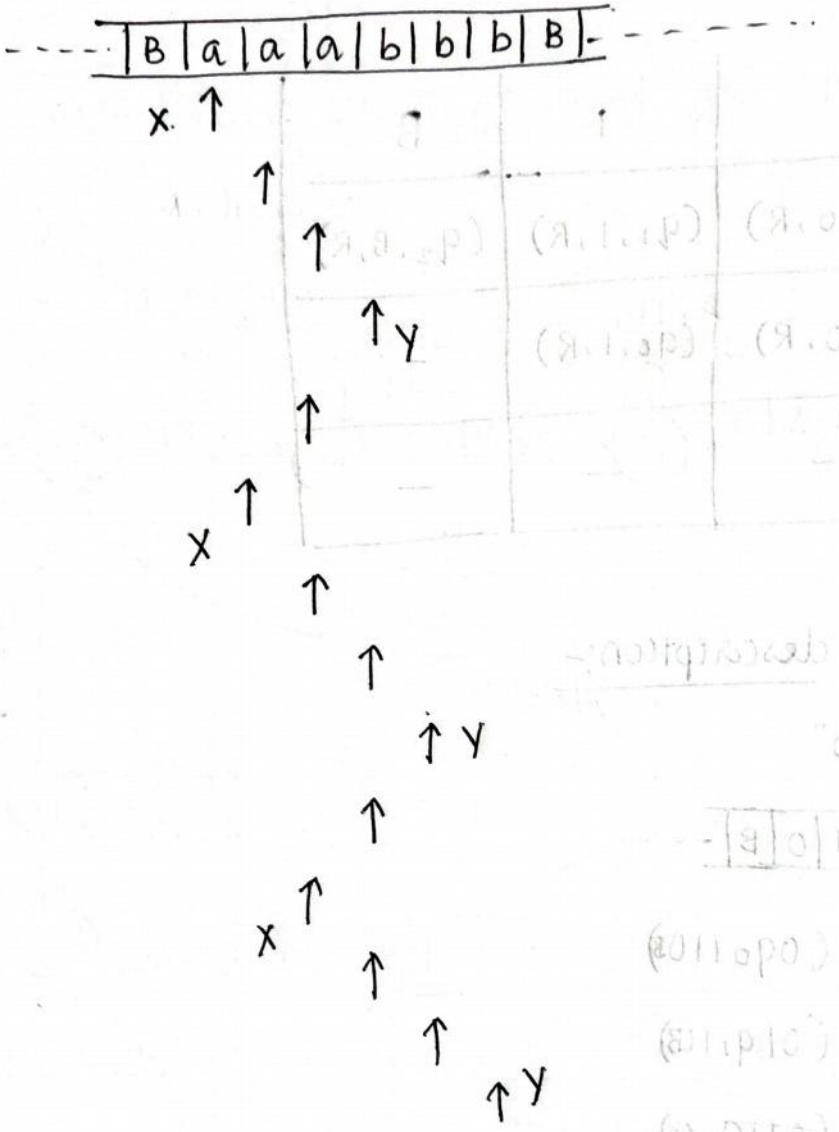
---

2). Given,

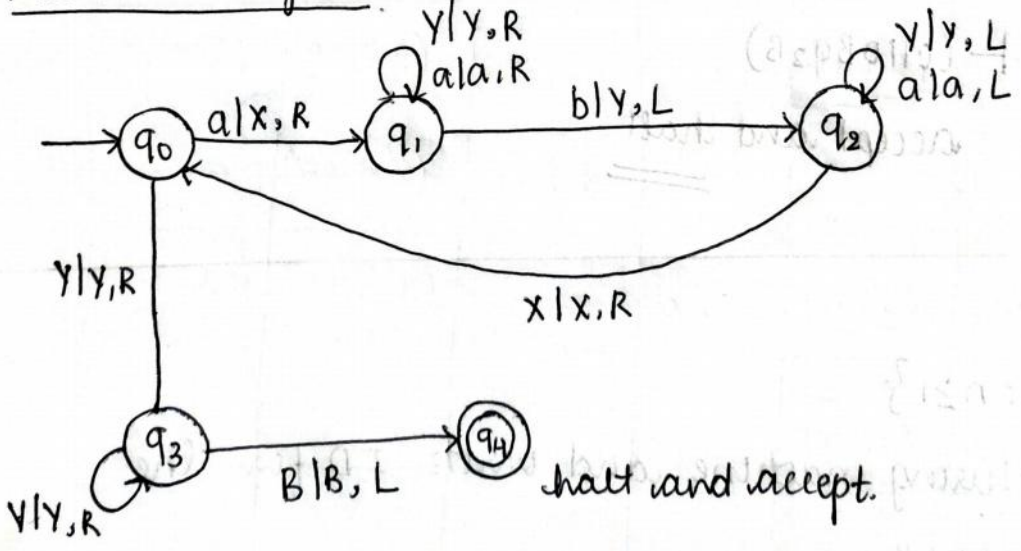
$L = \{a^n b^n : n \geq 1\}$

construct turing machine and write ID for the string "aaabbb"

Sol:- Logic:-



Transition diagram:-



Transition table:-

Q \ T	a	b	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	(q_1, a, R)	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	(q_2, a, L)	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, L)
q_4	-	-	-	-	-

Instantaneous description:-

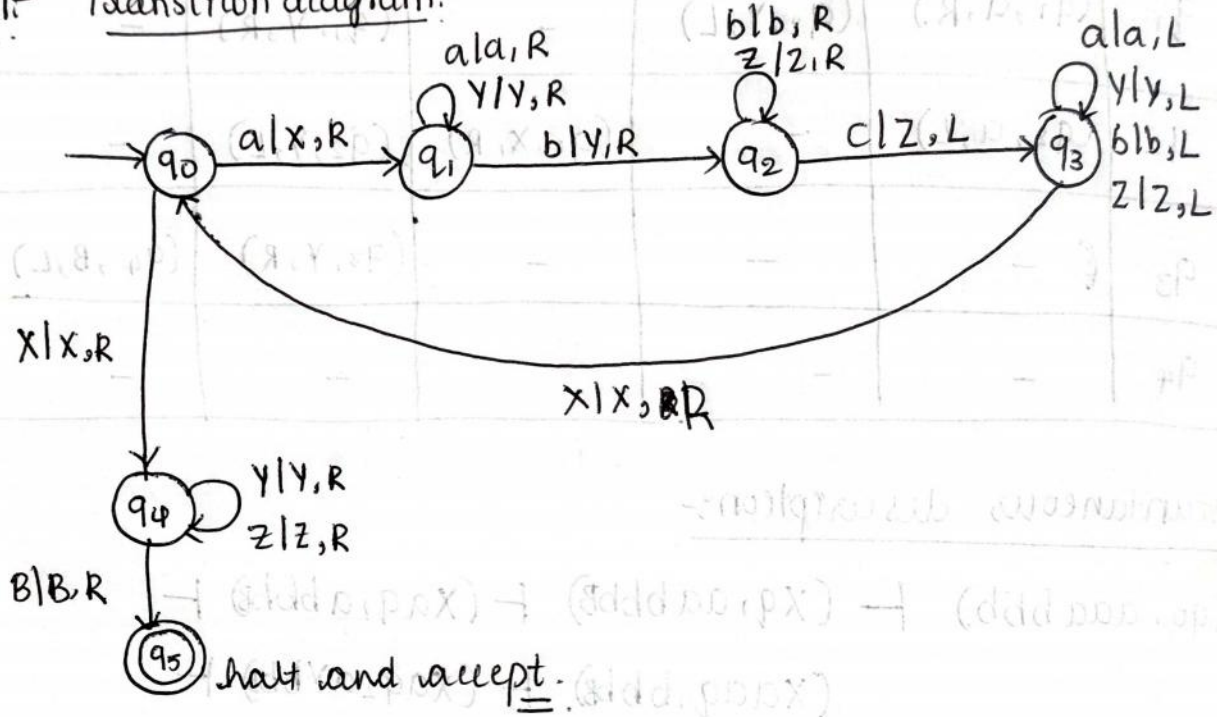
$\delta(q_0, aaabbb) \vdash (Xq_1 aabbbB) \vdash (Xaq_1 abbbB) \vdash$
$(Xaaq_1 bbbB) \vdash (Xaq_2 aYbbB) \vdash$
$(Xq_2 aaYbbB) \vdash (q_2 XaaYbbB) \vdash$
$(Xq_0 aaYbbB) \vdash (XXq_1 aYbbB) \vdash$
$(XXaq_1 YbbB) \vdash (XXaYq_1 bbB) \vdash$
$(XXaq_2 YYbB) \vdash (XXq_2 aYYbB) \vdash$
$(Xq_2 XaYYbB) \vdash (XXq_0 aYYbB) \vdash$
$(XXXq_1 YYbB) \vdash (XXXYq_2 YB) \vdash$
$(XXXq_2 YYYB) \vdash (XXq_2 XYYYB) \vdash$
$(XXXq_0 YYYB) \vdash (XXXYq_3 YY) \vdash$
$(XXXYYq_3 YB) \vdash (XXXYYYq_3 B)$
$(XXXYYYBq_4 B)$ halt and accept

3). Given,

$L = \{a^n b^n c^n : n \geq 1\}$

construct turing machine and give IP for the string "aabbcc"

Sol:- Transition diagram:-



Transition table:

Q \ T	a	b	c	X	Y	Z	B
q_0	(q_1, X, R)	–	–	–	(q_4, Y, R)	–	–
q_1	(q_1, a, R)	(q_2, Y, R)	–	–	(q_1, Y, R)	–	–
q_2	–	(q_2, b, R)	(q_3, Z, L)	–	–	(q_2, Z, R)	–
q_3	(q_3, a, L)	(q_3, b, L)	–	(q_0, X, R)	(q_3, Y, L)	(q_3, Z, L)	–
q_4	–	–	–		(q_4, Y, R)	(q_4, Z, R)	(q_5, B, R)
q_5	–	–	–		–	–	–

Instantaneous description:

Consider, "aabbcc"

$\delta(q_0, aabbcc) \vdash (Xq_1abbccB) \vdash (Xaq_1bbcc)$

$\vdash (XaYq_2bcc) \vdash (XaYbq_2cc)$

$\vdash (XaYq_3bZc) \vdash (Xaq_3YbZc)$

$\vdash (Xq_3aYbZc) \vdash (q_3XaYbZc)$

$\vdash (Xq_0aYbZc) \vdash (XXq_1YbZc)$

$\vdash (XXYq_1bZc) \vdash (XXYYq_2Zc)$

$\vdash (XXYYZq_2c) \vdash (XXYYq_3ZZ)$

$\vdash (XXYq_3YZZ) \vdash (XXq_3YYZZ)$

$\vdash (Xq_3XYYZZ) \vdash (q_0XXYYZZ)$

$\vdash (Xq_0XYYZZ) \vdash (XXq_4YYZZ)$

$\vdash (XXYq_4YZZ) \vdash (XXYYq_4ZZ)$

$\vdash (XXYYZq_4Z) \vdash (XXYYZZq_4B)$

$\vdash (XXYYZZBq_4)$

accept and halt.

