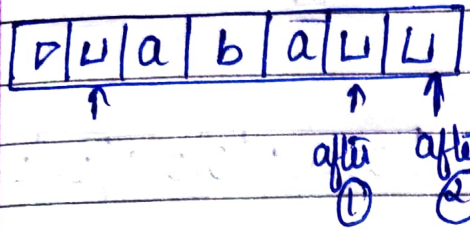
Date : / /
Page No.

⑥ Turing Machine

① $>R^2_{\sqcup}$ | $>R_{\sqcup}R_{\sqcup}$] two components
① ②

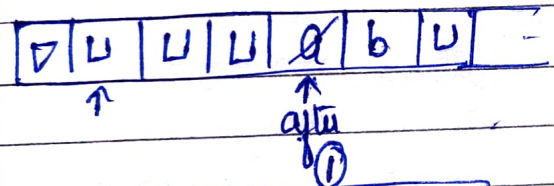
I/P tape

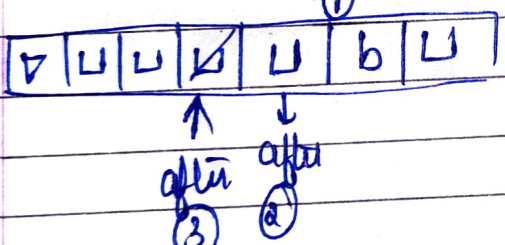


② $>R^{\sqcup}_{①}$ $\xrightarrow[a]{x \neq \sqcup}$ $\sqcup L x$ (② ③ ④)

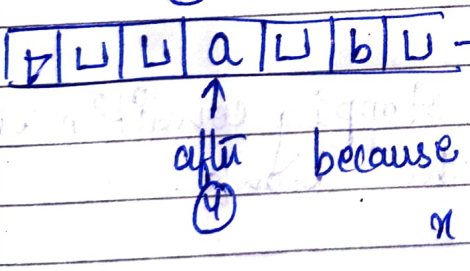
→ this is the condition that apply on the current tape square.

Now,
$x \neq \sqcup$
LHS





Variable (var) that hold the content of current tape square.



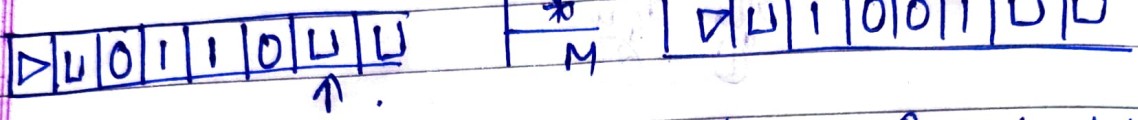
because currently the value stored in x is 'a'.

Date : / /
Page No.

Q Create a machine that performs a function that shows the one's complement of the given word.

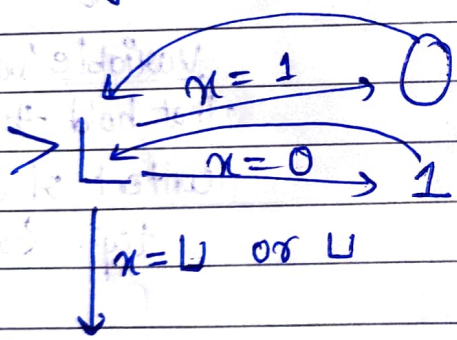
→ shows the several steps performed

Initial configuration. $\triangleright \sqcup w \sqcup \vdash^{*}_{M} \triangleright \sqcup w' \sqcup$



we are scanning the string from right to left because we specified the initial configuration at the left most cell.

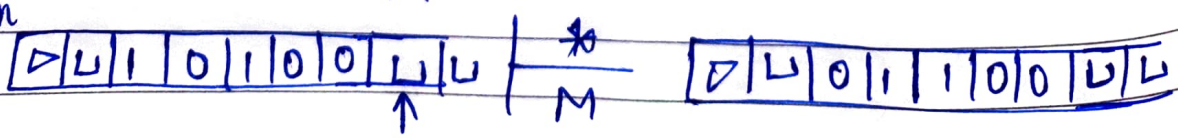
→ Turing Machine



$R_{\sqcup}$ → This is the stopping condition when you find the blank.

