CHAPTER- 4:-

TM.

Turing Machine:-

Introduction - Taught.

Definition of Turing Machine. - Taught

4.0.1. Notation of Turing Machine:-

Turing machine is a Quintuple - 5-tuple i.e.

$TM = (Q, \Sigma, \delta, q_0, H)$

Where, Q is finite set of states

Σ is alphabet set containing blank symbol ⊔/# and left end symbol, ▷.

i.e. if Σ_0 is set of input alphabet,

$\Sigma = \Sigma_0 \cup \{⊔/\# , ▷\}$

$q_0 = S$ is initial state.

$H \subseteq Q$ is the halting state.

$\delta : (Q-H) \times \Sigma \rightarrow Q \times (\Sigma \cup \{\leftarrow, \rightarrow\})$

is transition function such that

$\forall q \in (Q-H)$, $\delta(q, ▷) = (p, b)$ then $b = \rightarrow$

$\forall q \in (Q-H)$, $\delta(q, a) = (p, b)$ then $b \neq$

Thus when turing machine is in state q and scans 'a', it enters the state 'p' and either writes 'b' if $b \in \Sigma$

or moves left if $b = \leftarrow$

or moves right if $b = \rightarrow$

Machine stops computation after reaching the halting state, $h \in H$.

→ It always moves right on left end symbol.

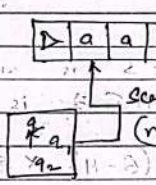
Configuration of TM:-

Configuration of TM

$T = (Q, \Sigma, \delta, q_0, H)$ is the member of $(Q \times \triangleright \Sigma^* \times (\Sigma^* (\Sigma - \{\sqcup, \#\}) \cup \{\epsilon\}))$

Example.

Any instance of TM, T, can be given by its Configuration.



$(q_1, \triangleright a, ab)$

or $(q_1, \triangleright, \underline{a}ab)$

This means, the machine is at state q and the scan head is at the first input symbol 'a'.

This can also be represented by another way as,

$(q, \triangleright \underline{a} ab)$ for simplicity where underscore gives the

scan head position. Note all configurations are assumed to start with the left end symbol (▷) and never end with a blank unless the blank is currently scanned.

Configuration $\vdash$ Configuration of TM (move relation)

For TM $T = (Q, \Sigma, \delta, q_0, H)$

and its configurations, $(q_1, w_1 \underline{a_1} u_1)$ and $(q_2, w_2 \underline{a_2} u_2)$,

$$(q_1, w_1 a_1 u_1) \vdash_{TM} (q_2, w_2 a_2 u_2)$$

if and only if, for some $b \in \Sigma \cup \{\leftarrow, \rightarrow\}$

$\delta(q_1, a_1) = (q_2, b)$ and

either $b \in \Sigma$, $w_1 = w_2$, $u_1 = u_2$, $a_2 = b$

or, $b = \leftarrow$, $w_1 = w_2 a_2$ and

either $u_2 = a_1 u_1$ if $a_1 \neq \#, u_1 \neq e$

or, $u_2 = e$ if $a_1 = \#, u_1 = e$

or, $b = \rightarrow$, $w_2 = w_1 a_1$ and

either $u_1 = a_2 u_2$

or, $u_1 = u_2 = e = \# / \sqcup$

$\vdash^*$ is reflexive transitive closure of $\vdash$ so that configuration C_1 yields C_2 then,

$$C_1 \vdash^* C_2$$

Pb(1.) Consider the Turing machine $M = (K, \Sigma, \delta, s, \{h\})$. where,

$K = \{q_0, q_1, h\}$

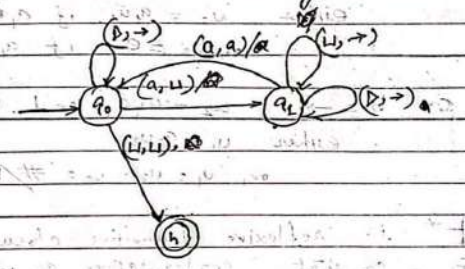
$\Sigma = \{a, \sqcup, \triangleright\}$

$s = q_0$

and δ is given by

q_i	σ	$\delta(q_i, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_0	$\triangleright$	$(q_0, \rightarrow)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_1, \rightarrow)$
q_1	$\triangleright$	$(q_1, \rightarrow)$

(1) Draw its transition diagram



Pb(2.) Consider the Turing machine of example (pb-1) if M is started in configuration $(q_0, \triangleright \sqcup a$ then list its all computation in move relation. claim the acceptance.

Soln:-

Given $(q_0, \triangleright \sqcup aaaa) \vdash_M (q_0, \triangleright \sqcup \underline{a} aaa)$

$\vdash_M (q_1, \triangleright \sqcup \sqcup aaa)$

$\vdash_M (q_0, \triangleright \sqcup\sqcup \underline{a} aa)$

$\vdash_M (q_1, \triangleright \sqcup\sqcup\sqcup aa)$

$\vdash_M (q_0, \triangleright \sqcup\sqcup\sqcup\sqcup \underline{a} a)$

$\vdash_M (q_1, \triangleright \sqcup\sqcup\sqcup\sqcup\sqcup a)$

$\vdash_M (q_0, \triangleright \sqcup\sqcup\sqcup\sqcup\sqcup \underline{a})$

$\vdash_M (q_1, \triangleright \sqcup\sqcup\sqcup\sqcup\sqcup\sqcup)$

$\vdash_M (q_0, \triangleright \sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup)$

$\vdash_M (h, \triangleright \sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup)$

$\therefore (q_0, \triangleright \sqcup aaaa) \vdash_M^* (h, \triangleright \sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup)$ Consisting of 10 steps the computation is completed.