29/11/2024

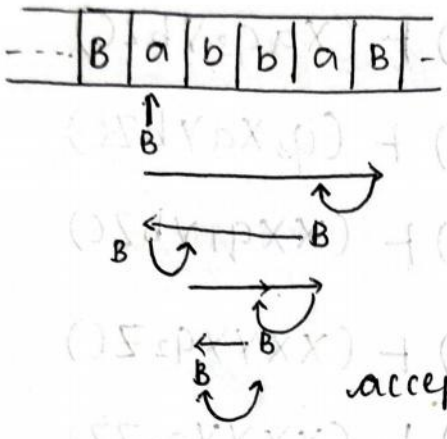
4) Construct turing machine that accepts even length palendrome. (or) $L = \{ww^R \mid \in (a+b)^*\}$

Sol:-

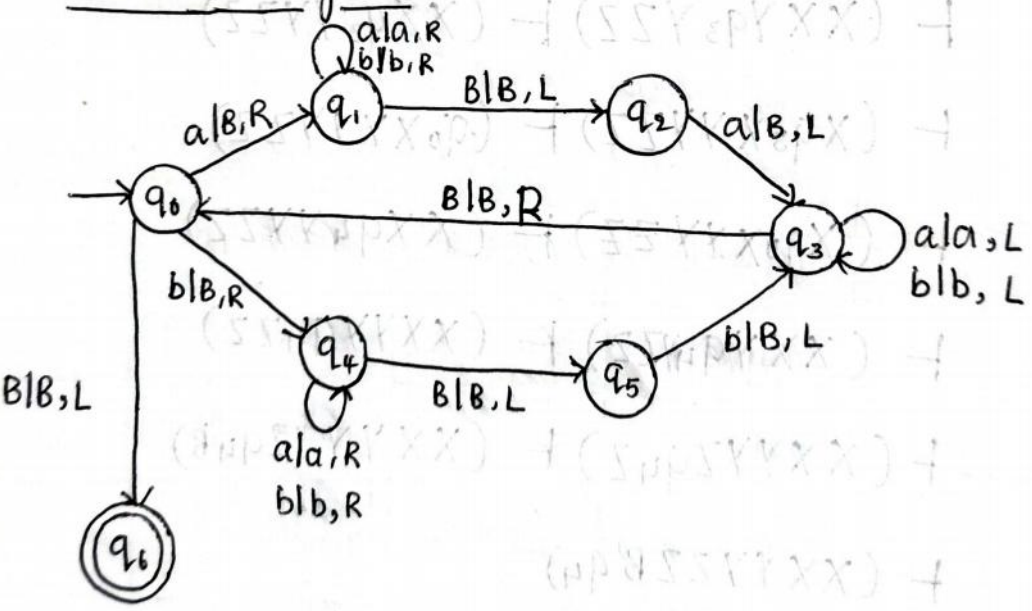
consider $w = ab$

$w^R = ba$

$ww^R = abba$



Transition diagram:-



Halt and accept.

Transition table:-

Q \ T	a	b	B
q_0	(q_1, B, R)	(q_4, B, L)	(q_6, B, L)
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, L)
q_2	(q_3, B, L)	–	–
q_3	(q_3, a, L)	(q_3, b, L)	(q_0, B, R)
q_4	(q_4, a, R)	(q_4, b, R)	(q_5, B, L)
q_5	–	(q_6, B, L)	–
q_6	–	–	–

Instantaneous description:-

consider "abba"

~~$\delta(q_0, abba) \vdash (Bq_1.bbaB) \vdash (Bbq_1, ba$~~

$\delta(q_0, abba) \vdash (Bq_1bbaB) \vdash (Bbq_1baB) \vdash (Bbbq_1aB)$

$\vdash (Bbbaq_1B) \vdash (Bbbq_2aB) \vdash (Bbq_3bBB)$

$\vdash (Bq_3bbBB) \vdash (q_3BbbBB) \vdash (Bq_0bbBB)$

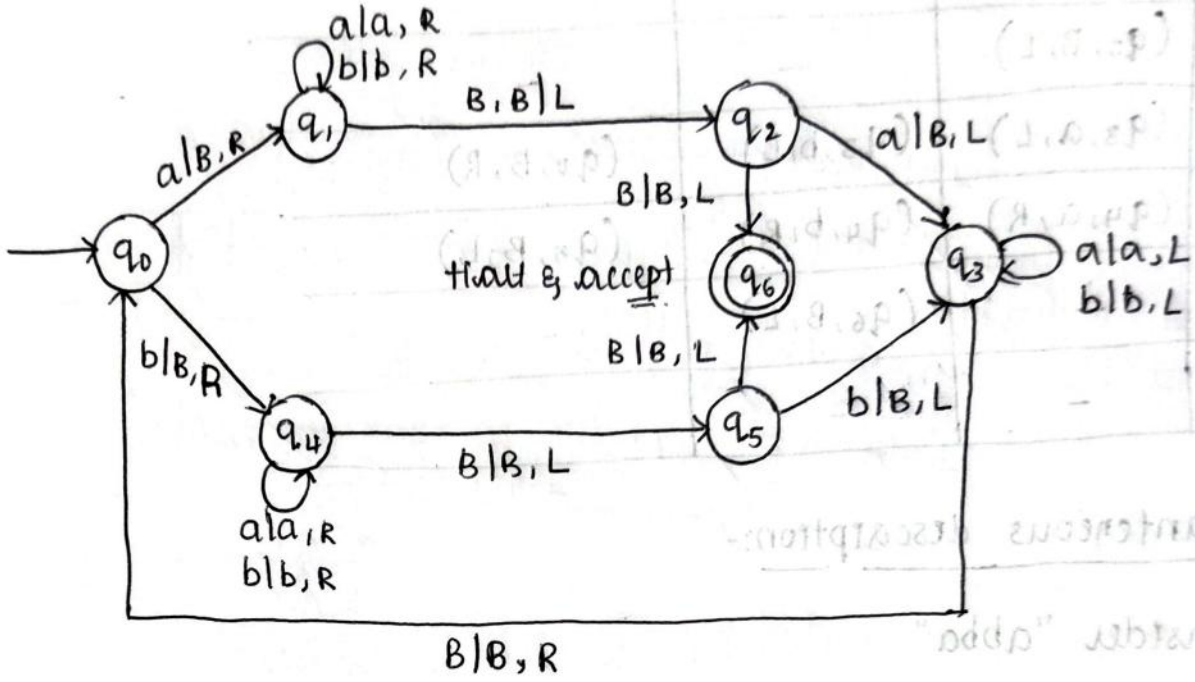
$\vdash (BBq_4bBB) \vdash (BBbq_4BB) \vdash (BBq_5bBB)$

$\vdash (Bq_3BBBB) \vdash (BBq_6BBB) \vdash (Bq_6BBBB)$

5) Construct a turing machine that accepts odd length palendrome.

Sol:-

Transition diagram:-



Transition table:-

q|T	a	b	B
q_0	(q_1, B, R)	(q_4, B, R)	-
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, L)
q_2	(q_3, B, L)	-	(q_6, B, L)
q_3	(q_3, a, L)	(q_3, b, L)	(q_0, B, R)
q_4	(q_4, a, R)	(q_4, b, R)	(q_5, B, L)
q_5	-	(q_3, B, L)	(q_6, B, L)
q_6	-	-	-

Instantenous description:-

consider string,

"ababa"

$\delta(q_0, ababa) \vdash (Bq_1 babaB) \vdash (Bbq_1 abaB)$

$\vdash (Bbaq_1 baB) \vdash (Bbabq_1 aB)$

$\vdash (Bbabaq_1 B) \vdash (Bbabq_2 aB)$

$\vdash (Bbaq_3 bBB) \vdash (Bbq_3 abBB)$

$\vdash (Bq_3 babBB) \vdash (q_3 Bbabb BB)$

$\vdash (Bq_0 babBB) \vdash (BBq_4 abBB)$

$\vdash (BBaq_4 bBB) \vdash (BBabq_4 BB)$

$\vdash (BBaq_5 bBB) \vdash (BBq_3 aBBB)$

$\vdash (BBq_3 BaBBB) \vdash (BBq_0 aBBB)$

$\vdash (BBBq_1 BBB) \vdash (BBq_2 BBBB)$

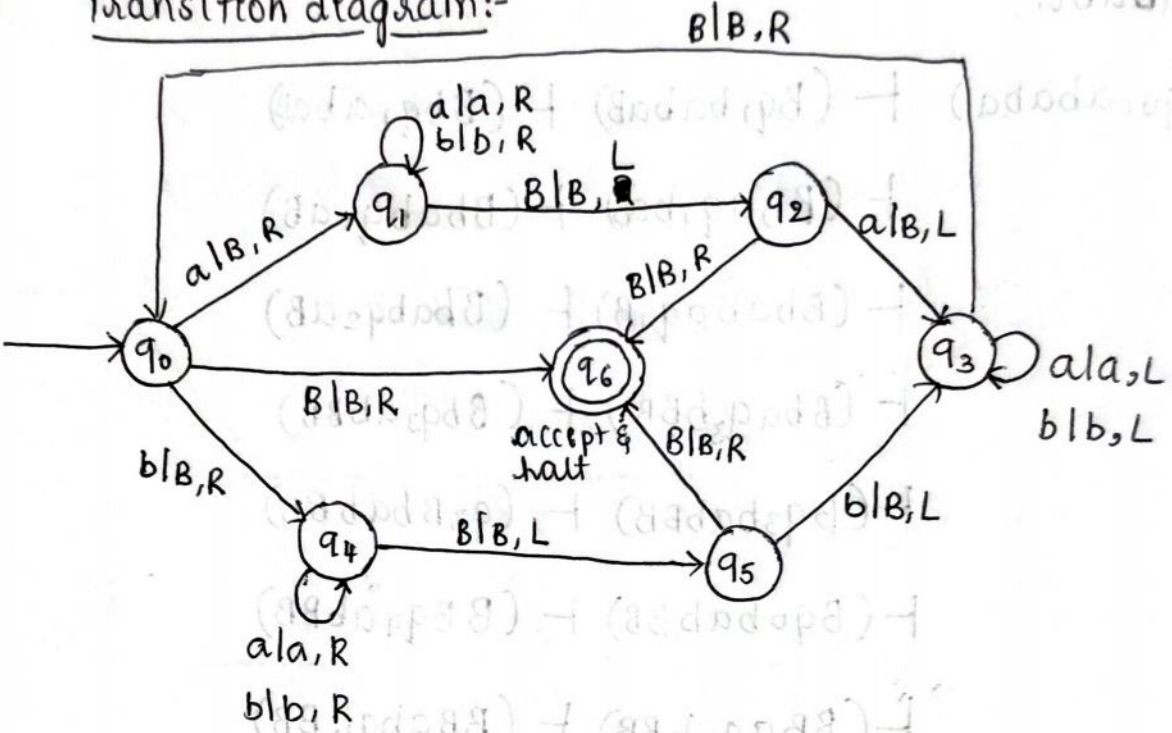
$\vdash (Bq_5 BBBBB)$

accept and halt

Note :- small change in deagram for even length and both odd and even length.

6). Construct a turing machine that accepts a palendrome (both odd and even length).