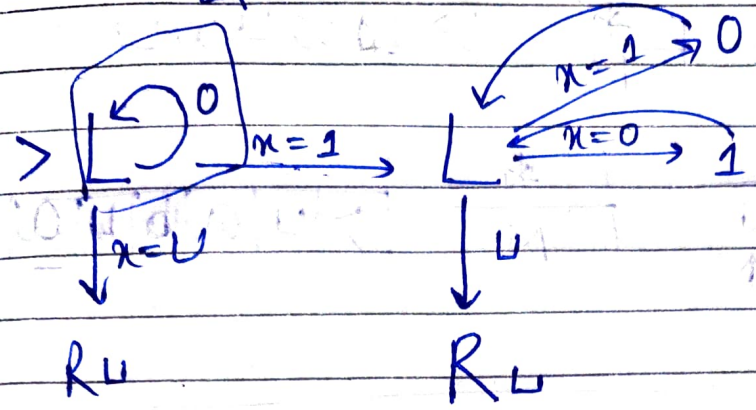
Q Create a machine for two's complement of the given word.

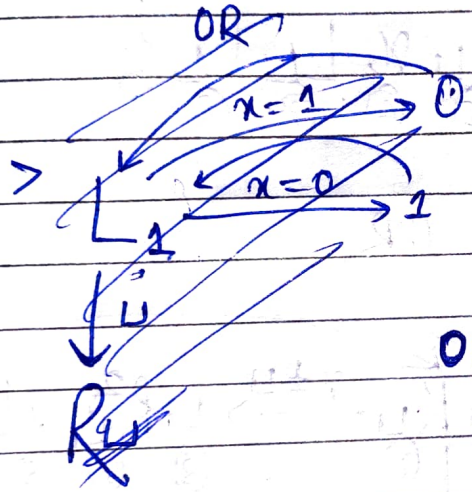
Initial Configuration $\triangleright \sqcup w \sqcup \vdash^{*}_{M} \triangleright \sqcup -w \sqcup$



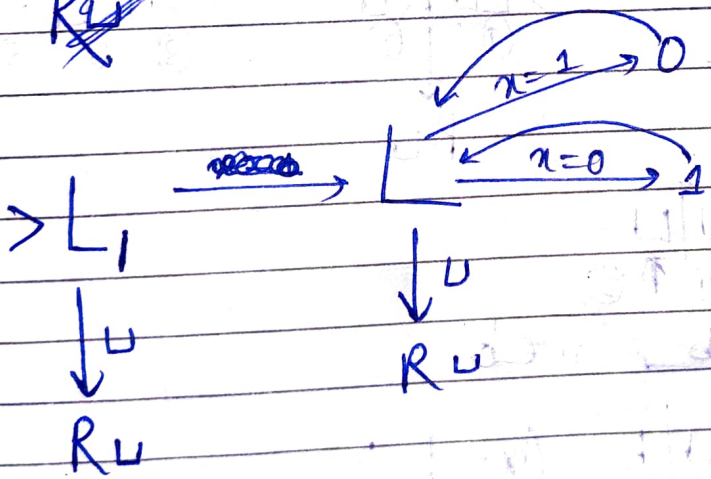
Date : / /
Page No.

→ Turing Machine





OR

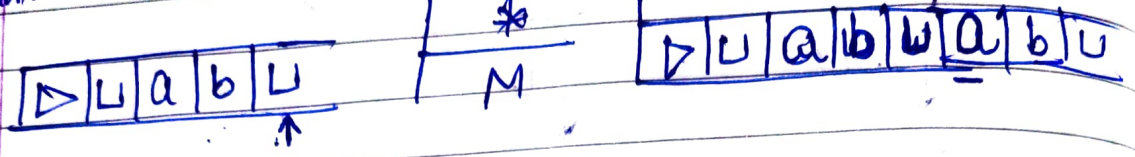


Date : / /
Page No.