Pb (5.) Design a Turing machine that changes non-blank symbols into blank (i.e acts as an eraser) over an alphabet $\Sigma = \{a, \sqcup, \triangleright\}$. Hence, Test your design for $\triangleright \sqcup a a \sqcup$

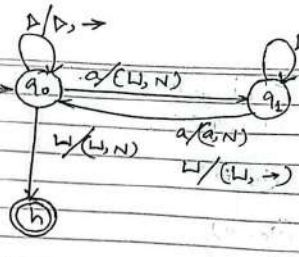
Sol^n. Let $M = (K, \Sigma, \delta, s, H)$ be required Turing machine.

Where, $K = \{q_0, q_1, h\}$
$\Sigma = \{a, \sqcup, \triangleright\}$
$s = q_0$
$H = \{h\}$

and the transition function δ is as follows.

q_i	σ	$\delta(q_i, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_0	$\triangleright$	$(q_0, \rightarrow)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$
q_1	$\triangleright$	$(q_1, \rightarrow)$

The transition diagram be:



Now, checking this design for given input, $\triangleright \sqcup a a \sqcup$

$q_0, \triangleright \sqcup a a \sqcup \vdash_M (q_1, \sqcup \sqcup a \sqcup)$
$\vdash_M (q_0, \triangleright \sqcup \sqcup a \sqcup)$
$\vdash_M (q_1, \triangleright \sqcup \sqcup \sqcup \sqcup)$
$\vdash_M (q_0, \triangleright \sqcup \sqcup \sqcup \sqcup)$
$\vdash_M (h, \triangleright \sqcup \sqcup \sqcup \sqcup)$ //

Using the another notation # instead of ⊔.
The design be,
Let $M = (K, \Sigma, \delta, s, \{h\})$ where,
$K = \{q_0, q_1, h\}$
$\Sigma = \{a, \#\}$
$s = q_0$
and δ is given by,

TM-8

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \#)$
q_0	#	$(h, \#)$
q_1	a	(q_0, a)
q_1	#	(q_0, R)

Now, checking for # a a #

$(q_0, \#\underline{a}a\#) \vdash_M (q_1, \#\underline{\#}a\#)$
$\vdash_M (q_0, \#\#\underline{a}\#)$
$\vdash_M (q_1, \#\#\#\underline{\#})$
$\vdash_M (q_0, \#\#\#\underline{\#})$
$\vdash_M (h, \#\#\#\underline{\#})$ ✓

Pb (8) Design a Turing machine TM, that accepts the language $L = \{w \in (a,b)^* : w$ contains substring $aab\}$.
Given that the initial configuration $(q_0, \triangleright \# w \#)$

Sol: Let the required Turing machine be,
$T = (Q, \Sigma, \delta, q_0, H)$ where,
Set of states, $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

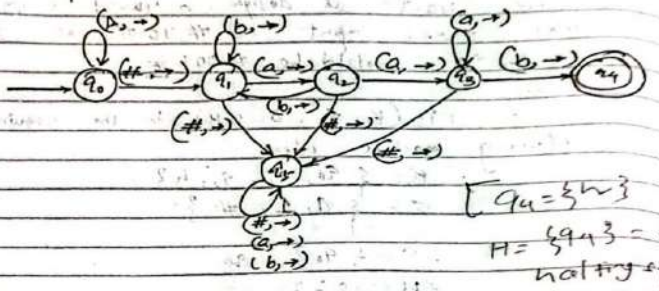
$H = \{q_4\}$

Alphabets, $\Sigma = \{a, b, \#, \triangleright\}$

$\Sigma_0 = \{a, b\}$

$w \in \Sigma_0^*$

The design looks



$[q_4 = \{h\}$
$H = \{q_4\} = \{$
not my set

The transition functions, $\delta = \{((q_0, \triangleright)(q_0, \rightarrow)),$
$((q_0, \#)(q_1, \rightarrow)),$
$((q_3, a)(q_3, \rightarrow)), ((q_3, b)$
$(q_3, \#)(q_5, \rightarrow)\}$
etc.

Q(5) Let $\Sigma_0 = \Sigma_1 = \{0, 1\}$ and let $f: \Sigma_0^* \to \Sigma_1^*$ be defined as for any $\omega \in f(\omega) = \omega'$, where ω' is the result of replacing each occurrence of '0' in ω by '1' and vice versa. Complementing Turing machine. Design a Turing machine M which computes f and check your design using the string '110'.

Sol. We want to design a Turing machine that takes input as #110# then its output would be #001#.

Let,

$M = (K, \Sigma, \delta, s, H)$ be the required turing machine where,

$K = \{q_0, q_1, q_2, h\}$

$\Sigma = \{0, 1, \underline{\#}\}$

$s = \{q_0\} = q_0$

$H = \{h\}$

and the transition ~~diagram~~ δ is given by

q	ω	δ(q, ω)
q_0	0	(q_1, L)
q_0	1	(q_1, L)
q_0	#	(q_1, L)
q_1	0	$(q_2, 1)$
q_1	1	$(q_1, 0)$
q_1	#	(q_2, R)

q_2	0	(q_2, R)
q_2	1	(q_2, R)
q_2	#	$(h, \#)$

Now testing the string for #110#.