Sol:- Transition diagram:-



Transition function:-

Q\T	a	b	B
q_0	(q_1, B, R)	(q_4, B, R)	(q_6, B, R)
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, L)
q_2	(q_3, B, L)	–	(q_6, B, R)
q_3	(q_3, a, L)	(q_3, b, L)	(q_0, B, R)
q_4	(q_4, a, R)	(q_4, b, R)	(q_5, B, L)
q_5	–	(q_3, B, L)	(q_6, B, R)
q_6	–	–	–

Instantaneous description:-

consider string,

"ababa"

$\delta(q_0, ababa) \vdash (Bq_1baba B) \vdash (Bbq_1aba B)$

$\vdash (Bbaq_1ba B) \vdash (Bbabq_1a B)$

$\vdash (Bbabaq_1 B) \vdash (Bbabq_2 a B)$

$\vdash (Bbaq_3 bBB) \vdash (Bbq_3 abBB)$

$\vdash (Bq_3 babBB) \vdash (q_3 Bbab BB)$

$\vdash (Bq_0 babBB) \vdash (BBq_4 abBB)$

$\vdash (BBaq_4 bBB) \vdash (BBabq_4 BB)$

$\vdash (BBaq_5 bBB) \vdash (BBq_3 aBBB)$

$\vdash (Bq_3 BaBBB) \vdash (BBq_0 aBBB)$

$\vdash (BBBq_1 BBB) \vdash (BBq_2 BBBB)$

$\vdash (BBBq_2 BBB)$

accept and halt.

30/11/2024

7). Construct a tureng machine that performs addition of 2 unary numbers.

Sol:- Unary numbers (1/0):-

4 → 1111 (or) 0000.

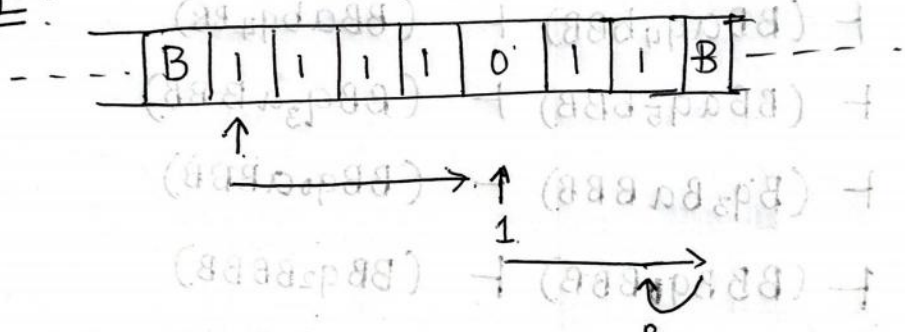
2 → 11 (or) 00.

here we consider 1 for digit and 0 for operator (delimiter)

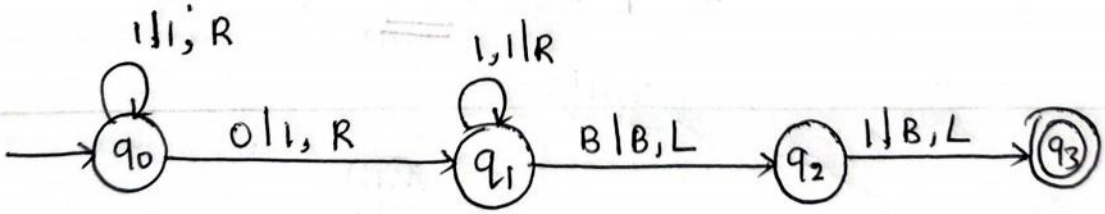
consider streng 4+2 = 6.

⇒ 1111011 = 111111.

Logic:-



Transition diagram:-



table

Transition ~~diagram~~:-

Q\T	0	1	B
q_0	$(q_0, 1, R)$	$(q_0, 1, R)$	–
q_1	–	$(q_1, 1, R)$	(q_2, B, R)
q_2	–	(q_3, B, L)	–
q_3	–	–	–

Instantenous description:-

consider "1111011",

$\delta(q_0, 1111011) \vdash (1q_1 1011) \vdash (11q_0 1011)$

$\vdash (111q_0 1011) \vdash (1111q_0 011)$

$\vdash (11111q_1 11) \vdash (111111q_1 1)$

$\vdash (1111111q_1 B) \vdash (111111q_2 1B)$

$\vdash (11111q_3 1BB)$

accept and halt

8). Construct turing machine for subtraction of 2 unary numbers (or). [monus (or) proper subtraction]

$\downarrow$

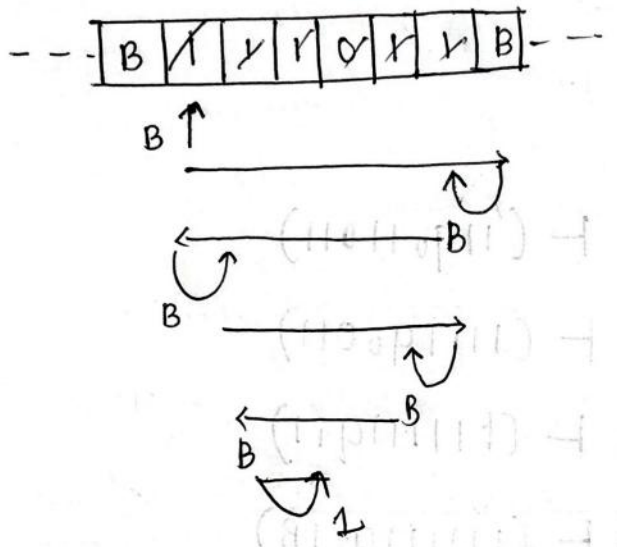
$(\dot{-})$

Sol:- Let 'm' & 'n' be two unary no's.

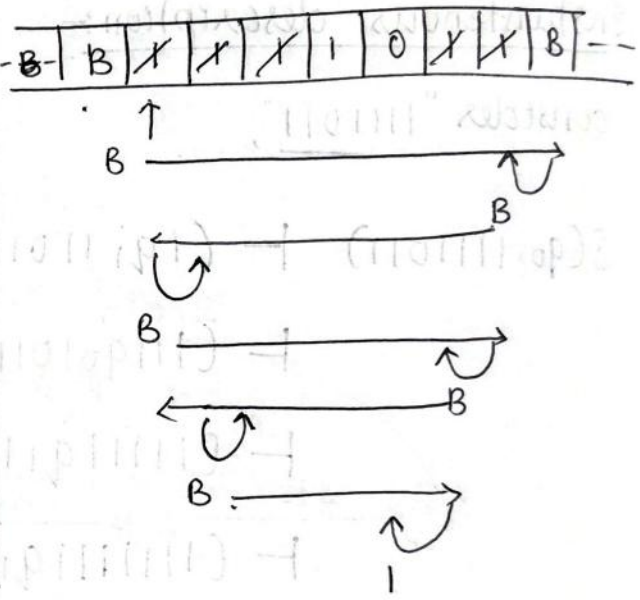
$$f(m,n) = \begin{cases} m-n, & \text{if } m > n \\ 0, & \text{if } m \leq n \end{cases}$$

Logic:-

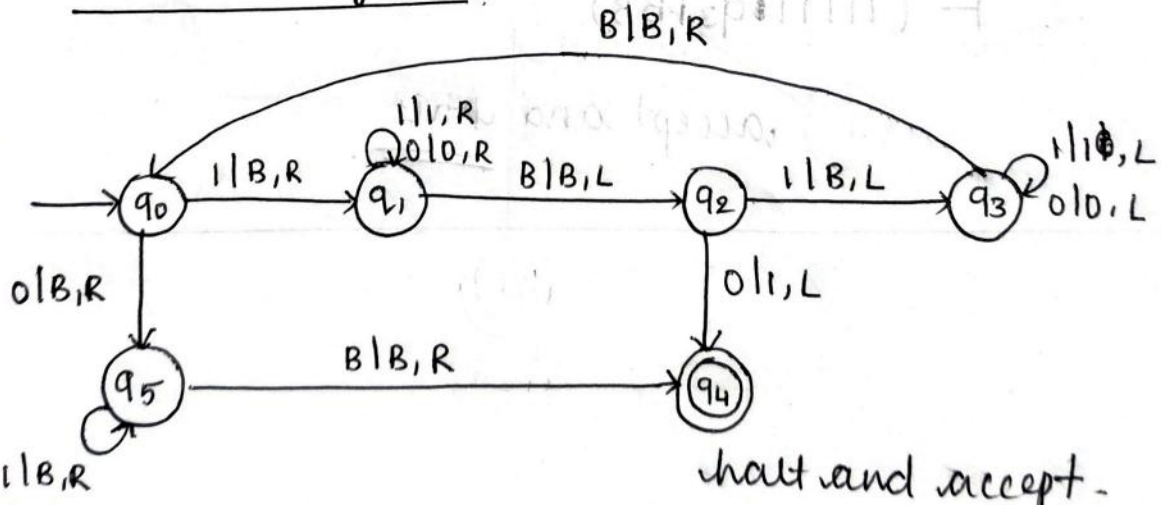
3-2.



4-2.



Transition diagram:



halt and accept.

Transition function(table) :-

Q \ T	0	1	B
q_0	(q_5, B, R)	(q_1, B, R)	-
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_2, B, L)
q_2	$(q_4, 1, L)$	(q_3, B, L)	-
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4	-	-	-
q_5	-	(q_5, B, R)	(q_4, B, R)

Instantenous description:

consider, 111011

$\delta(q_0, 111011) \vdash (Bq_1 11011\,B) \vdash (B1q_1 1011B) \vdash (B11q_1 011B)$

$\vdash (B110q_1 11B) \vdash (B1101q_1 1B) \vdash (B11011q_1 B)$

$\vdash (B1101q_2 1B) \vdash (B110q_3 1BB) \vdash (B11q_3 01BB)$

$\vdash (B1q_3 101BB) \vdash (Bq_3 1101BB) \vdash (q_3 B1101BB)$

$\vdash (Bq_0 1101BB) \vdash (BBq_1 101BB) \vdash (BB1q_1 01BB)$

$\vdash (BB10q_1 1BB) \vdash (BB101q_1 BB) \vdash (BB10q_2 1BB)$

$\vdash (BB1q_3 0BBB) \vdash (BBq_3 10BBB) \vdash (Bq_3 B10BBB)$

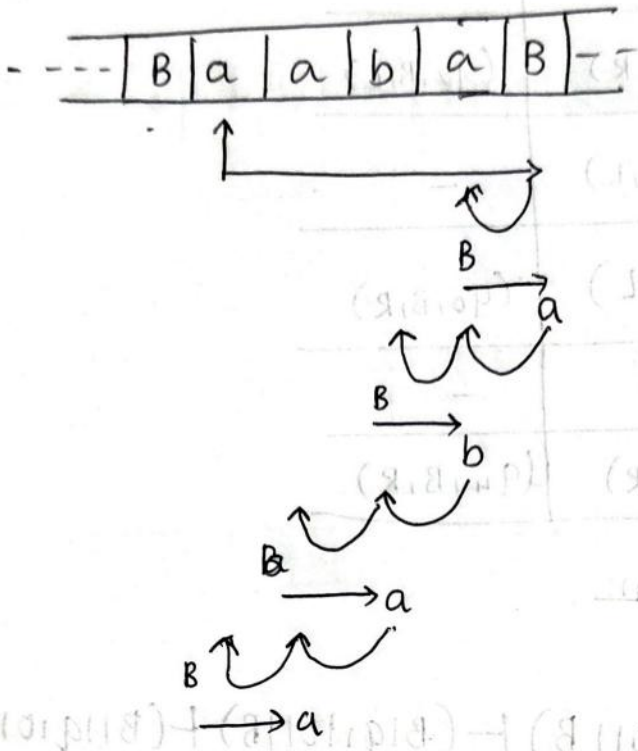
$\vdash (BBq_0 10BBB) \vdash (BBBq_1 0BBB) \vdash (BBB0q_2 BBB)$

