— THEORY OF COMPUTATION —

MODULE - 5

PART - A :

① Construct a Turing machine that accepts the language $L = \{a^{2n} b^n \mid n \geq 0\}$. Give the transition diagram for the TM.

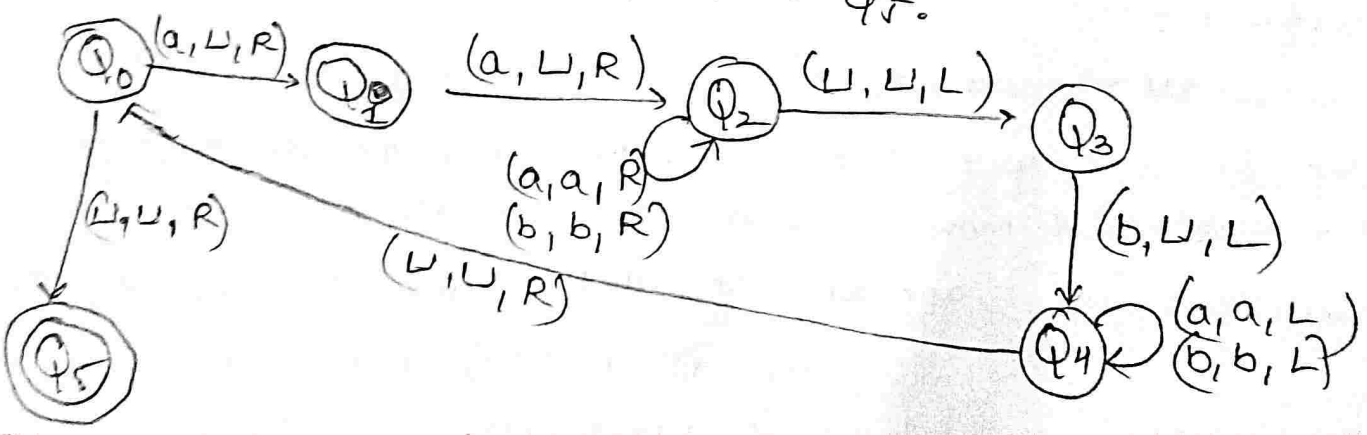
A) Procedure:

i) As we know the string will have twice number of 'a' in starting than number of 'b's. So, we will first make the first two a's Blank (B) and go from state Q_0 to Q_1, and from state Q_1 to Q_2

ii) After making them blank, we will traverse to the end of the string till we get the rightmost 'b' in state Q_3 and make it blank reaching state Q_4.

iii) Now, we will traverse back till we get the left most 'a' in string and return to state Q_0 from Q_4

iv) At a certain point after repetetions, if the string belongs to the language then it will be left empty and hence get accepted at state Q_5, which is final state. If the final string is empty it will also get accepted at Q_5.



Transition Table :

States	a	b	⊔
Q_0	$(Q_1, \sqcup, R)$	-	$(Q_5, \sqcup, R)$
Q_1	$(Q_2, \sqcup, R)$	-	-
Q_2	(Q_2, a, R)	(Q_2, b, R)	$(Q_3, \sqcup, L)$
Q_3	-	$(Q_4, \sqcup, L)$	-
Q_4	(Q_4, a, L)	(Q_4, b, L)	$(Q_0, \sqcup, R)$
Q_5	-	-	-

---

(2) Construct a Turing Machine that gives two's complement for given binary number.

*) The logic we will be following here is,

- We will be assuming that the input starts with a blank space,

for example,

⊔ 0 1 0 0 0 0 1 0 0 ⊔⊔⊔....

Procedure :

i) We move all the way to the right, skipping all 0's and 1's

ii) When we end up on blank (⊔), then move one step to left.

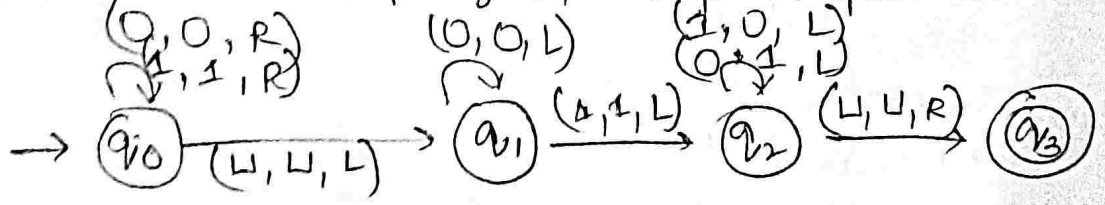
iii) Then we will move to left skipping all 0's.

iv) When the first 1 appears, we skip it, and move one step left.

v) From now on, we will make all 1's to 0's and 0's to 1's.

vi) We repeat until we reach blank (⊔).

vii) Then, move one step right to stop on first character.



for the above example Output would be,

⊔ 1 0 1 1 1 1 1 0 0 ⊔⊔...

③ Examine Type 3 and Type 2 grammars with examples

④

④ Extend Type 1 and Type 0 grammars with examples.