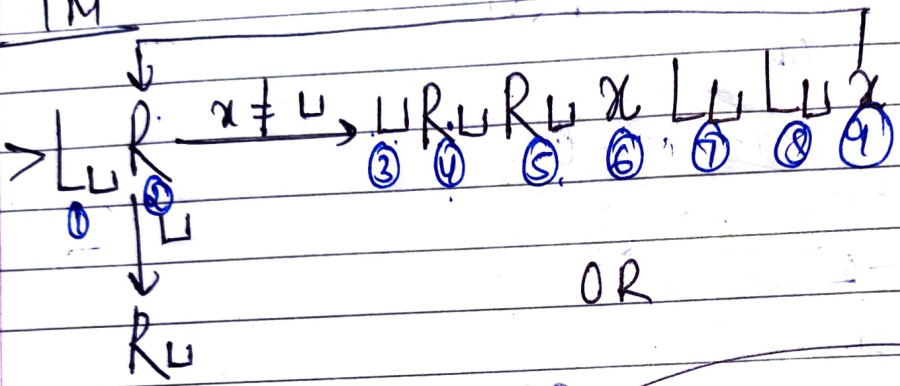
Turing Machine

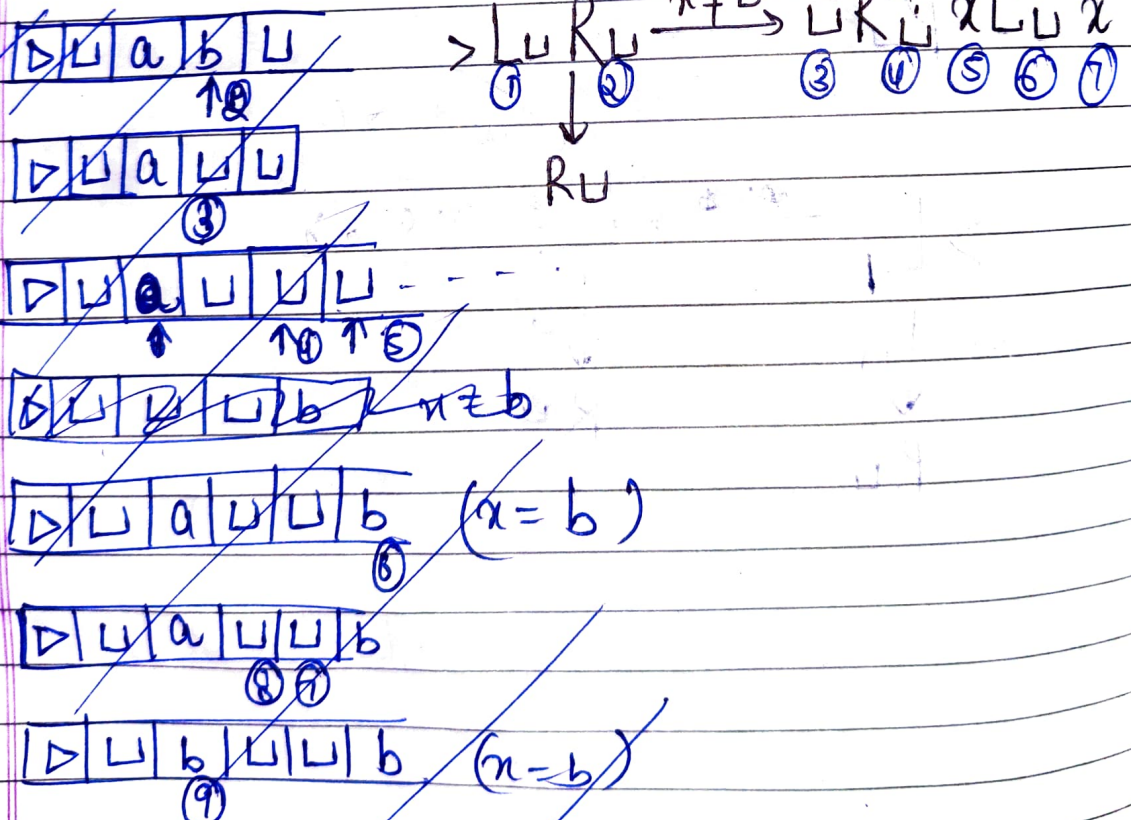
Current Configuration.