$\vdash (BBBq_2 0BBB) \vdash (BBq_4 B1BBB)$

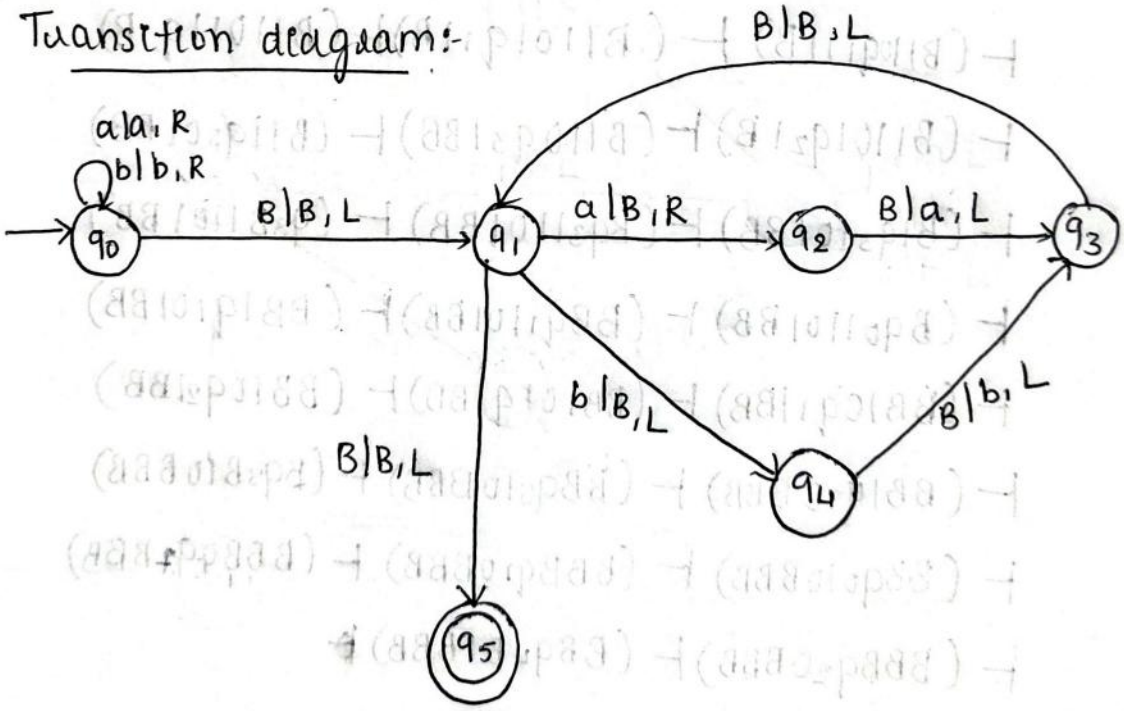
accept & halt

9). Construct turing machine that shifts the input string by one unit.

Sol: Logic:



Transition diagram:-



halt & accept.

Transition table:-

Q \ T	a	b	B
q_0	(q_0, a, R)	(q_0, b, R)	(q_1, B, L)
q_1	(q_2, B, R)	(q_4, B, R)	(q_5, B, L)
q_2	-	-	(q_3, a, L)
q_3	-	-	(q_1, B, L)
q_4	-	-	(q_3, b, L)
q_5	-	-	-

Instantenous description:-

consider aaba,

$\delta(q_0, aaba) \vdash (aq_0aba) \vdash (aaq_0ba) \vdash (aabq_0a) \vdash (aabaq_0B) \vdash (aabq_1aB) \vdash (aabBq_2B) \vdash (aabq_3Ba) \vdash (aaq_1bBa) \vdash (aaBq_4Ba) \vdash (aaq_3Bba) \vdash (aq_1aBba) \vdash (aBq_2Bba) \vdash (aq_3Baba) \vdash (q_1aBaba) \vdash (Bq_2Baba) \vdash (q_3Baaba) \vdash (q_1BBaaba) \vdash (q_5BBaaba)$

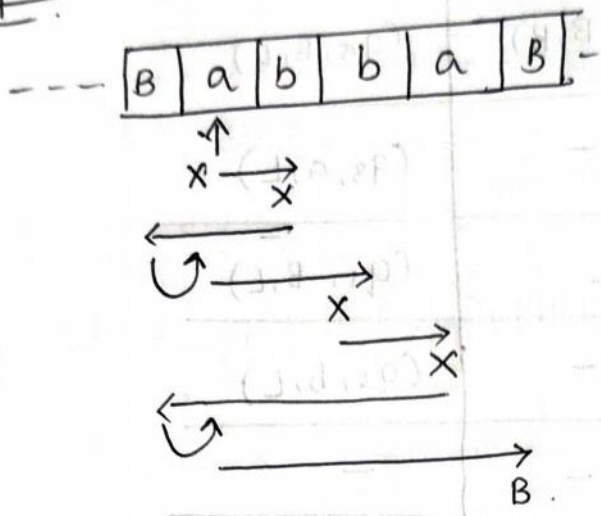
accept & halt.

10) Construct turing machine to accept the language

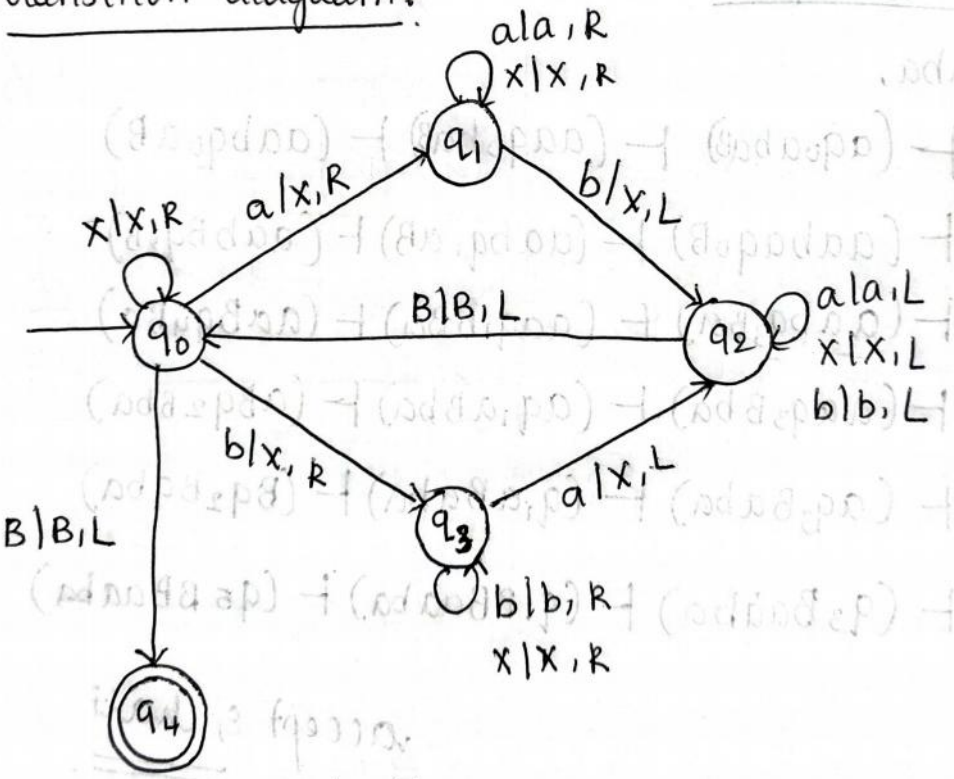
$$L = \{ N_a(w) = N_b(w) : w \in (a+b)^* \}$$

Sol:

Logic:



Transition diagram:-



accept and halt

Transition table:

Q \ T	a	b	X	B
q_0	(q_1, X, R)	(q_3, X, R)	(q_0, X, R)	(q_4, B, L)
q_1	(q_1, a, R)	(q_2, X, L)	(q_1, X, R)	–
q_2	(q_2, a, L)	(q_2, b, L)	(q_2, X, L)	(q_0, B, L)
q_3	(q_2, X, L)	(q_3, b, R)	(q_3, X, R)	–
q_4	–	–	–	–

ID:-

consider, aabbbbaa,

$\delta(q_0, aabbbbaa) \vdash (Xq_1abbbbaa) \vdash (Xaq_1bbbbaa)$

$\vdash (Xq_2aXbbbaa) \vdash (q_2^BXaXbbbaa) \vdash (q_0BXaXbbbaa)$

$\vdash (Xq_0aXbbbaa) \vdash (XXq_1bbbaa) \vdash (Xq_2XXq_1bbbaa)$

$\vdash (XXq_2XXbbbaa) \vdash (Xq_2XXXbbbaa) \vdash (q_2BXXXXbbbaa)$

$\vdash (q_0XXXXbbbaa) \vdash (Xq_0XXXbbbaa) \vdash (XXq_0XXbbbaa)$

$\vdash (XXXq_0Xbbbaa) \vdash (XXXXq_0bbbaa) \vdash (XXXXXq_3bbbaa)$

$\vdash (XXXXXbq_3baa) \vdash (XXXXXq_2bXa) \vdash (XXXXq_2XbXa)$

$\vdash (XXXq_2XXbXa) \vdash (XXq_2XXXbXa) \vdash (Xq_2XXXXbXa)$

$\vdash (q_2XXXXXbXa) \vdash (q_0XXXXXbXa) \vdash (Xq_0XXXXbXa)$

$\vdash (XXq_0XXXbXa) \vdash (XXXq_0XXbXa) \vdash (XXXXq_0XbXa)$

$\vdash (XXXXXq_0bXa) \vdash (XXXXXXq_3Xa) \vdash (XXXXXXXq_3a)$

$\vdash (XXXXXXq_2XX) \vdash (XXXXXq_2XXX) \vdash (XXXXq_2XXXX)$

$\vdash (XXXq_2XXXXX) \vdash (XXq_2XXXXXX) \vdash (Xq_2XXXXXXX)$

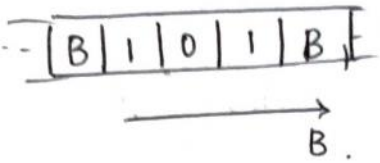
$\vdash (q_2BXXXXXXXX) \vdash (q_0BXXXXXXXX) \vdash (q_4BXXXXXXXX)$

accept and halt

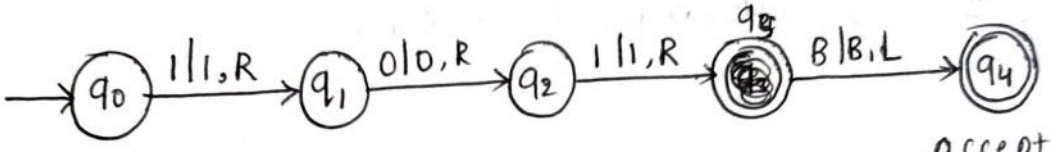
03/12/2024

11). Construct a turing machine to accept a string "101"