→ ⑧ Both of the above questions answers merged. ⑧

i) Type 3 Grammar: [Regular Grammar]

- They generate regular languages
- They must have a single non-terminal on left-hand side and a right-hand side and consisting of a single terminal or single terminal followed by a single non-terminal.
- The productions must be in the form, $X \to a$ (or) $X \to aY$ where $X, Y \in N$ (non-terminal) and $a \in T$ (terminal)

Ex: $X \to a \mid aY$; $Y \to b$

ii) Type 2 Grammar: [Context-Free Grammar]

- They generate context-free languages.
- The production must be in form $A \to \gamma$ where $A \in N$ (Non Terminal) and $\gamma \in (T \cup N)^*$ (String of terminals and non-terminals)
- These languages are recognized by a non-deterministic pushdown automata.

Ex:
$S \to Xa$
$X \to a$
$X \to aX$
$X \to abc$
$X \to \varepsilon$

iii) Type 1 Grammar: [Context Sensitive Grammar]

- They generate context-sensitive languages.
- The productions must be of the form,

$\alpha A \beta \to \alpha \gamma B$, where $A \in N$ (Non Terminal)

$\alpha, \beta, \gamma \in (T \cup N)^*$ (String of Terminals and Non-Terminals)

- The strings α, β may be empty, but γ must not be empty.
- These languages are recognized by a linear bounded automata.

Ex:
$AB \longrightarrow AbBc$
$A \longrightarrow bcA$
$B \longrightarrow b$

iv) Type 0 Grammar: [Recursively Enumerable / Unrestricted]

- They generate recursively enumerable languages.
- They have no restrictions
- They are recognised by Turing machines.
- The productions can be in form of $\alpha \rightarrow \beta$, where
  - α is a string of terminals and non-terminals with at least one non-terminal and α cannot be null.
  - β is a string of terminals and non-terminals.

Ex: $S \rightarrow ACaB$
$Bc \rightarrow acB$
$CB \rightarrow DB$
$aD \rightarrow Db$

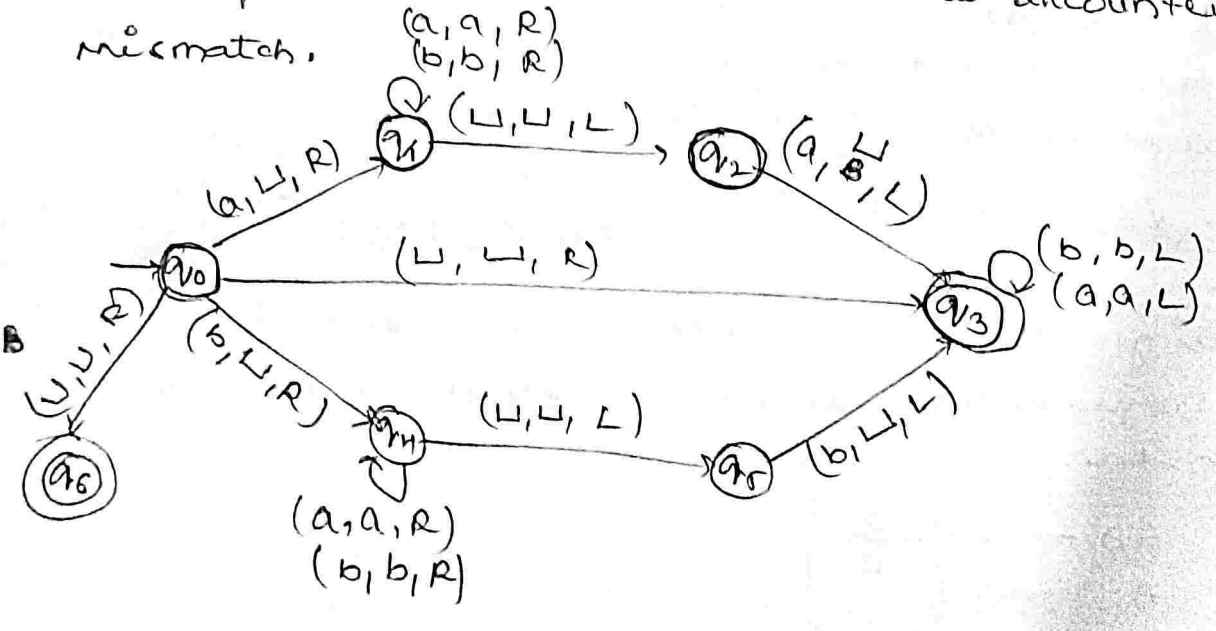
---

(5) Design a Turing Machine that accepts the sets of all even palindrome over $\{0,1\}$

*) Procedure:

i) Match the first and last element and erase them and go on doing the same. Once we reach epsilon without any mismatch then the string is even-length palindrome

ii) For this, a TM is defined after the machine runs and erases first and last element without encountering a mismatch.