$\triangleright \sqcup w \sqcup \overset{*}{\underset{M}{\vdash}} \triangleright \sqcup w \sqcup w \sqcup$

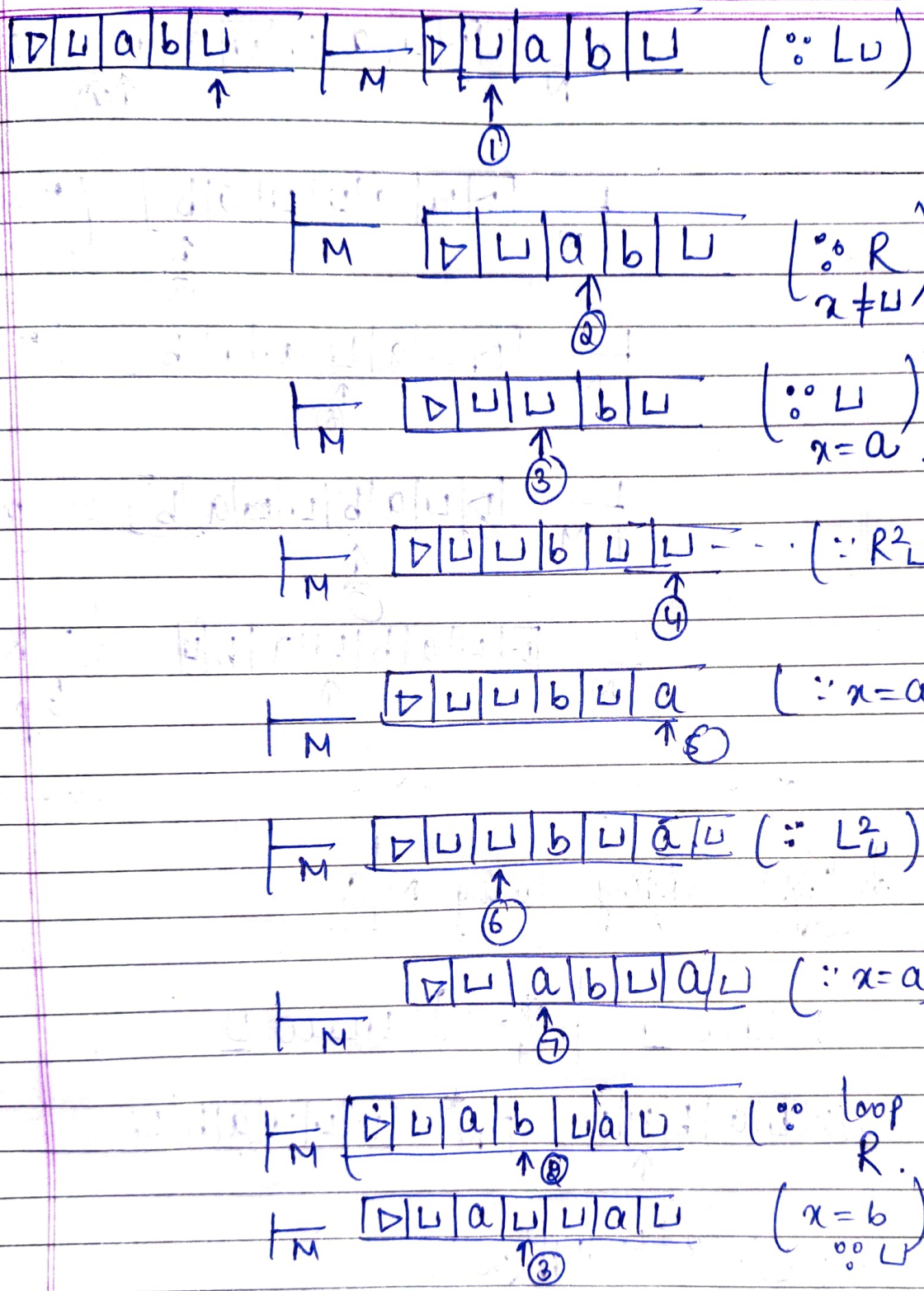


→ TM





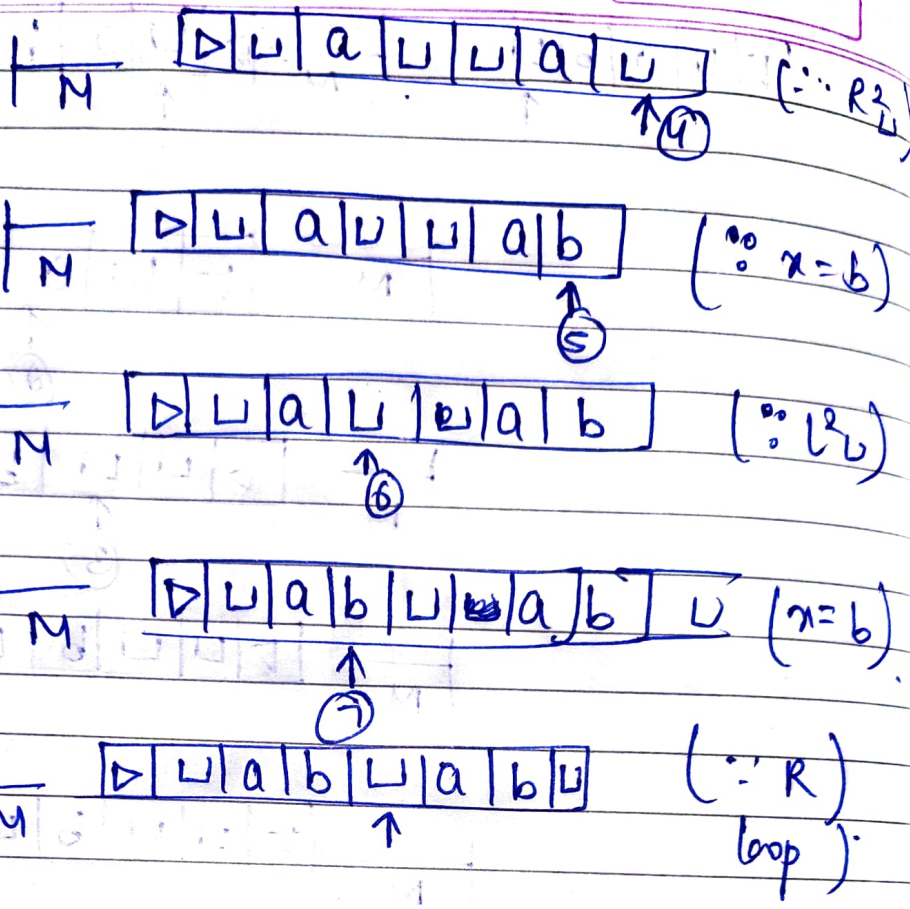
Date : / /
Page No.



Conclusion: This machine is making a copy of the given word i.e.

$$\sqcup w \sqcup \overset{*}{\underset{N}{\vdash}} \sqcup w \sqcup w \sqcup$$