Sol:-



Transition diagram:-



accept & halt

Transition table:

Q \ T	0	1	B
q_0	-	$(q_1, 1, R)$	-
q_1	$(q_2, 0, R)$	-	-
q_2	-	$(q_3, 1, R)$	-
q_3	-	-	(q_4, B, L)
q_4	-	-	-

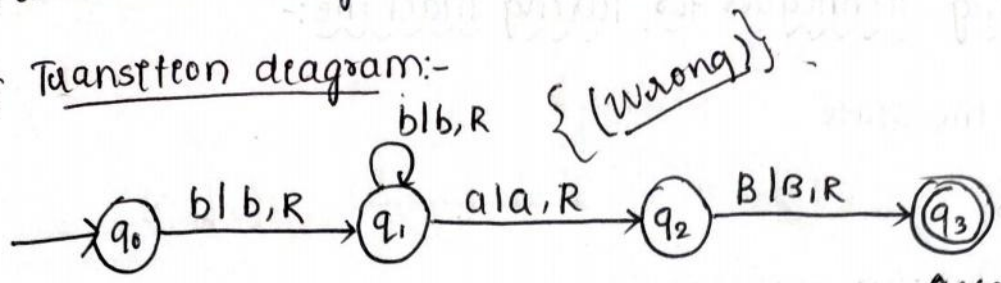
ID:-

$\delta(q_0, 101) \vdash (1q_1 01B) \vdash (10q_2 1B) \vdash (101q_3 B) \vdash (10q_4 1B)$

accept & halt.

12) Construct turing machine to accept bba.

Sol: Transition diagram:-



Transition table:

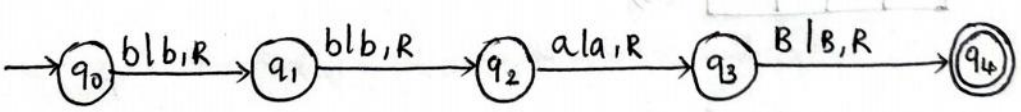
Q \ T	a	b	B
q_0	-	(q_1, b, R)	-
q_1	(q_2, a, R)	(q_1, b, R)	-
q_2	-	-	(q_3, B, R)
q_3	-	-	-

ID:

$\delta(q_0, bba) \vdash (bq_1ba) \vdash (bbq_1aB) \vdash (bbaq_2B) \vdash (bbaBq_3)$

accept & halt

Transition diagram:-



also make changes in transition table & ID.

Imp (10m)

Programming Techneques for Turing Machine:-

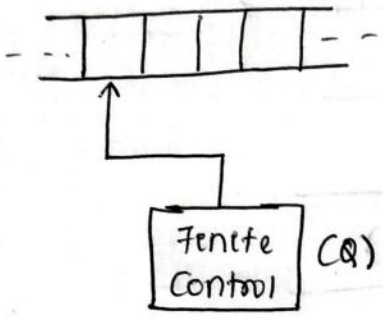
1) Storage in the state

2) Subroutines

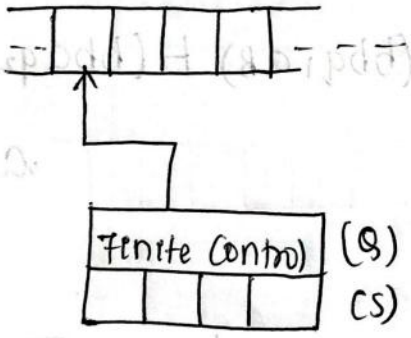
3) Multi track turing machine.

Storage in the state:-

Generally, turing machine model is



but adding storage state, it looks like



and the representation is

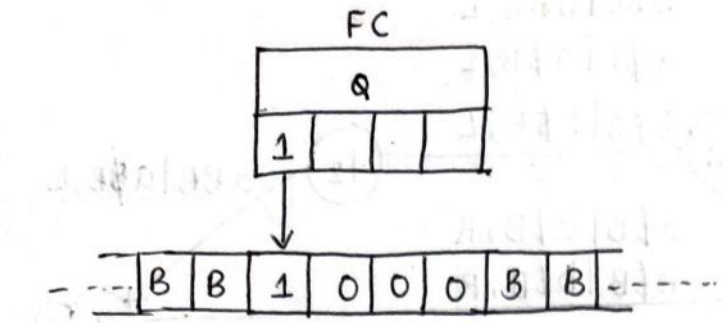
$$M_{TM} = (Q, \Sigma, T, \delta, q_0, B, F)$$

$\Downarrow$ Storage in the state

$$\boxed{M_{TM} = (Q \times T, \Sigma, T, \delta, (q_0, B), B, (q_f, B))}$$

05/12/2024

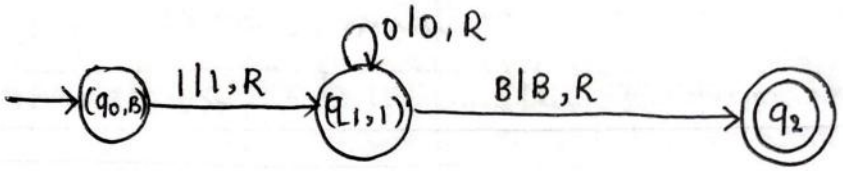
Ex:- 10^*.



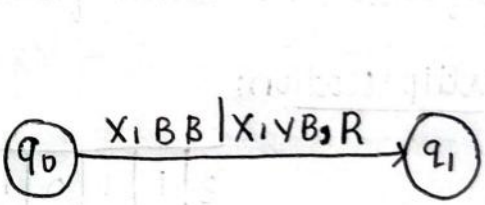
$\delta((q_0, B), 1) = ((q_1, 1), 1, R)$

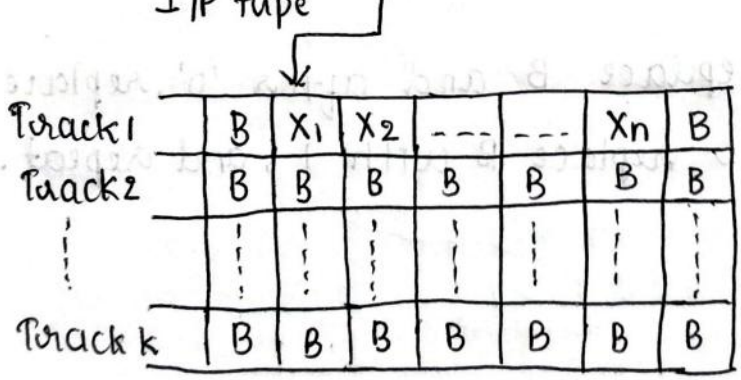
$\delta((q_1, 1), 0) = ((q_1, 1), 0, R)$

$\delta((q_1, 1), B) = ((q_2, B), B, R)$



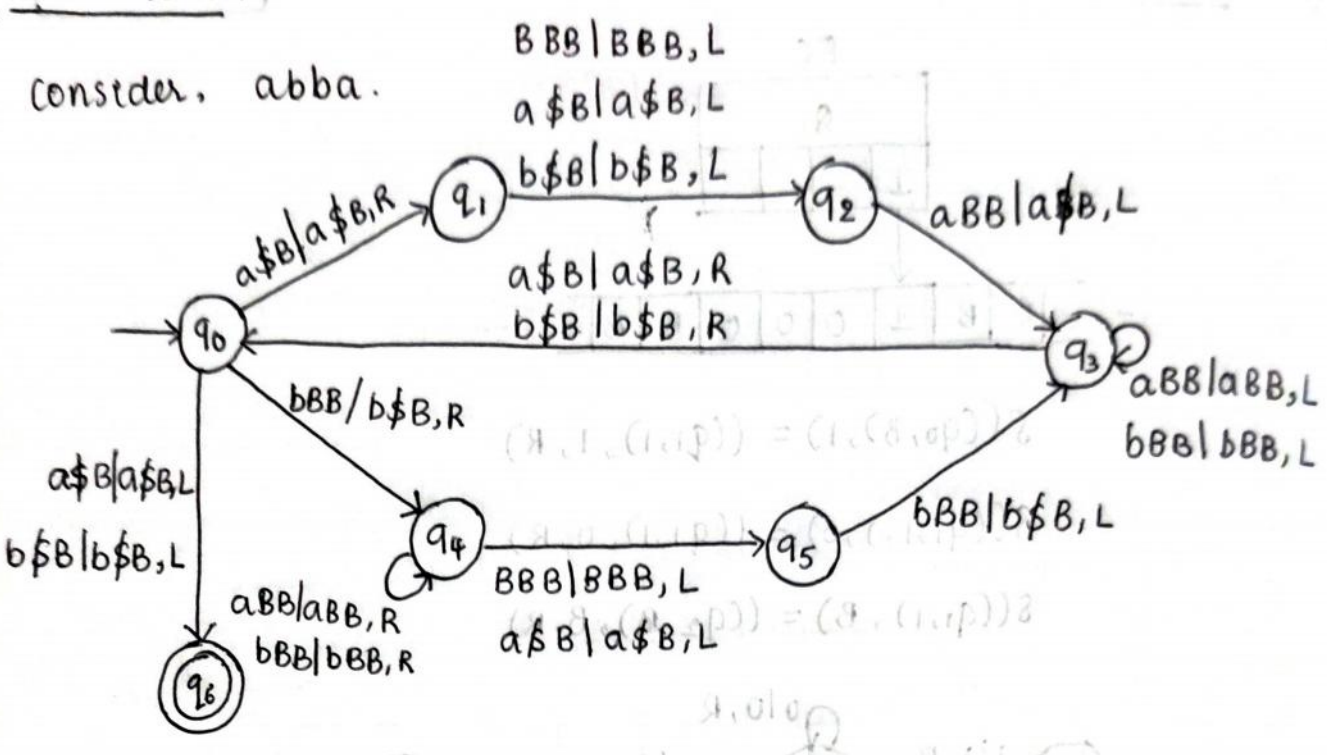
Multi-track tureng machine-





Palendrome:-

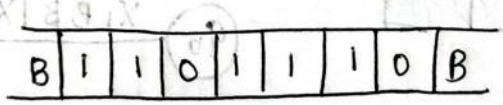
consider, abba.