(6) Design Turing Machine for $L = \{a^n b^n c^n \mid n \geq 1\}$

A) Procedure:

i) Replace first 'a' with 'X'

ii) Move right, skipping all 'a's, until 'b' is found.

iii) Replace first 'b' with 'Y'

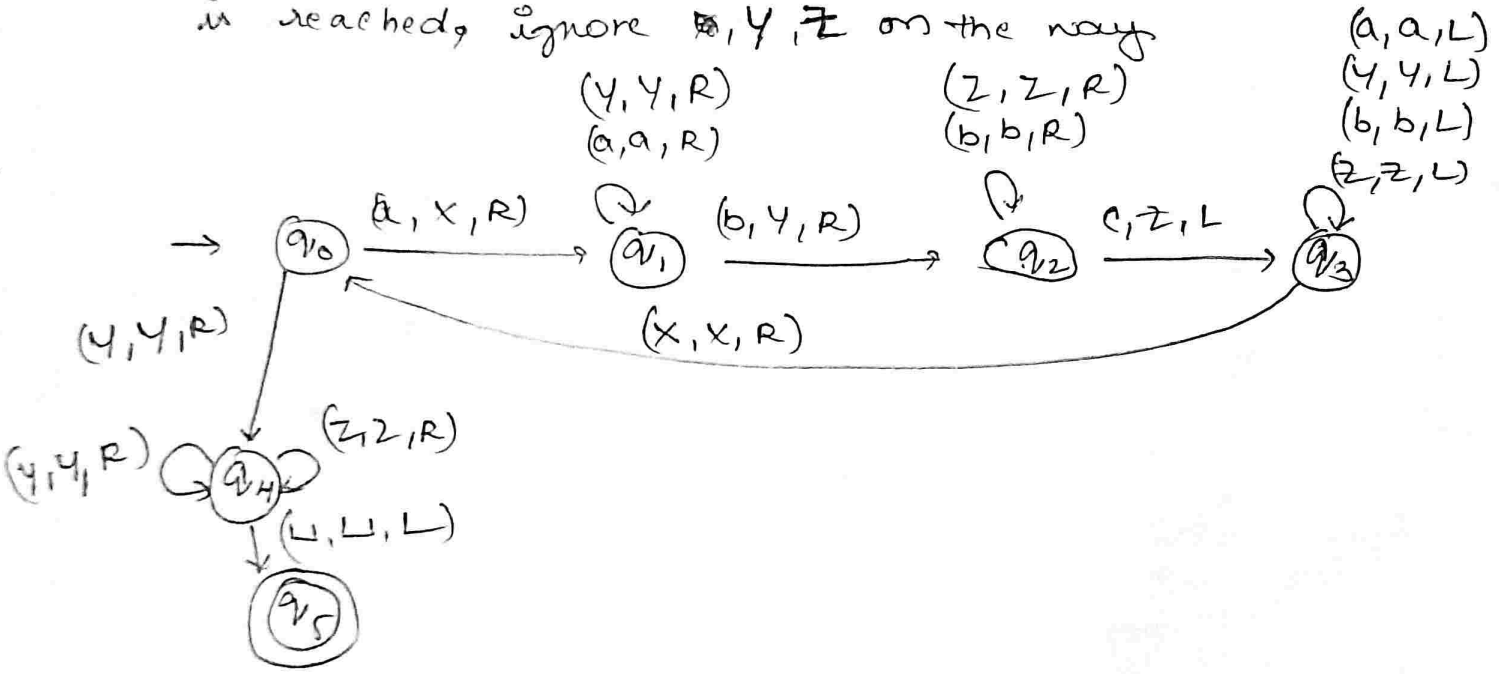
iv) Move right, skipping all 'b's, until 'c' is found.

v) Replace first 'c' with 'Z'

vi) Move ~~left~~ left, until reach 'X', move right one step.

vii) Goto, first state again.

viii) Now, continue until all 'a', 'b', 'c' are replaced, and blank is reached, ignore X, Y, Z on the way