$(q_0, \#110\underline{\#}) \vdash_M (q_1, \#11\underline{0})$

$\vdash_M (q_0, \#11\underline{1})$

$\vdash_M (q_1, \#1\underline{1}1)$

$\vdash_M (q_0, \#1\underline{0}1)$

$\vdash_M (q_1, \#\underline{1}01)$

$\vdash_M (q_0, \#\underline{0}01)$

$\vdash_M (q_1, \underline{\#}001)$

$\vdash_M (q_2, \#\underline{0}01)$

$\vdash_M (q_2, \#001)$

$\vdash_M (q_2, \#001\underline{\#})$

$\vdash_M (h, \#001\#)$

proved

Pb(5) Design a Turing machine for the regular expression $r = aa^*$.

let, the Turing machine TM,

$M = (K, \Sigma, \delta, s, H)$

Where,

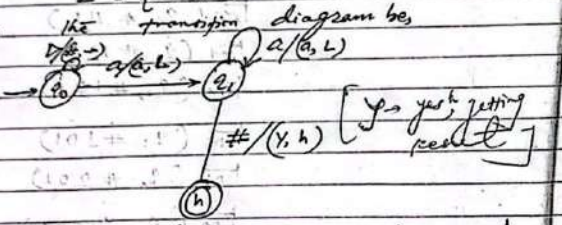
$K = \{q_0, q_1, h\}$

$H = \{h\}$

$s = q_0$

$\Sigma = \{a, \#\}$

Then, the transition diagram be,



Here, Y stands for yes string is accepted here halt state Y_h signifies as accepting.

State. [Test the result for, #.aaa#] ✓

Now, $(q_0, \#aaa\#) \vdash_M q_0, \#aaa\#$

$q_1, \#a\underline{a}a\#$

✓

Pb-(4) Design a Turing machine for the language

$L = \{(a\,b)^n \text{ for } n \geq 0\}$

or for the regular expression. $r = (ab)^*$

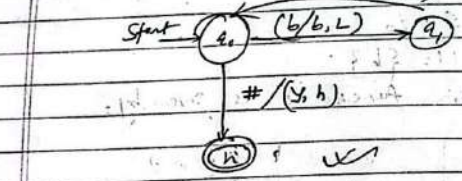
Sol:- TM $M = (Q, \Sigma, \delta, s, H)$

$H = \{h\}$

$\Sigma = \{a, b, \#\}$

$Q = \{q_0, q_1, h\}$

Then, the transition diagram be,



✓

Test your answer for;

#.abab# ✓ do ✓

Note. Y = Yes it is accepted

Q. (E) Design a Turing machine that recognizes the set of all strings of 0's & 1's containing at least one 1. Machine should print 'y' if 1 is found and should print 'n' if No 1 is found. Hence, test your design for #100#.

Sol: Let $M = (K, \Sigma, \delta, s, H)$ be required turing machine where,

$K = \{q_0, q_1, h\}$

$\Sigma = \{0, 1, \#\}$

$s = q_0$

$H = \{h\}$

and transition function δ is given by,

q	σ	$\delta(q, \sigma)$
q_0	0	$(q_1, 0)$
q_0	1	$(q_1, 1)$
q_0	#	(q_1, L)
q_1	0	(q_1, L)
q_1	1	(h, y)
q_1	#	(h, n)

Draw its transition diagram.

Given input #100#' # (#100#)

$(q_0, \#100\#) \vdash_M (q_1, \#100\#)$

$\vdash_M (q_1, \#100\#)$

$\vdash_M (q_1, \#100\#)$

$\vdash_M (h, \#y00\#)$

Pb (5) Design a Turing machine that scans to right until it finds two consequitive 'a's, & then halts. The alphabet of the Turing machine should be $\Sigma = \{a, b, \triangleright, \sqcup\}$ Hence, test your design for $\triangleright\sqcup a\sqcup\sqcup aba\,a$.

Sol: Let $M = (K, \Sigma, \delta, s, H)$ be required turing machine.

where, $K = \{q_0, q_1, h\}$

$\Sigma = \{a, b, \triangleright, \sqcup\}$

$s = q_0$

$H = \{h\}$

and the transition function δ is,

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \rightarrow)$
q_0	b	$(q_0, \rightarrow)$
q_0	$\sqcup$	$(q_0, \rightarrow)$
q_0	$\triangleright$	$(q_0, \rightarrow)$
q_1	a	(h, a)
q_1	b	$(q_0, \rightarrow)$

TM-16

q_1	$\sqcup$	$(q_0, \rightarrow)$
q_1	$\triangleright$	$(q_0, \rightarrow)$

for the input $\triangleright \sqcup a \sqcup abaa$

$(q_0, \triangleright \underline{\sqcup} a \sqcup abaa) \vdash_M (q_0, \triangleright \sqcup \underline{a} \sqcup abaa)$

$\vdash_M (q_1, \triangleright \sqcup a \underline{\sqcup} abaa)$

$\vdash_M (q_0, \triangleright \sqcup a \sqcup \underline{a}\, baa)$

$\vdash_M (q_1, \triangleright \sqcup a \sqcup a \underline{b} aa)$

$\vdash_M (q_0, \triangleright \sqcup a \sqcup ab \underline{a} a)$

$\vdash_M (q_1, \triangleright \sqcup a \sqcup aba \underline{a})$

$\vdash_M (h, \triangleright \sqcup a \sqcup aba \underline{a})$

Short Hand Notation for Turing Machine:- TM-1

For the short hand notation following literals are used to describe the turing machine.