Subroutine:-

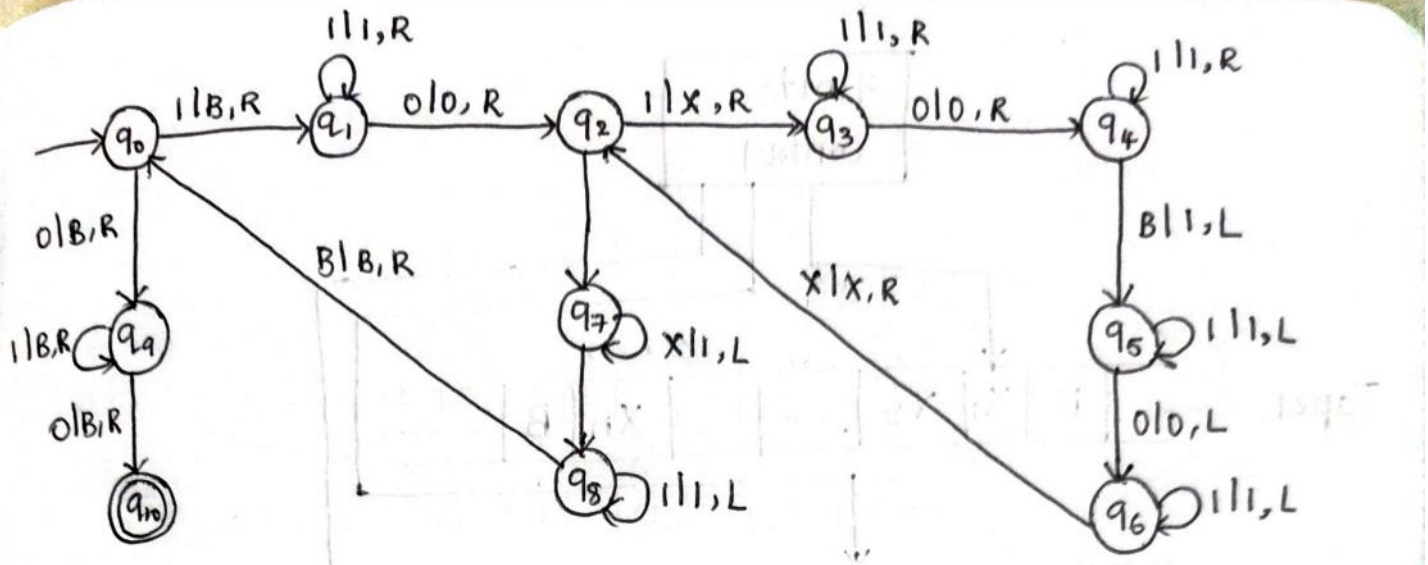
*) Subroutine is same as in program where it repeatedly perfoam the same function.

Multiplication:



Logic:-

On scanning first '1', replace B and after '0', replace 1 with X and after 0 replace B with 1; and repeat.



09|12|2024

Variants of turing machine:-

The two main variants of turing machine are:

a) Turing machine with more than one tape called as Multitape Turing machine and

b) Turing machine where $\delta(q,a) = \{(P_1, y_1, D_1), (P_2, y_2, D_2), \ldots, (P_r, y_r, D_r)\}$ called as non-deterministic turing machine.

Multi-tape Turing machine:-

A multitape Turing machine has

*) A finite set Q of states

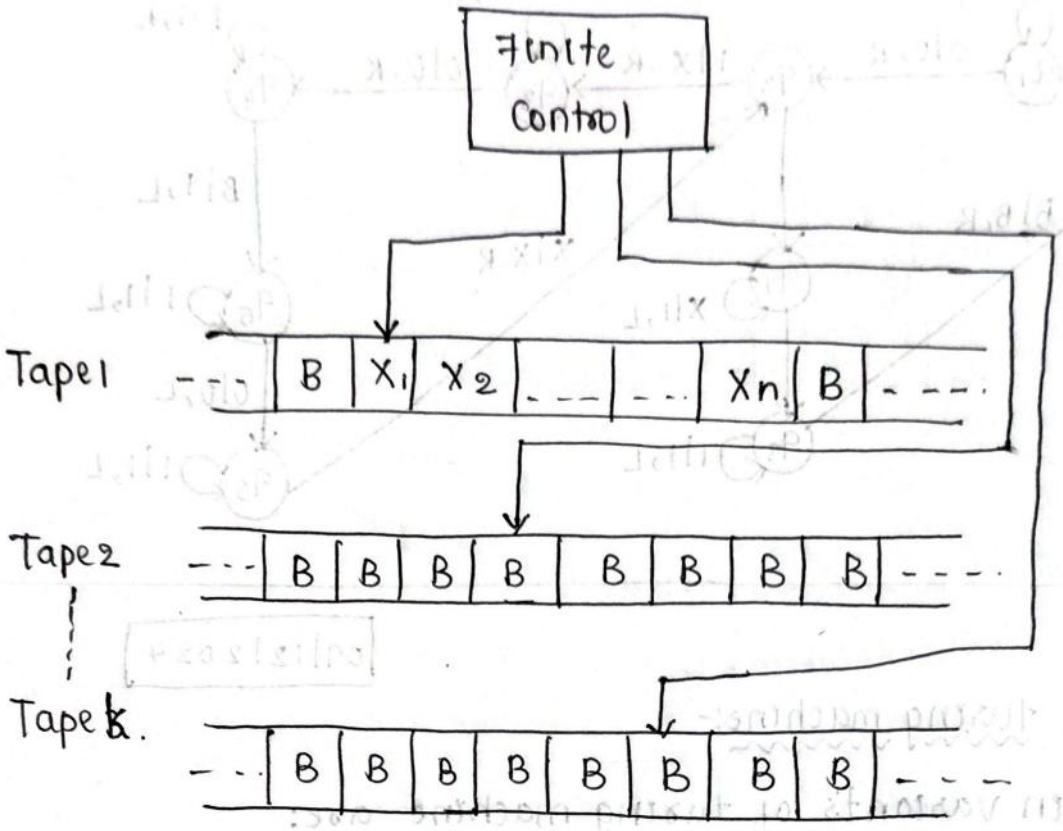
*) An initial state q_0

*) A subset F of Q called the set of final states

*) A set P of tape symbols

*) A new symbol Δ, not in P called the blank symbol.

Δ (or) B



*) There are k-tapes, each divided into cells.

*) Initially,

→ First tape holds inputstring 'w'.

→ other tapes will have B(blank) symbol.

→ Head of the first tape is at left end of input 'w'.

→ Heads of other tapes can be at any cell.

*) "δ" is a partial function:

$$Q \times T^k \longrightarrow Q \times T^k \times (L, R, S)$$

(Stationary head)

*). A particular move depends on:

→ the current state and

→ k-tape symbols under k-tape heads.

*) In a typical move,

→ Machine enters newstate.

→ On each tape, a new symbol is written on the cell under the head.

→ Each tape head moves to left, right (or) remains stationary.

Note:- Movement is independent

*). Every language accepted by a multi-tape Turing machine is accepted by single-tape turing machine.

Non-Deterministic turing machine:-

A non-deterministic turing machine is a 7-tuple representation.

$$M_{NDTM} = (Q, \Sigma, T, \delta, q_0, B, F)$$

where,

Q: Finite non-empty set of states.

Σ: Non-empty subset of T, Δ B $\notin \Sigma$.

T: Finite non-empty set of tape symbols.

B: Blank symbol, $B \in T$

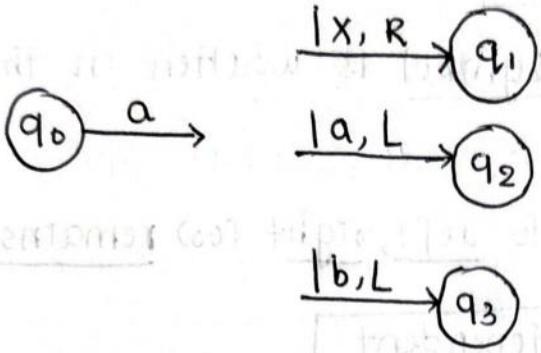
q_0: Initial state.

F: Set of final states, $F \subseteq Q$.

δ: Transition function

$$\delta: Q \times T \longrightarrow 2^{(Q \times T \times D)}$$

Ex:-



*) If M is a non-deterministic turing machine, then there is a deterministic turing machine M_1, such that,

$$T(M) = T(M_1)$$

Other variants of turing machine includes:

i) Multiple track turing machine

ii) Two way infinite tape turing machine

iii) Multi-tape multi-head turing machine

iv) Multi-dimensional turing machine

v) Off line turing machine.

Multiple track turing machine:-

*) It has k-tracks.

*) One R|W head, that reads and writes all of them one by one.

*) It can be simulated by a single track turing machine.

Two way infinite tape turing machine:-

*) It is bounded in both directions, ie., left and right.

*) It can be simulated by one-way turing machine.

Multi-tape Multi-head turing machine:-

*) It has multiple heads and multiple tapes.

*) Each tape is controlled by separate head.

*) It can be simulated by standard turing machine.

Multi-Dimensional turing machine:-

*) It has multi-dimensional tape.

*) Tape head can move in any direction, ie., left, right, up and down.

*) It can be simulated by a 1D-turing machine.

Offline Turing machine:-

*) It is similar to multi-tape turing machine, but it's input is read only.

*) It has two end markers for input ~~on~~ on input tape.

*) Remaining tapes are two-way infinite tape.

Recursive and Recursively Enumerable language:-

Given a turing machine, there are the possibilites that exists:

i) It halts and accepts the string.

ii) It halts and rejects the string.

iii) It never halts.

Recursive language:-

*) If there exists a turing machine, then that language 'L' is recursive.

*) For every string that is present in language, the machine accepts it.

*) Whereas, it rejects all strings not there in the language.

*) Consider that there is a language 'L' and it has a problem.

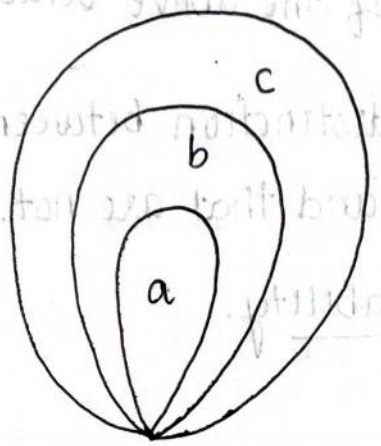
*) the problem is:

a) Decidable, if 'L' is recursive and
b) Undecidable, otherwise.

Recursively Enumerable language:-

*) If there exists a turing machine, then the language is accepted by the machine. Such language are called recursively enumerable language.

*) These are the languages for which if problem exists, then those problems are decidable.



Relationship between languages.

where, a : recursive language

b : recursively enumerable language.

c : non-recursively enumerable language.

Decidable and undecidable language:-

*) Decidability :- It is the process which helps us to know whether a turing machine halts (or) not.

*) As an algorithm halts, eventually the turing machine also halts.

*) The turing machine halts in following conditions:

i) When turing machine reaches final state, it halts.

ii) When turing machine reaches a state 'q', when input symbol scanned is 'a' and if $\delta(q,a)$ is not defined, it halts.

*) There are some turing machines which will not halt on an input string in any of the above situation.