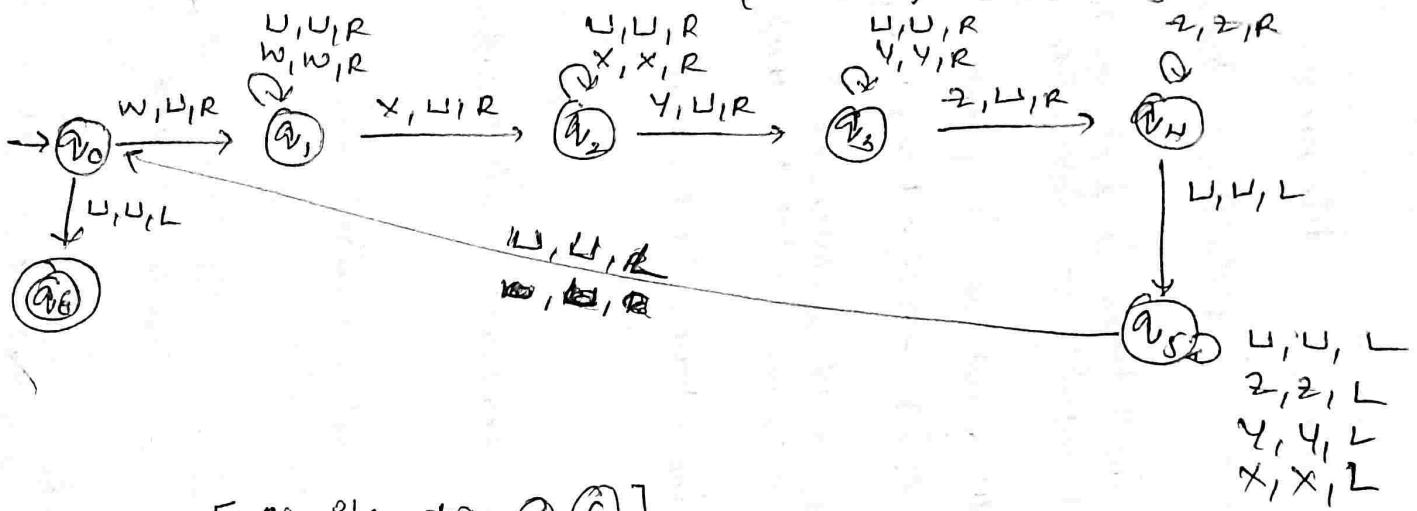


ix) Now, after all replacements you will reach 'X', but there is no 'a' to move to next transition. So, a new state is introduced, and is transitioned by ignoring all 'Y' and 'Z' along the way to final state (q_5).

⑧ Compare and Contrast the FSM, PDA and Turing Machine.

BASIS OF COMPARISION	FINITE AUTOMATA	PUSHDOWN AUTOMATA	TURING MACHINE
• Computational power	Low Computational Power	Higher Computational power than FA	High Computational Power
• Input length	Finite on both sides	Finite on both sides	The input tape is finite on left but infinite on right side.
• Language recognition	Regular languages	Context free languages	All kinds of languages
• Transition function	$\delta : Q \times \Sigma \rightarrow Q$	$\delta : Q \times (\Sigma \cup \{\in\}) \times \Gamma \rightarrow Q \times \Gamma$	$\delta : Q \times T \rightarrow Q \times T \times \{L, R\}$.
• States	Finite set of states, Finite set of input symbols.	A finite set of states, input symbols and stack.	A finite set of states, input symbols, infinite tape and stack.
• Memory	Minute amount of memory, precise of size of a state	A stack data structure is used with a FSM, basically unlimited memory	An input tape, Stack, FSM.
• Head operations	Only read, no write operation	Only read, No write operation	Both read and write operation
• Head movement	Only one way movement	Only one way movement	Both left and right directional movement.
• Designing	It is quite easy to design a FSM.	It is fairly difficult to design a PDA.	It is highly difficult and as complex to design a TM.
• Acceptance	Accepts input alphabets by going up to final states	Accepts input alphabets by final state of an empty stack.	Accepts input tape by final state of empty stack

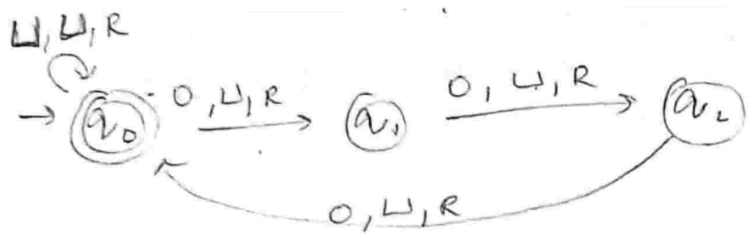
(9) Construct a TM to accept $L = \{ w^n x^n y^n z^n \mid n \geq 1 \}$



Procedure : [Similar to (8) (6)]

---

(10) Design a Turing machine that accepts the language denoted by regular expression $(000)^*$.



Procedure : [write steps in detail]

(4) PART-B:

(1) Describe short notes on Context sensitive language and linear bounded automata.

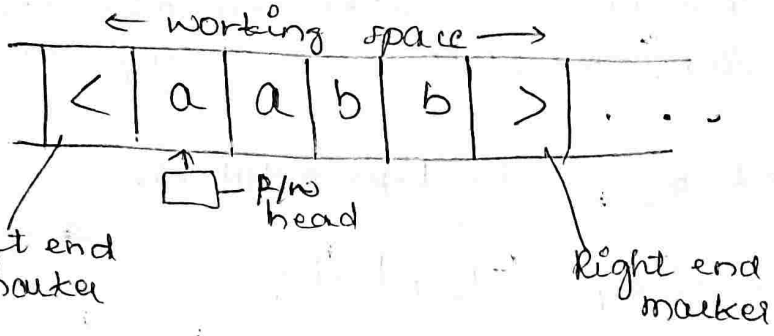
A) for notes on Context sensitive language, refer PART A-Q3.

- Linear bounded automata:

A Turing machine that uses only the tape space occupied by the i/p string is called as LBA. A LBA is described by 8 tuple notation ie