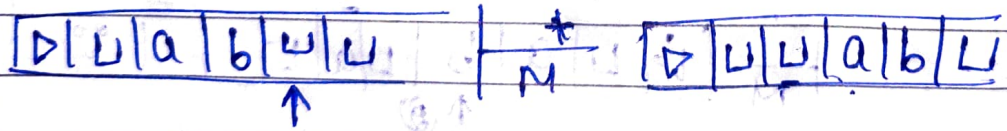
Date : / /
Page No.

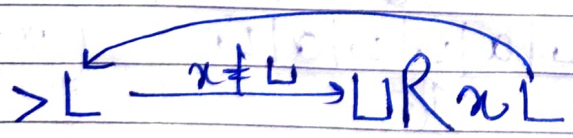


x x x

* Right Shifting Turing M|C

$\triangleright \sqcup \omega \underline{\sqcup} \vdash_M^* \triangleright \sqcup\sqcup \omega \underline{\sqcup}$

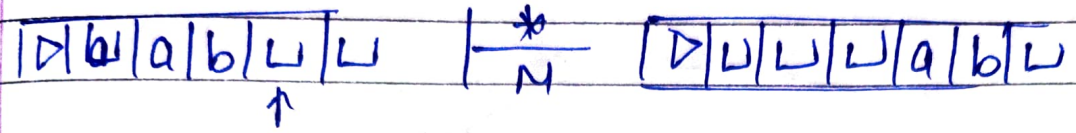


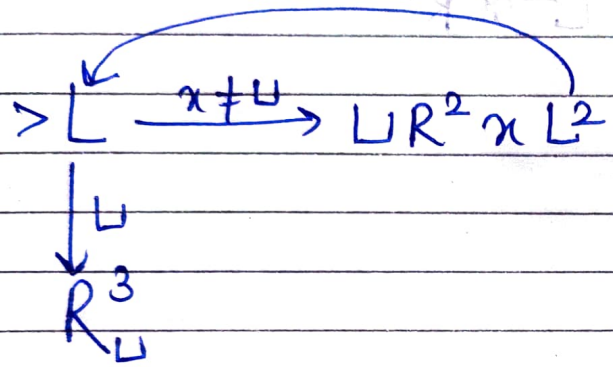


Date : / /
Page No.