→ L, R represents left to right moves.

a ⇒ Write A.

> ⇒ Starting point

$\xrightarrow{a}$ R ⇒ move right if you read 'a'.

$\xrightarrow{\bar{a}}$ R ⇒ move right if any other symbols than 'a' i.e.

$L^{\bar{a}}$ ⇒ Keep on moving left as long as you see 'a' or move left as long as you see 'a' and stop if you do not see 'a'.

$L^{a} \equiv L\bar{a}$ ⇒ Keep moving left till you read 'a' an stop when you read $\bar{a}$.

$\underline{L}a$ ⇒ stop moving left if you see 'a'.

R → R_2 ⇒ for any symbol you read, move right t $\equiv RR \Rightarrow R^2$

$\xrightarrow{a\,\#}$ R ⇒ Move right if you see any sy 'a' which is not # (blank-symbol)

Pb (10.) Design a Turing Machine that accepts the language of all strings which contain 'aba' as substring.

Sol^n Let the Turing machine is $TM = (Q, \Sigma, \Gamma, \delta, q_0, h)$ where,

$\Sigma = \{a, b\}$

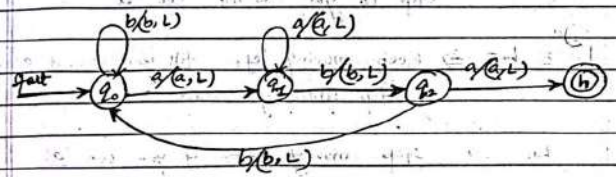
$\Gamma = \{a, b, \#\}$

$Q = \{q_0, q_1, q_2, h\}$

The initial state 's' = q_0

& the halting state $H = \{h\}$.

Then the transition diagram be:



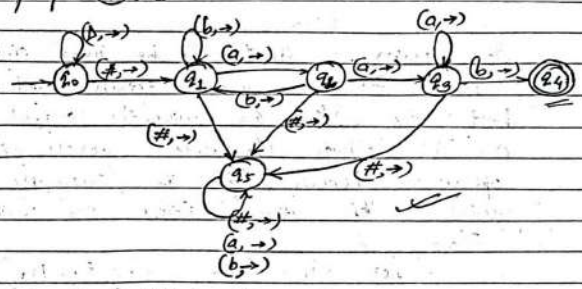
The transition table for δ be,

	a	b	#
q_0	(q_1, a, L)	(q_0, b, L)	–
q_1	(q_1, a, L)	(q_2, b, L)	–
q_2	(h, a, L)	(q_0, b, L)	$(h, \# N)$
h	–	–	Accept

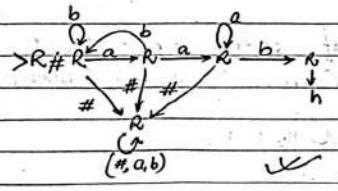
Where N – not specified move for halting.

Pb (11.) Write down the short hand Notation of turing machine accepting the Language $L = \{w \in (a,b)^* : w \text{ contains } aab\}$ as the pb (4). Where, the initial Configuration is $(q_0, \triangleright \# W \#)$.

Sol^n The short hand notation following the transition diagram of pb (4), i.e.



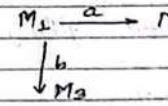
Its short hand Notation be:



M-20

Combining Turing Machines

Turing machines can be combined in the same way suggestive of the structure of finite automata. Following diagram shows the simple combination of Turing Machines:-



It operates - start in the initial state of M_1, operate as M_1 would operate until M_1 would halt; then, if the currently scanned symbol is an 'a' initiate M_2 and operate as M_2 would operate; otherwise, if the currently scanned symbol is 'b', then initiate M_3 and operate M_3.

Let $M_1 = (K_1, \Sigma, \delta_1, s_1, H_1)$

$M_2 = (K_2, \Sigma, \delta_2, s_2, H_2)$

& $M_3 = (K_3, \Sigma, \delta_3, s_3, H_3)$

Assuming, set of states are disjoint, combined machine $M = (K, \Sigma, \delta, s, H)$ where,

$K = K_1 \cup K_2 \cup K_3$

$s = s_1$

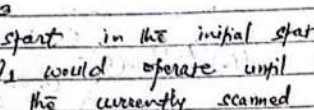
$H = H_2 \cup H_3$

$\forall a \in \Sigma$ & $q \in (K - H)$, $\delta(q, a)$ is defined as:

1) If $q \in (K_1 - H_1)$, then $\delta(q, a) = \delta_1(q, a)$
2) If $q \in (K_2 - H_2)$, then $\delta(q, a) = \delta_2(q, a)$
3) If $q \in (K_3 - H_3)$, then $\delta(q, a) = \delta_3(q, a)$.
4) Finally, if $q \in H_1$ - the only case remaining is $\delta(q, a) = s_2$ if $a = a$, $\delta(q, a) = s_3$ if $a = b$ and $\delta(q, a) \in H$ otherwise.