$$M = (Q, \Sigma, \gamma_0, \delta, F, \Gamma, <, >)$$

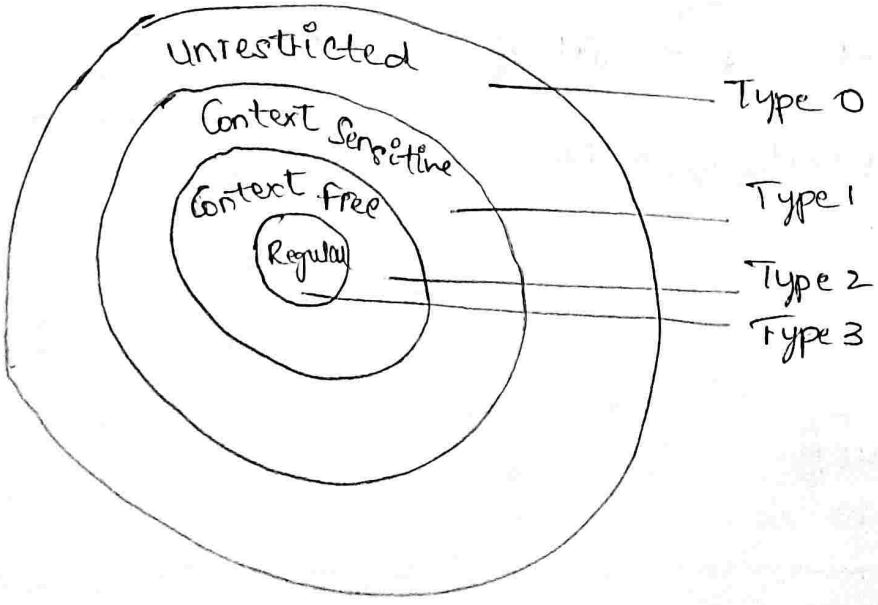
$$\delta : Q \times N \longrightarrow Q \times N \times \{L, R\}$$



---

(2) Classify briefly about Chomsky hierarchy of languages.

A) Write down Part A Q(3) and Q(9), and then →



(3) Describe a Turing Machine. With a neat diagram explain the working of a Turing Machine.

*) • A Turing Machine is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 Grammar.

• It was invented by Alan Turing in 1936.

• A Turing Machine is a mathematical model which consists of an infinite length tape divided into cells on which input is given.

• It consists of a head which reads the input tape. A state register stores the state of the Turing Machine.

• It is represented by a 7-tuple notation,

$$(Q, X, \Sigma, \delta, q_0, B, F)$$

where,

Q — Finite set of states
X — Tape Alphabet
Σ — Input Alphabet
δ — transition function
q_0 — initial state
B — blank symbol
F — set of final states

• Time and Space Complexity of a TM.

$$T(n) = O(n \log n)$$

$$S(n) = O(n)$$

• [Give a example here].

(4) Compare Turing Machine with other Automata.

A) Refer Part A – Q8.

---

(6) Express short notes on Recursive and Recursively Enumerable languages

(7) Describe the properties of recursive and recursively enumerable languages

A) Recursive Enumerable (RE) or Type-0 Language:

RE language or type 0 languages are generated by type 0 grammars.

An RE language can be accepted or recognized by TM, which means it will enter into final state for strings of language and may or may not enter into rejecting state for string which are not part of the language.

It means TM can loop forever for strings which are not part of language.

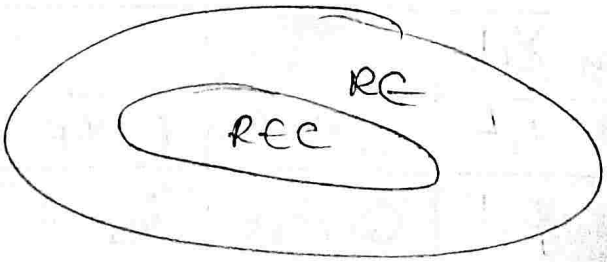
RE languages are also called Turing Recognizable languages

Recursive Language:

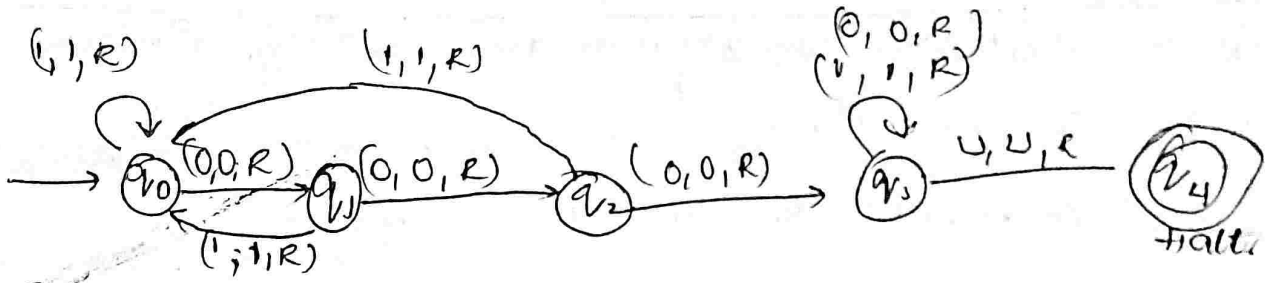
A recursive language (subset of RE) can be decided by TM which means it will enter into final state for the strings of language and rejecting state for the strings which are not part of the language.

REC languages are also called Turing decidable languages.