* Double Right Shifting Turing M/c

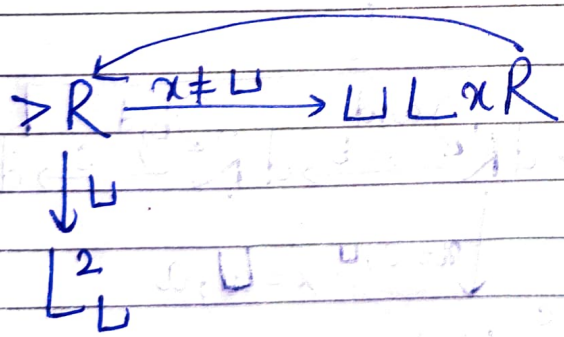
$\triangleright \sqcup w \sqcup \vdash_M^* \triangleright \sqcup\sqcup \sqcup w \sqcup$





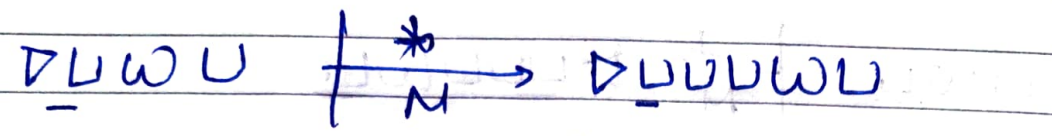
* Left Shifting Turing M/c

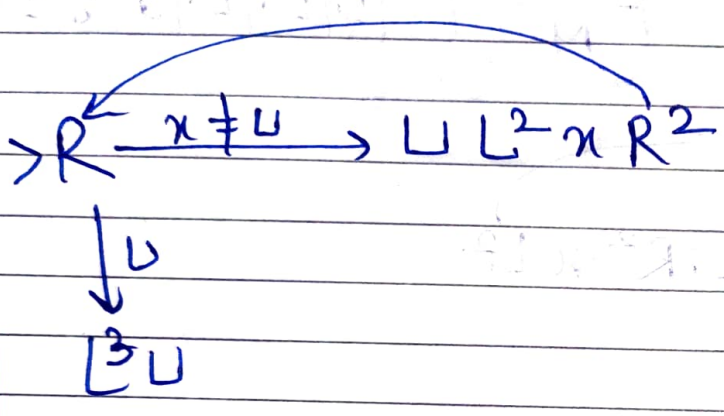
$\triangleright \sqcup w \sqcup \vdash_M^* \triangleright \sqcup\sqcup w \sqcup$



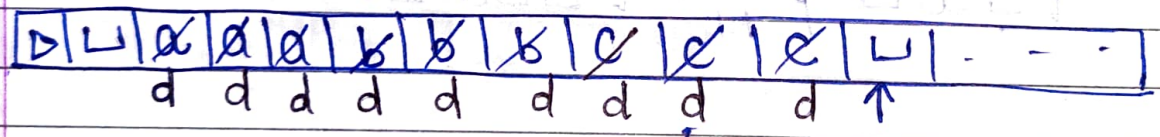
Date : / /
Page No.

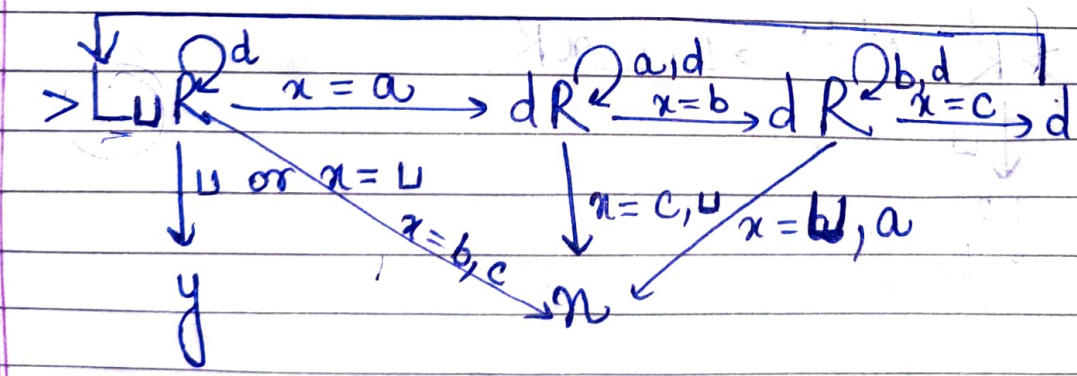
* Double left shifting Turing M/c.