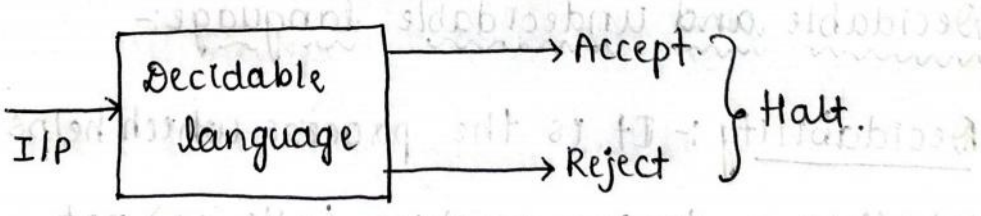
*) Hence, we need to make a distinction between language accepted by turing machine and that are not.

*) This is referred to as decidability.

Decidable language:-

*) If a language is recursive, then it is called as decidable language.

*) For all decidable languages, turing machine halts with accept/reject output.

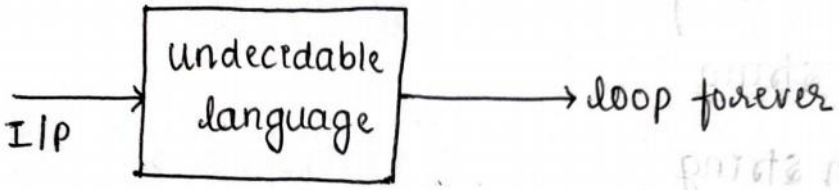


Turing machine.

Undecidable language:-

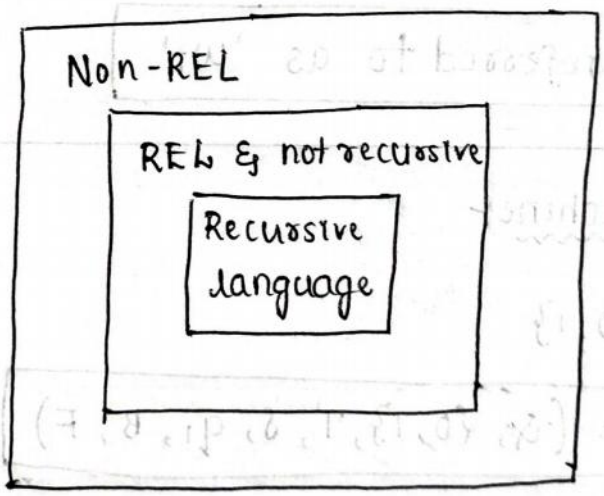
*) If a language is not recursive, then it is called as undecidable language.

*) For an undecidable language, turing machine does not Halt. Infact there is no turing machine.



Turing machine.

Relation between languages:-



REL : Recursively enumerable language

Non-REL : Non recursively enumerable language.

Enumerating Binary strings:-

*) If 'w' is a binary string, then treat '1w' as binary integer 'i'.

ie., we call 'w' the i^{th} string.

ε - first string

0 - second string

1 - third string

00 - fourth string

01 - fifth string and so on.

*) Strings are ordered by length.

*) Strings of equal length are ordered lexicographically.

'i^{th}' string is referred to as 'w_i'

Coding the turing machine:-

Input alphabet : {0, 1}

Turing machine, $M = (Q, \{0,1\}, T, \delta, q_1, B, F)$

*) M_i is the i^{th} turing machine.

*) To represent 'M' as a binary string, we must assign integers to:

a) States
b) Tape symbols and
c) Directions (i.e., L & R)

*) Assume that the states are $q_1, q_2, ----, q_k$ for some 'k'. start state is 'q_1' and accepting state is 'q_2'.

Note:- Turing machine halts when it enters the accepting state. So only one accepting state is there.

*) Assume that the tape symbols are $X_1, X_2, ----, X_m$ for some 'm'.

X_1 : '0' , X_2 : '1' and X_3 : 'B'

Note:- Other tape symbols are assigned with integers arbitrarely.

*) We shall refer directions as below:

→ L : D_1

→ R : D_2

*) After establishing integer to represent each state, tape symbol and direction, we can encode the transition function, 'δ', say

$$\boxed{\delta(q_i, X_j) = (q_k, X_l, D_m)}$$

be a rule for some integer i, j, k, l & m.

*) the encoding can be done by using the following rule:

$$\boxed{0^i 1 0^j 1 0^k 1 0^l 1 0^m}$$

where, $i, j, k, l, m \geq 1$.

Note:- there should be no occurences of two (or) more consecutive 1's within code for single transition.

*) the final code for turing machine will be each individual codes separated by 2 consecutive 1's.

i.e., $\boxed{C_1 11 C_2 11 C_3 11 \text{----} 11 C_n}$

where $C_i \rightarrow$ code for i^{th} transition function.

---

1) Encode the transition function,

$$\delta(q_1, 0) = (q_4, B, R)$$

Sol:- here, 0 : X_1

B : X_3

R :- D_2

i.e., $\delta(q_1, X_1) = (q_4, X_3, D_2)$ {this is of the form $\delta(q_i, X_j) = (q_k, X_l, D_m)$}

$\therefore i=1, j=1, k=4, l=3, m=2$

The required code is : $0^i 1 0^j 1 0^k 1 0^l 1 0^m$.

$= 0^3 1 0^2 1 0^4 1 0^3 1 0^1$

$= 0001001000010001 0$

$=$.

2) Encode,

$\delta(q_3, 1) = (q_4, 1, L)$

Solt here, 1 : X_2

~~R : D_2~~ L : D_1

ie., $\delta(q_3, X_2) = (q_4, X_2, D_1)$

i=3, j=2, k=4, l=2, m=1.

The required code is : $0^i1 0^j1 0^k1 0^l1 0^m$

$= 0^3 1 0^2 1 0^4 1 0^2 1 0^1$

$= 0001001000010010$

09/12/2024
(evening)

3) Encode,

$\delta(q_1, B) = (q_6, B, R)$

Solt B : X_3

R : D_2

ie., $\delta(q_1, X_3) = (q_6, X_3, D_1)$

i=1, j=3, k=6, l=3, m=1

The required code is : $0^i1 0^j1 0^k1 0^l1 0^m$

$= 0^1 1 0^3 1 0^6 1 0^3 1 0^1$

$= 0100010000000100010$

4) Encode,

$\delta(q_3, 1) = (q_4, B, L)$

Sol:

$1 \longrightarrow X_2$

$B \longrightarrow X_3$

$L \longrightarrow D_1$

i.e., $\delta(q_3, X_2) = (q_4, X_3, D_1)$

$\therefore i=3, j=2, k=4, l=3, m=1$

The required code is: $0^3 1 0^2 1 0^4 1 0^3 1 0^1$

$= 0001001000010001 0$.

---

5) Encode,

$\delta(q_4, B) = (q_5, B, R)$

Sol:

$B \longrightarrow X_3$

$R \longrightarrow D_2$

i.e., $\delta(q_4, X_3) = (q_5, X_3, D_2)$

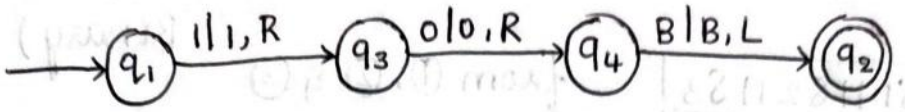
$i=4, j=3, k=5, l=3, m=2$.

$\therefore$ The required code is: $0^4 1 0^3 1 0^5 1 0^3 1 0^2$

$= 000010001000001000100$.

---

6). Convert the following turing machine into binary code.



Sol: Consider, $\delta_1 : \delta(q_1, 1) = (q_3, 1, R)$

$1 : X_2$

$R : D_2$

$\therefore \delta(q_1, X_2) = (q_3, X_2, D_2)$

$\Rightarrow i=1, j=2, k=3, l=2, m=2$

The required code is : $0^i 1 0^j 1 0^k 1 0^l 1 0^m$

$= 0^1 1 0^2 1 0^3 1 0^2 1 0^2$

$= 0100100010010010 \longrightarrow (1)$

consider, $\delta_2 : \delta(q_3, 0) = (q_4, 0, R)$

$0 : X_1$

$R : D_2$

$\therefore \delta(q_3, X_1) = (q_4, X_1, D_2)$

$i=3, j=1, k=4, l=1, m=2$

The required code is : $0^i 1 0^j 1 0^k 1 0^l 1 0^m$

$= 0^3 1 0^1 1 0^4 1 0^1 1 0^2$

$= 0001010000101001 \longrightarrow (2)$

Consider, $\delta_3 : \delta(q_4, B) = (q_2, B, L)$

$B : X_3$

$L : D_1$

i.e., $\delta(q_4, X_3) = (q_2, X_3, D_1)$

$i=4, j=3, k=2, l=3, m=1$

The required code is : $0^4 1 0^3 1 0^2 1 0^3 1 0^1$

$= 0000100010010001 0 \longrightarrow (3)$

∴ Final encoding of tureng machine (Turing machine to Binary) is

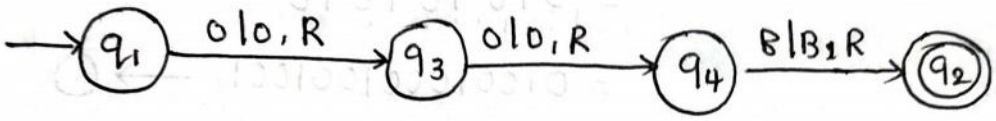
$= \boxed{\delta 1 11 \delta 2 11 \delta 3}$ from ①, ② & ③

$= 0100100010010011000101000010100110000100010010001 0$

---

7) Construct a tureng machine to accept '00' and convert it into binary.

Sol: Turing machine:- Given, streng → '00'



consider, δ_1 : $\delta(q_1, 0) = (q_3, 0, R)$

$0 : X_1$

$R : D_2$

i.e., $\delta(q_1, X_1) = (q_3, X_1, D_2)$

$i=1, j=1, k=3, l=1, m=2$.

The required code is : $0^1 1 0^1 1 0^3 1 0^1 1 0^2$

$= 0101000101008$ ⟶ ①

consider δ_2 : $\delta(q_3, 0) = (q_4, 0, R)$

$0 : X_1$

$R : D_2$

i.e., $\delta(q_3, X_1) = (q_4, X_1, D_2)$

$i=3, j=1, k=4, l=1, m=2$.

The required code is: $0^3 1 0^1 1 0^4 1 0^1 1 0^2$

$= 0001010000010100 \longrightarrow (2)$

Consider, δ_3: $\delta(q_4, B) = (q_2, B, R)$

$B : X_3$

$R : D_2$

ie., $\delta(q_4, X_3) = (q_2, X_3, D_2)$

$i=4, j=3, k=2, l=3, m=2$

The required code is : $0^4 1 0^3 1 0^2 1 0^3 1 0^2$

$= 0000100010010000100 \longrightarrow (3)$

Final encoding is

$= \delta_1 11 \delta_2 11 \delta_3$

$= 0101000101001100010100001010011000010001001000100$

Diagonalization Language (L_d) :-

Let 'M_i' be the i^{th} turing machine and 'w_i' be the binary code of 'M_i'.

If 'w_i' is a valid code, then

$$w_i \in L(M_i)$$

If 'w_i' is not a valid code, then

$w_i \notin L(M_i)$, i.e., there is no turing machine that accepts 'w_i'.