→ The relationship between RE and REC,

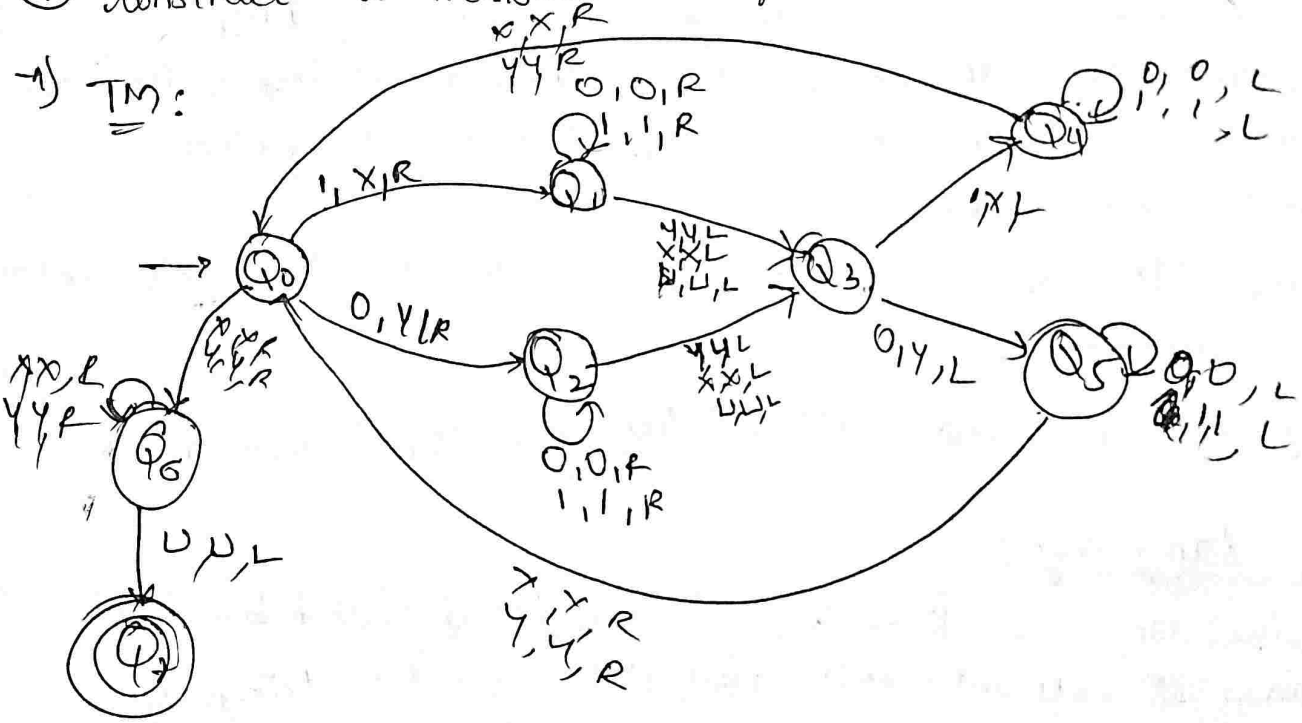


⑧ Develop TM to accept strings formed with 0s and 1s and having sub string 000



⑨ Construct a transition diagram to accept $L = \{ww^R \mid w \in (a+b)^*\}$

→) TM:



	0	1	X	Y	U
Q_0	Q_2, Y, R	Q_1, X, R	Q_6, X, R	Q_6, Y, R	
Q_1	$Q_8, 1, R$	$Q_1, 1, R$	Q_3, X, L	Q_3, Y, L	Q_3, U, L
Q_2	$Q_2, 0, R$	$Q_2, 1, R$	Q_3, X, L	Q_3, Y, L	Q_3, U, L
Q_3	Q_5, Y, L	Q_4, X, L			
Q_4	$Q_4, 0, L$	$Q_4, 1, L$	Q_0, X, R	Q_0, Y, R	
Q_5	$Q_5, 0, L$	$Q_5, 1, L$	Q_0, X, R	Q_0, Y, R	
Q_6			Q_7, X, R	Q_7, Y, R	
Q_7					U, U, L

(10) Design transition table for TM $L = \{a^n b^n c^n / n \geq 1\}$

→) Copy PART - (A) → (Q9), then.

	a	b	c	X	Y	Z	⊔
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_1 b L		q_0 X R	q_3 Y L	q_3 Z L	
					q_4 Y R	q_4 Z L	q_5 ⊔ L
q_4							
q_5	–	–	–	–	–	–	–

(11) Construct a transition table for TM $L = \{0^n 1^n 0^n / n \geq 1\}$

→) Just Replace a = 0, b = 1, c = 0, in above question.

(12) Construct TM that accept $L = \{1^n 2^n 3^n / n \geq 1\}$. Give transition diagram and also show moves made by TM for string, 111222333

→) Replace a = 1, b = 2, c = 3 in (Q10), Then.

→ Given string 111 222 333 ⊔⊔⊔ ... [in tape]

- Head points to 1, then updates it to X.
- Moves right until 2 is hit.
- The 2 is updated with Y, and moves right.
- Moves right until 3 is hit.
- 3 is updated with Z.
- Head moves left until X is hit, then moves on step to right
- If it is (1) above process is repeated. [Repeated 3 times]
- If it is Y, the head moves to right until '⊔' is hit.
- Halt.

(13) Enumerate Linear Bound Automata and explain it model.

A) Refer Part-B (Q1) ;

(14) Demonstrate the power and limitations of TM.