Other special Notations for Turing Machines:-

Ex-(1.)



or, $> R_1 \xrightarrow{a,b,\sqcup} R_1$

This diagram indicates,

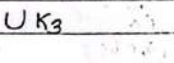
The machine represented by the diagram moves its head right one square if its content is an 'a' or 'b' or '∪' then it moves its head one square further right. The above may be written as,

$R \rightarrow R$

So the above machine becomes,

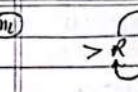
$RR \rightarrow R^2$

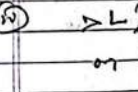
Ex-(11.)

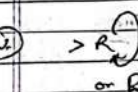


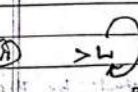
⇒ This indicates, the machine scans or runs its tape to the right until finds a-blanks.

$a \neq \sqcup$ is read "any symbol 'a' other than

(iii)  ⇒ $R_{\sqcup}$, which finds the first blank square to the right of the currently scanned square.
or $R_{\sqcup}$

(iv)  ⇒ $L_{\sqcup}$, which finds the first blank square to the left of the currently scanned square.
or $L_{\sqcup}$

(v)  ⇒ $R_{\overline{\sqcup}}$, which finds the first non-blank square to the right of the currently scanned square.
or $R_{\overline{\sqcup}}$

(vi)  ⇒ $L_{\overline{\sqcup}}$ which finds the first nonblank square to the left of the currently scanned square.
or $L_{\overline{\sqcup}}$

Ex:

(12) Design the Turing machine which erases all the a's in the tape. Use short hand notation.

Sol:-

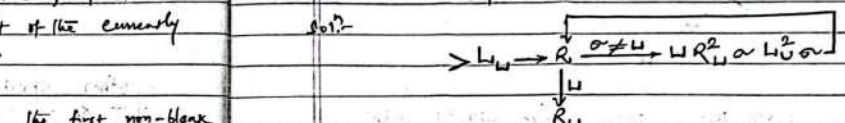
It operates as. [for "$\sqcup aa$"]

$\sqcup aa \vdash \sqcup\sqcup a \vdash \sqcup\sqcup\sqcup$

Ex (13) Design Turing Machine using the short hand notation which performs.

" Copying machine 'C', which transforms $\sqcup w \sqcup$ into $\sqcup w \sqcup w \sqcup$ with the alphabet $\Sigma = \{a, b, \sqcup\}$."

Sol:-



operation,

Let $w = ab$, then the above Turing machine should perform $\sqcup ab \sqcup$ into $\sqcup ab \sqcup ab \sqcup$. Which can be done as,

	Next loop
$\sqcup ab \sqcup \vdash \sqcup ab$	$\vdash^{R} \sqcup ab \sqcup a$
$\vdash^{R} \sqcup a b$	$\vdash^{\sqcup} \sqcup ab \sqcup\sqcup a$
$\vdash^{\sqcup} \sqcup\sqcup b$	$\vdash^{R^2_{\sqcup}} \sqcup a \sqcup\sqcup a \sqcup$
$\vdash^{R^2_{\sqcup}} \sqcup\sqcup b \sqcup\sqcup$	$\vdash^{\sigma} \sqcup a \sqcup\sqcup ab$
$\vdash^{\sigma} \sqcup\sqcup b \sqcup a$	$\vdash^{L^2_{\sqcup}} \sqcup a \sqcup\sqcup ab$
$\vdash^{L^2_{\sqcup}} \sqcup\sqcup b \sqcup a$	$\vdash^{\sigma} \sqcup ab \sqcup ab$
$\vdash^{\sigma} \sqcup ab \sqcup a$	

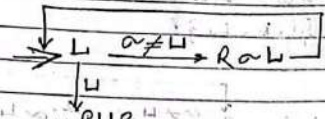
Next loop:- $\vdash^{R} \sqcup ab \sqcup ab$

$\vdash^{R_{\sqcup}} \sqcup ab \sqcup\sqcup ab \sqcup$

End

Pb (14.) Design Right shifting turing machine that performs ␣W␣ into ␣W␣W␣ turing short hand method.

Sol?-



Let us check this design for ␣ab␣ into ␣W␣ab␣.

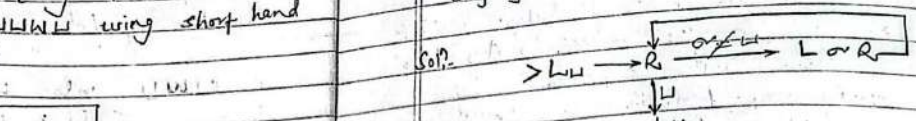
␣ab␣ ⊢ L ␣ab
⊢ R ␣ab␣ [No right symbol so blank ␣]
⊢ a ␣abb [␣ changed to b]
⊢ L ␣abb

Next loop:
⊢ L ␣abb
⊢ R ␣abb
⊢ a ␣aab
⊢ L ␣aab

Next loop ⊢ L ␣aab
⊢ R ␣aab
⊢ ␣ ␣␣ab
⊢ ␣ ␣W␣ab␣

Pb (15.) Use short handed method to design the left shifting machine that transforms ␣W␣W␣ into ␣W