Diagonalization language (L_d) is the language that consists of 'w_i's such that

$$\boxed{w_i \notin L(M_i)}$$

M:

	1	2	3	4	5	6	- - - -
1	0	1	1	0	0	1	- - - -
2	1	0	1	0	1	0	- - - -
3	0	1	1	1	0	1	- - -
4	1	0	1	1	1	0	- - -
5	0	0	1	1	1	0	- - -
6	1	1	1	0	0	1	- -
⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Note:-
The process of complementing the diagonal is called as diagonalization.

Consider diagonal, [0 0 1 1 1 1]

↓ complement

$\Rightarrow$ [1 1 0 0 0 0]

↓

This does not contain a turing machine.

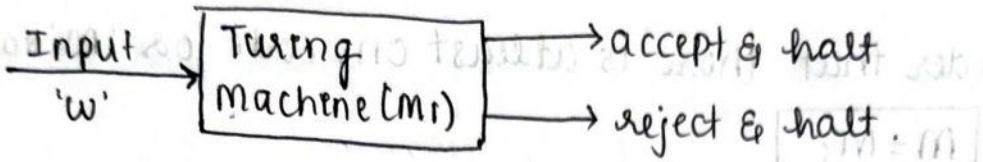
Imp (5-6)m

Halting problem:-

*) It is not actually a problem, instead it is a question that whether it is possible to tell that a given machine will halt for some given input (or) not.

*) Let us assume that there exists a turing machine that solves a given problem i.e., the machine will halt eventually.

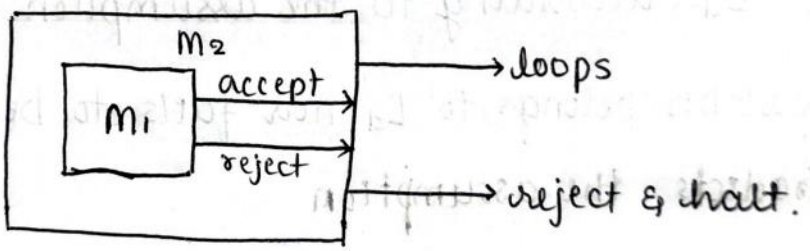
Let us represent the machine as follows:



i.e., there is a machine 'M_1' which either accepts 'w' and halt (or) rejects 'w' and halt.

*) Let us consider there is another machine, say M_2, which depends on output of M_1

i.e., M_2 is defined as below,



*) The working of M_2, which itself is another turing machine is as below,

→ If M_1 rejects 'w', M_2 also rejects 'w' and halt.

→ If M_1 accepts 'w', then M_2 loops forever on 'w'.

*) This contradicts our assumption

∴ Halting problem is undecidable.

5m Imp.

10/12/2024

Theorem:-

Diagonalization language is not a recursively enumerable language.

Proof:- *) Suppose $L_d \in L(m)$ for some turing machine 'm'.

*) W.K.T L_d is a language over $\{0,1\}$, hence M would be in the list of turing machine's constructed.

Let us consider that there is atleast one code for 'M' say i such that $\boxed{M = M_i}$

i) If $w_i \in L_d$, then M_i accepts w_i

But W.K.T $w_i \notin L_d$, according to definition of L_d.

ii) If $w_i \notin L_d$, then M_i does not accepts (rejects) w_i.

But $w_i \in L_d$, according to the assumption.

Thus w_i neither belongs to L_d nor fails to be in L_d.

This contradicts the assumption

$\therefore L_d$ is not recursively enumerable language

Hence, the proof.

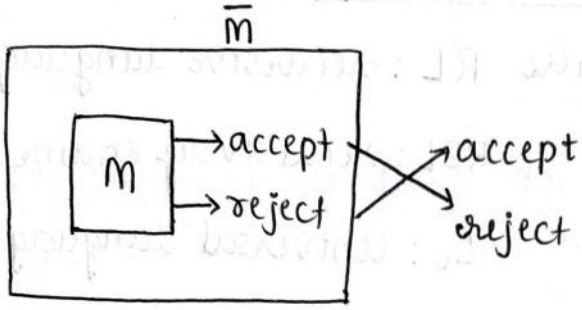
Imp. Theorem :-

If L is recursive language, so is 'L' complement.

Proof :- Let 'L' $\in L(M)$ for some turing machine 'M' always halts. We construct a TM, '$\bar{M}$' (compliment) such that

$\boxed{\bar{L} \in L(\bar{M})}$

i.e., $\bar{M}$ behaves just like M, however it is modified as below



i) the accepting states of 'M' are made as non-accepting states of $\bar{M}$.

ii) $\bar{M}$ has new accepting state 'R' such that there are no transitions from R.

iii) For each combination of non-accepting state of M and a tape symbol of 'M' such that M has no transition, add a transition to accepting state 'R'.

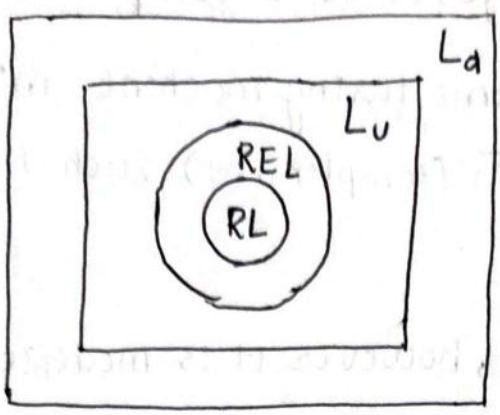
$\because$ M is guaranteed to halt, $\bar{M}$ also halts.

Also $\bar{M}$ accepts all strings not accepted by M

$\therefore$ $\bar{L}$ is recursive

Hence, the proof.

Note:-



where RL : recursive language

REL : recursively enumerable language

L_U : Universal language

L_d : Diagonalization language.

Universal Turing machine (U) and universal Language (L_u):-

Universal language(L_U):-

*) It is a set of binary strings of the form (M, w)

where M → Turing machine

w → streng , $w \in (0+1)^*$

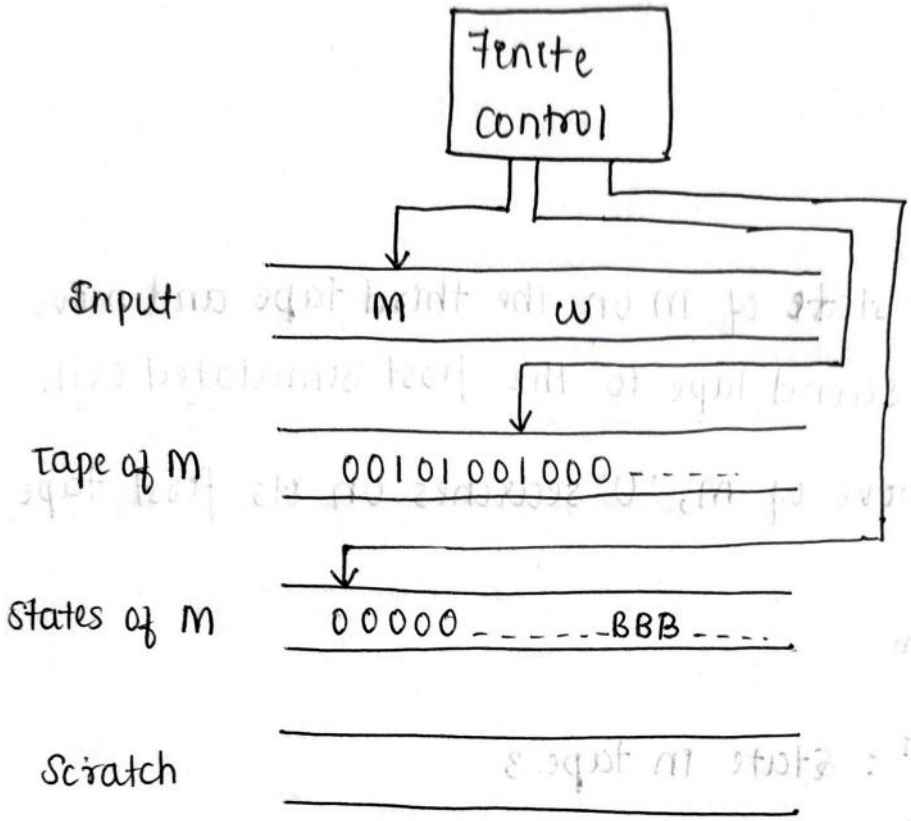
such that $\boxed{w \in L(M)}$

* Generally denoted by L_u.

Universal turing machine (U):-

A turing machine which is capable of accepting universal language is called as universal turing machine (U).

ie., $L_U = L(U)$



Consider a multi-tape turing machine with

tape 1:- (M, w) in coded form

tape 2: holds the simulated tape of M in coded form.

tape 3: holds the state of M in coded form.

tape 4: for any calculation purpose (if required).

The operation is as below:

(PTO)

1) Examine the input to make sure that code for M is a legitimate code for some turing machine.

↳ If not, 'U' halts without accepting.

2) Initialize the second tape to contain the input 'w' in encoded form

ie., 0 ⟶ 10
1 ⟶ 100
B ⟶ 1000

3) Place 0, the start state of M on the third tape and move the head of U's second tape to the first simulated cell.

4) To simulate a move of M, 'U' searches on its first tape for transition:

$0^i 1 0^j 1 0^k 1 0^l 1 0^m$

such that, 0^i : state in tape 3

0^j : tape symbol of M that begin at a position in tape 2

Now, 'U' should:

a) Change the contents of tape to '0^k'.

b) Replace '0^i' on tape 2 by '0^l'.

c) Move the head on tape 2 to the position of next 1 to the left (or) right respectively.

5) If 'M' has no transition that matches the simulated state & tape symbol, then no transition is found.

↳ M halts in the simulated configuration.

6) If M enters its accepting state, then 'U' accepts. In this way 'U' simulates 'M' on 'w'. 'U' accepts coded (M, w) if 'M' accepts 'w'

Theorem:-

If both L and $\bar{L}$ are recursively enumerable, then L is recursive.

Proof:- Let $L = L(M_1)$ and

$\bar{L} = L(M_2)$

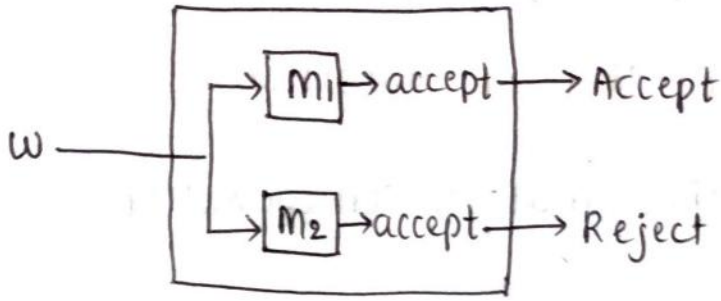
where M_1 and M_2 are simulated in parallel by a turing machine 'M'.

Let us consider 'M' as two-tape turing machine to make simulation easy,

i.e., one tape simulates M_1 and other will simulate M_2

Here, the states of M_1 and M_2 are components of M

(PTO)



i) If 'w' to 'M' is in 'L', then M_1 will eventually accept.

ii) If 'w' is not in 'L', then M_2 will eventually accept.

In both case, we can observe that 'M' halts and $L(M) = L$ with this we can conclude that L is recursive.

xxx completed xxx