A) Power of TM:

The Turing machine has great computational capabilities. So, it can be used as a general mathematical model for modern computers too.

Turing Machine can model even recursively enumerable languages

Thus the advantage of TM is that it can model all the computable functions as well as the languages for which the algorithm is possible

Limitations of TM:

- They do not model the strengths of a particular arrangement well.
- When TM are used as basis for bounding running times a false lower bound may be proven, because of no memory indexings.
- The solutions are / may not be optimal because of not implementing an indexed memory.
- Another limitation of TM, is that they do not model concurrency well.
- There are always - halting concurrent systems with no inputs that can compute large integers, unlike the TM.

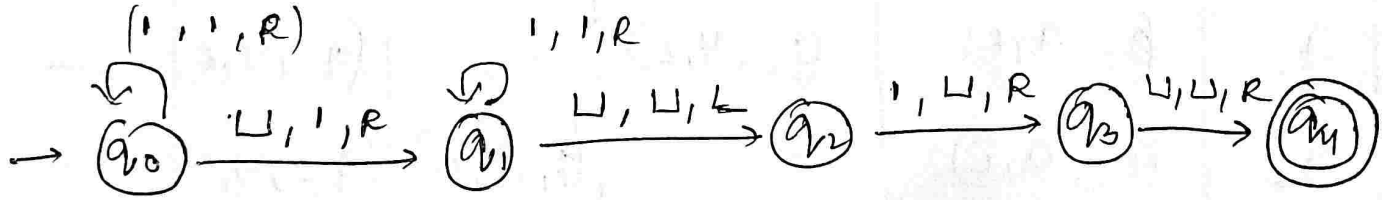
(15) Construct TM diagram for $L = \{a^n b^n c^n \mid n \geq 1\}$

A) Refer (Q10);

(16) Construct a Transition diagram to implement unary addition. (X + Y)

A) Procedure:

- The input tape is of format → X ␣ Y ␣ ␣ ␣ ...
- The head start from the left-most end;
- It moves right skipping all 1's, until a '␣' is hit.
- The '␣' is updated to '1', and the head moves right
- The head moves right until '␣' is hit.
- The head moves one step left and hits '1'.
- This '1' is updated to '␣',
- The head move right to hit '␣',
- Halt.

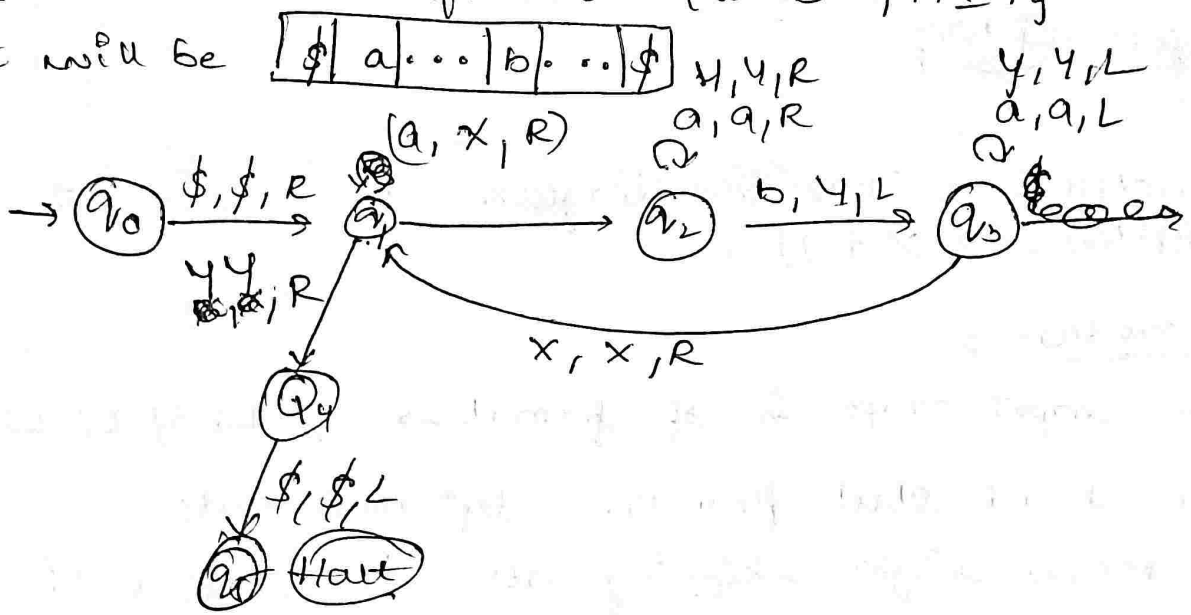


(*) Transition diagram:

	1	␣
q_0	$(q_0, 1, R)$	$(q_1, 1, R)$
q_1	$(q_1, 1, R)$	$(q_2, ␣, L)$
q_2	$(q_3, ␣, R)$	–
q_3	–	$(q_4, ␣, R)$
q_4	–	–