Sol?-



Let us test this design for ␣W␣ab␣ into ␣a

␣W␣ab␣ ⊢ ␣ ␣W␣ab
⊢ R ␣W␣b
⊢ L ␣W␣ab
⊢ a ␣aab
⊢ R ␣a␣b

Next loop:
⊢ R ␣aab
⊢ L ␣a␣b
⊢ a ␣abb
⊢ R ␣abb

Next loop:-
⊢ R ␣abb␣
⊢ L ␣abb
⊢ ␣ ␣ab␣

Computing with Turing machine:-

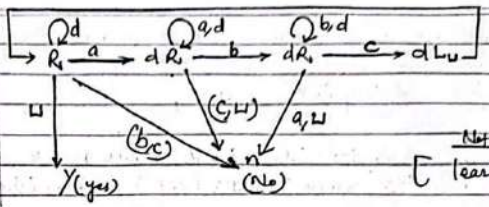
Let $M=(K, \Sigma, \delta, s, H)$ be a turing machines such that $H=\{y, n\}$ consists of two distinguished halting states ('y' and 'n' for 'yes' and 'No'). Any halting Configuration whose state component is 'y' is called accepting configuration and 'n' is rejected one. We say that M accepts an input $w \in (\Sigma - \{\sqcup, \triangleright\})^*$ if $(s, \triangleright \sqcup w)$ yields an accepting configuration and M rejects 'w' if $(s, \triangleright \sqcup w)$ yields a rejecting Configuration.

The turing machine decides the language L if, when started with input 'w', it always halts and does so in a halt state that is the correct response to the input; 'y' (yes) if $w \in L$, 'n' (No) if $w \notin L$. No guaranteed are given if inputs are blanks and left end symbols.

Example:-

(1) Design a Turing machine that recognizes the Language given by $L = \{a^n b^n c^n : n \geq 0\}$ with the alphabet $\Sigma = \{a, b, c, \sqcup\}$

Soln:- Let us introduce a new symbol 'd' in the alphabet which help us to prepare $a^n b^n c^n$. Then it is easy to write short-hand machine to get $L = \{a^n b^n c^n : n \geq 0\}$ be:



let us check this design for ⊔abc

		Next loop
⊔abc	⊔a bc	R ⊔ddd
d	⊔d bc	R ⊔ddd
R	⊔d b c	R ⊔ddd
d	⊔dd c	R ⊔ddd
R	⊔dd c	R ⊔dddu
d	⊔ddd	y ⊔ddd y Accepted.
⊔u	⊔ddd	

Again, let us check for ⊔aabc

		Nex loop:
⊔aabc	⊔a abc	R ⊔dadd
d	⊔d abc	R ⊔da dd
R	⊔d a bc	d ⊔ddddd
R	⊔da bc	R ⊔dddd
d	⊔dadc	R ⊔dddd
R	⊔dad c	R ⊔ddddu
d	⊔dadd	n ⊔ddd n rejected
⊔u	⊔dadd	

pb (16) Construct a turing machine that computes the following function.

$$f(n,m) = n+m$$

Sol: Let I represents 1, #II# represents 2 and so on. We assume that input numbers (that is n and m) are written on the tape separated by #. That is if we want to add 2 and 3, the input string will be # II # III #.

Now we want to such a turing machine which gives output as follows:

$f(2,3) = 2+3 = 5$

i.e.

$$\#II\#III\# \vdash \#IIIII\#$$

When we analyze the output then it becomes clear that for constructing turing machine for $f(n,m) = n+m$, we have to remove #, which separates n & m.

In fact, resulting turing machine works as a left-shift machine (say SL). It shift the each symbol of m portion of input by one cell towards left.

Let turing machine is

$$Tm = (Q, \Sigma, \Gamma, \delta, q_0, h)$$

$$Q = \{q_0, q_1, q_2, q_3, q_4, h\}$$

$$\Sigma = \{I\}$$

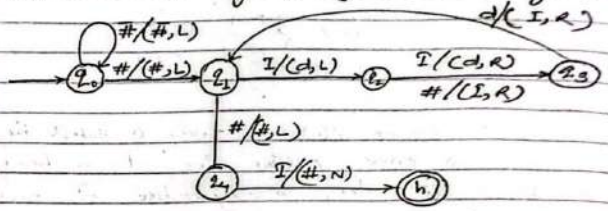
$$\Gamma = \{I, \#\} = \Sigma^*_0$$

[# → other than symbol]

$s = q_0$ is initial state

$H = \{h\}$ is halting state.

and δ is defined by following transition diagram.



In the above turing machine we are not showing the moves form every state and those move for input symbols are considered as path (transition) available for some inputs and machine rejects the input or not processed if it is not halting at h.

Check: $2+2 = 4$ [# II # II #]

$\#II\#q_0 \vdash \#II\#Iq_0I$

$\vdash \#II\#q_0II$

$\vdash \#II\#q_1II$

$\vdash \#II\#Iq_1II$

$\vdash \#II\#q_2dI$

$\vdash \#IId q_3I$

$\vdash \#IIIIq_1$

$\vdash \#IIIq_2d$

$\vdash \#IIId q_3$

$\vdash \#IIIId q_4\#$ $\vdash \#IIII\#h$

(11.) Design a Turing machine M1 prove that the following function is turing computable.

$$f(n) = \begin{cases} n-1, & n>0 \\ 0, & n=0 \end{cases}$$

Sol? If function $f(n)$ is turing computable then there exist a turing machine for it. so basically we have to design a turing machine for function $f(n)$. This turing machine should give output ## for # # (n=0) and # II# for # III # (n=3)

Let the turing machine be

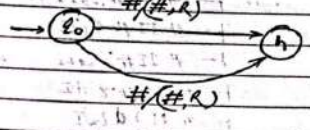
$T_m = (Q, \Sigma, \delta, q_0, h)$

$Q = \{q_0, h\}$

$\Sigma = \{\#, I\}$

q_0 is initial state and h is halting state.