* Turing m/c for $a^n b^n c^n$,



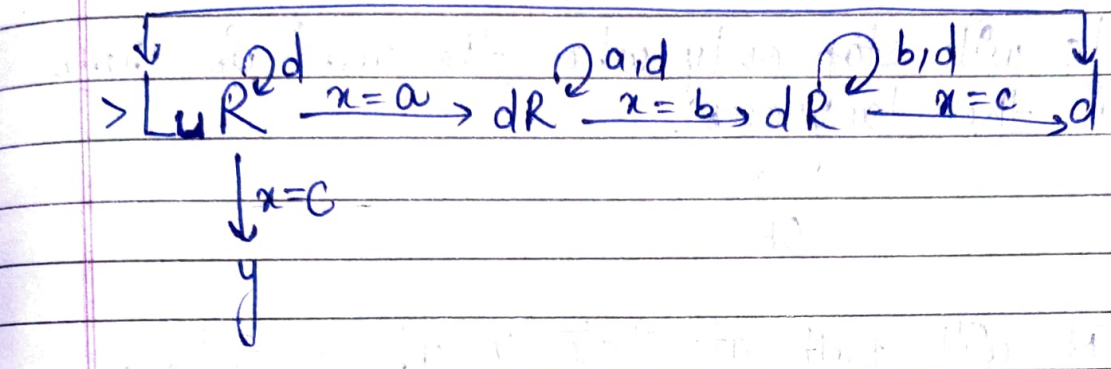


Date : / /
Page No.

Ques ~~$a^n b^n c^n$~~
~~$a^{2n} b^{2n} c^{2n}$~~

Ques $a^n b^n c^{n+1}$

check ?



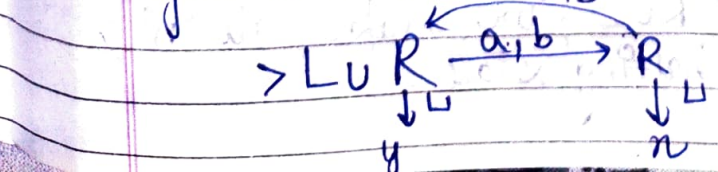
* Turing Machine (TM) decides L (TM will halt for all possible words)

$w \in L \longrightarrow$ yield y

$w \notin L \longrightarrow$ yield n

→ If we are able to create a machine and if it accepts all possible words, then it will halt or stops at state y. and if it does not accepts any words that is not in languages it gets to n state. This is known as Recursive languages.

eg. $w \in \{a,b\}$ having even length



Date : / /
Page No.

* Recursively enumerable lg.

↳ TM semidecides L

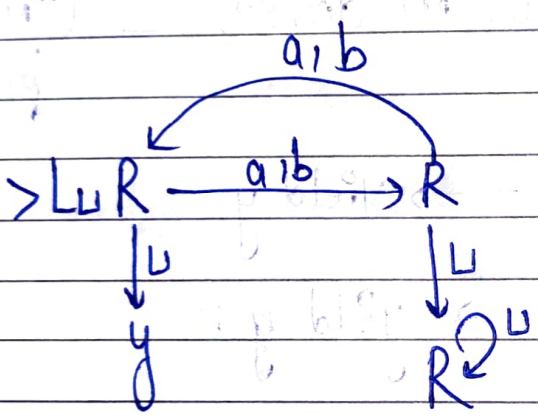
→ m/c will stop only when the word is there in the languages.

OR

→ TM will halt at state y iff $w \in L$.

Note: If $w \notin L$, the TM will not halt.

eg.



* Halting problem → here, we don't know when machine will stop or not.

Ques: All Recursively lang. are Recursively enumerable language?

→ Yes.

→ for making RL to REL, we make all 'n' state to non-halting state.