Transition relation δ is defined by following transition diagram:



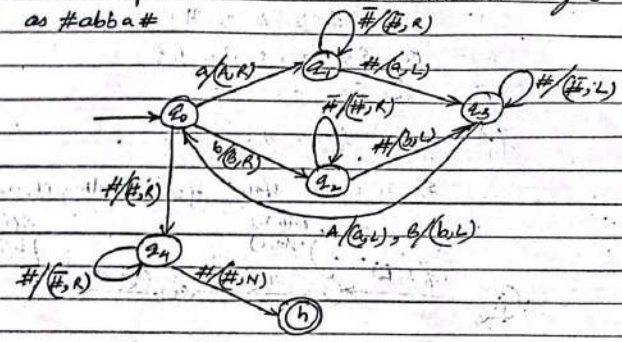
Let us process the following string: $w = \#III\#$

$\#III q_0 \vdash \#III\#h$ ✓

for string $\#q_0\# \vdash \#\#h$ ✓

(12.) Design a Turing machine which computes following function $f(w) = ww^R : w \in \{a,b\}^*$.

Sol? Let the input is # ab # then Tm should give output as #abba#



process for $\#ab q_0 \# \vdash \#ab\#q_2$ ✓

$\vdash \#abBq_3b$

$\vdash \#aq_4bb$

$\vdash \#Aq_1bb$

$\vdash \#Abq_1b$

$\vdash \#Abbq_1$

$\vdash \#Abb\#q_1$

$\vdash \#Abbq_2\#a$

$\vdash \#Abq_3ba$

$\vdash \#Aq_3bba$

$\vdash \#q_3Abba$

$\vdash \#q_0abba$

$\vdash \#aq_4bba$

$\vdash \#abq_4ba$

$\vdash \#abbaq_4$

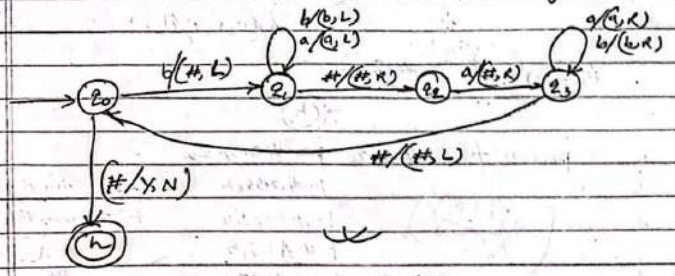
$\vdash \#abba\#q_4$

$\vdash \#abba\#h$.

OK.

(19) Design a Turing machine for the following Language
$L = \{a^n b^n \text{ for } n \geq 1\}$

Sol:- Let
$T_M = (Q, \Sigma, \delta, z_0, H)$
where $Q = \{z_0, z_1, z_2, z_3, h\}$
$\Sigma_0^* = \{a, b\}$
$\Sigma_0 = \{a, b, \#\}$
$s = z_0$ initial state, $H = \{h\}$ halting state



processing of $\omega = \#ab\#$

$\#ab z_0 \vdash \# a z_1 \#$
$\vdash \# z_1 a \#$
$\vdash \# a z_2 \#$
$\vdash \#\# z_3$
$\vdash \#\# z_0 \#$
$\vdash \# Y h \#$

Recursive function:-

Let $M = (K, \Sigma, \delta, s, \{h\})$ be a turing machine, let $\Sigma_0 \subseteq \Sigma - \{\cup, \triangleright\}$ be an alphabet and $\omega \in \Sigma_0^*$. Suppose that M halts on input ω i.e. $(s, \triangleright \sqcup \omega) \vdash_M^* (h, \triangleright \sqcup y)$ for some $y \in \Sigma_0^*$. y is called the output of M on input ω and is denoted by $M(\omega)$. Notice that $M(\omega)$ is defined only if M halts on input ω and in fact does so at a configuration of the form $(h, \triangleright \sqcup y)$ with $y \in \Sigma_0^*$.

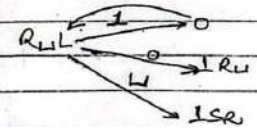
Now let f be any function from Σ_0^* to Σ_0^*. We say that M computes f if for all $\omega \in \Sigma_0^*$, $M(\omega) = f(\omega)$. i.e. for all ω, M eventually halts on input ω and when it does halt, its tape contains the string $f(\omega)$.

Hence, A function f is called recursive if there is an turing machine M that computes function f.

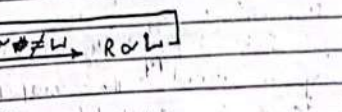
Ex

Pb (20) Design a Turing machine that computes the successor function $succ(n) = n+1$ with alphabet $\Sigma = \{0, 1, \sqcup\}$. Hence test your design for $\sqcup 11 \sqcup$ to $\sqcup 100 \sqcup$.

Sol:- We can design the following turing machine computes successor function.



Where S_R is right shifting machine given by,



This machine first finds right end of the input and then goes to the left as long as it sees 1's changing all of 1's to 0's. When it sees a '0', it changes it into a '1' and halts. If it sees a 'B' machine writes 1 in place of B and shifts the whole string one position right and halts.

We can test this design for given input B11B as:

$$B11B \vdash_{R} B11B$$
$$\vdash_{L} B11\underline{1}B$$
$$\vdash_{0} B10B$$
$$\vdash_{L} B\underline{1}0B$$
$$\vdash_{0} B00B$$
$$\vdash_{L} \underline{B}00B$$
$$\vdash_{1} \underline{1}00B$$
$$\vdash_{S_R} B100B$$

Recursively Enumerable Language (RE)

Let $T = (Q, \Sigma, \delta, z_0, H)$

$\Sigma_0 \subseteq \Sigma - \{\triangleright, \#\}$ be alphabet

$L \subseteq \Sigma_0^*$ be a language.

Then, T semi-decides L if for any $w \in \Sigma_0^*$, $w \in L$ if and only if 'T' halts on input w.

→ "A Language L is recursively enumerable if and only if there is Turing machine T, that semi-decides it."

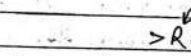
This means turing machine 'T' halt for every input, $w \in L$ but not guranteed to halt for $w \notin L$.

Ex.

(1.) Let $L = \{w \in \{a,b\}^* : w \text{ contains at least one } a\}$

Sol'

Then, we can design a turing machine for L as,



This machine scans to the right to find one a. If no a is found, it goes forever (infinite loop), never halting. It halts only if there is at least one 'a'. Therefore given language L is semi-decided hence recursively enumerable.

Note: [L_0, RE are, Undecidable, Turing-acceptable, Semi-decidable, partially decidable]

[$\overline{A}$ = Language]

Not Recursively Enumerable:- (Not partially decidable)

——×——>——×—— (Not partially decidable)

Not guaranteed to halt for any input.

Ex- A = CFL's that contains every thing
CFL, $L = \Sigma^*$ [$\overline{A}$ is not RE]

A = T.M. that halt on every input
[$\overline{A}$ is also not RE]

Features:- Characteristics (of R & RE Languages)

1. If A is RE then $\overline{A}$ is (not RE)
2. If A is Recursive, $\overline{A}$ is Recursively Enumerable
3. If A & $\overline{A}$ both are Recursively enumerable, then A is Recursive.
4. A language is Recursively enumerable if and only if it is turing enumerable (i.e. enumerator exist for that Language).
5. A language is Recursive iff it is lexicographically turing-enumerable.

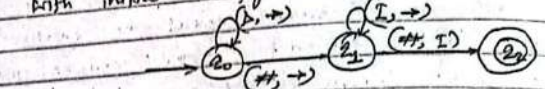
[Not- proof are done in Chapter 5]

pb 6. Construct T.M that computes $f(n) = n+1$ Cano

Let $T = (Q, \Sigma, z_0, \delta, h)$

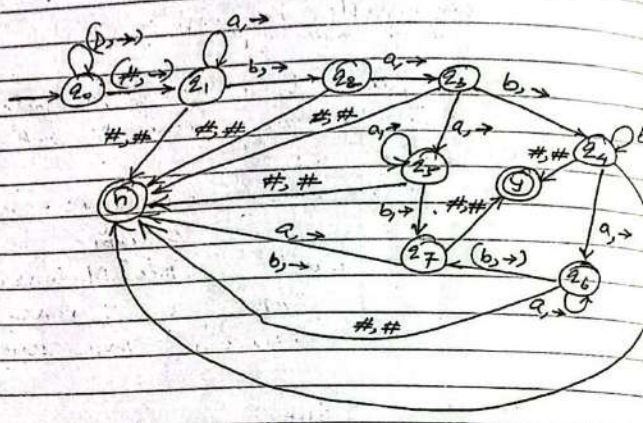
Where $\Sigma = \{\Delta, \#, I\}$ $z_0 = \{I\}$, $L \subseteq \Sigma$

with initial configuration $(z_0, \Delta \# w)$ & $h = \{$



pb A1. Construct TM that decides for i... Regula... expression as, $(a^*ba\,b^*a^*b)$ [via concept... DFA]

Soln



Say, $T = \{Q, \Sigma, \delta, z_0, h\}$

$\Sigma = \{\Delta, a, b, \#\}$, $z_0 = \{a, b\}$ $w \in \Sigma^*$ ...

$L \subseteq \Sigma^*$ let the starting configuration ... $(z_0, \Delta \# w)$, ...

* Recursive Language :-

A Language is recursive if there exist a turing machine that accepts every string of the Language and rejects every strings that are not in the Language.

→ So, we are sure about the rejection of strings which are not in the given Language.

→ Recursive Language always halts the Turing machine.

→ Recursive Language is subset of recursively enumerable Language $R \subseteq RE$.

* Turing recognizable Language :-

A Language L is turing recognizable if there exists a turing machine M such that for all strings w:

→ If $w \in L$, eventually M enters & accept

→ If $w \notin L$, either M enters & reject or M never terminates (infinite looping).

* Turing decidable Language :-

A Language L is turing decidable if there exists a turing machine M such that for all strings w:

→ If $w \in L$, M enters & accept

→ If $w \notin L$, M enters & reject.

Hence,

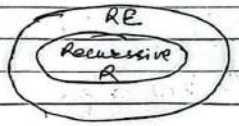
* Decides – always terminates
* Recognizes – Can run forever, then ~~it~~ without deciding

Theorem: "If a Language is recursive, then it is recursively enumerable".

proof:

A Language is recursively enumerable if there exists a turing machine that accepts every strings of the Language and a Language is recursive if there is a turing machine that accepts every string of the Language and does not accept a strings that are not in the Language.

So, every recursive Language is also recursively enumerable. Hence, the given statement is true. We can observe this concept with following diagram.



Hence proved