(14) Construct LBA for $L = \{a^n b^n \mid n \geq 1\}$

- Input will be | $ | a | ... | b | ... | $ |

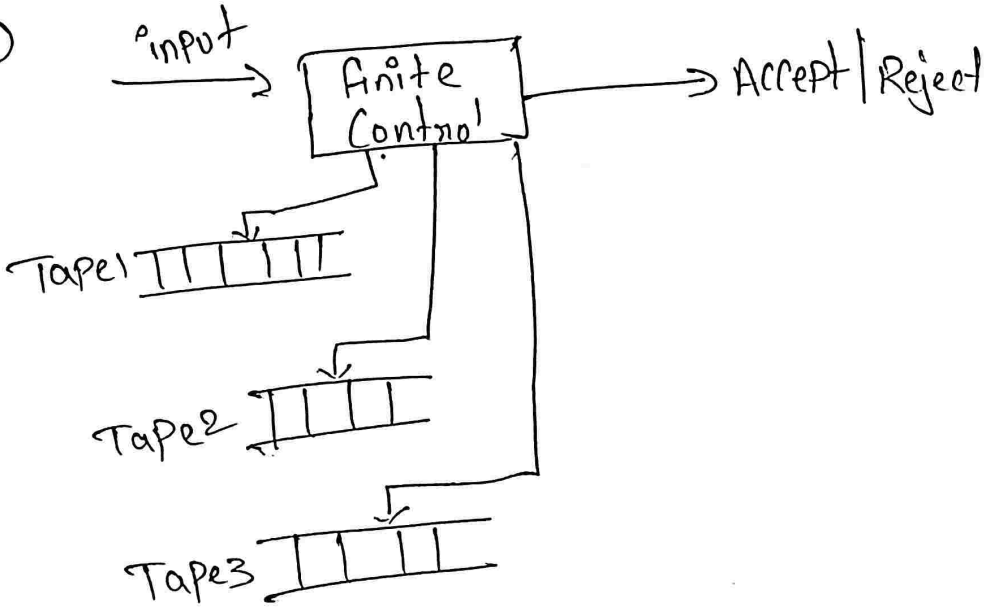


Transition Diagram:(Table):

	a	b	X	Y	$
q_0	—	—	—	—	$(q_1, \$, R)$
q_1	(q_2, X, R)	—	—	(q_4, Y, R)	—
q_2	(q_2, a, R)	(q_3, Y, L)	—	(q_2, Y, R)	—
q_3	(q_3, a, L)	—	(q_1, X, R)	(q_3, Y, L)	—
q_4	—	—	—	—	$(q_5, \$, L)$
q_5	—	—	—	—	—

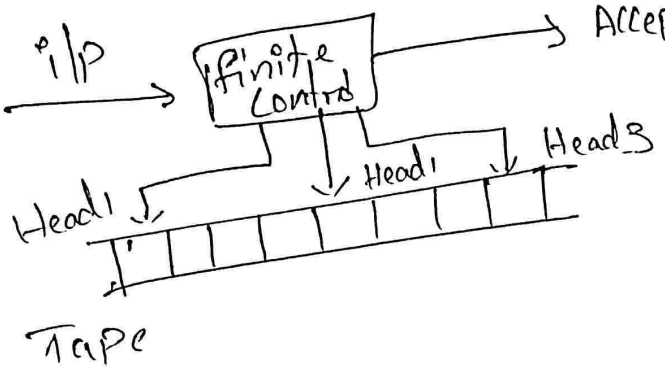
Types of turing machine:

1) Turing machine with multiple tapes
2) Turing machine with one tape mutiple Heads
3) Turing machine with Two Dimensional tape
4) Turing machine with infinite tape.
5) Non - deterministic turing machine

① 

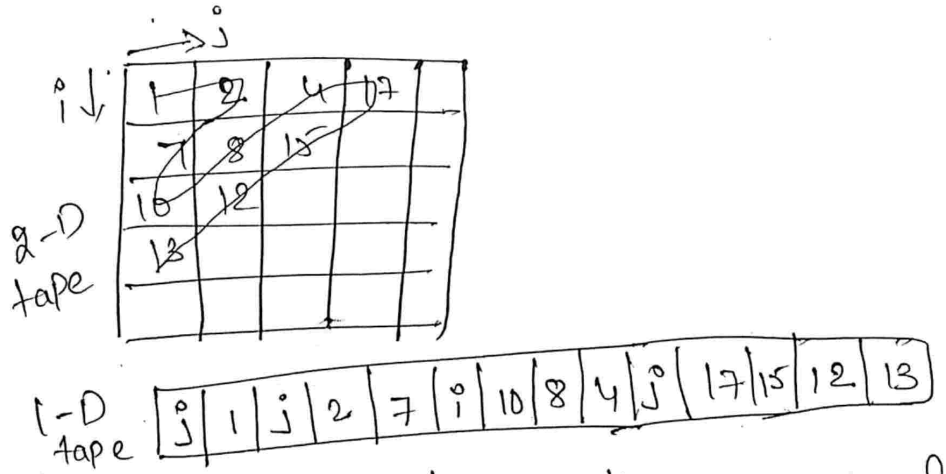
It has finite control, more than one tapes having its own read write head.

The language accepted by n-tape turing machine can be accepted by 1-tape turing machine

② 

There are n-heads but in state only one head can move. This type of turing machine are powerful as one tape turing machine

(3)

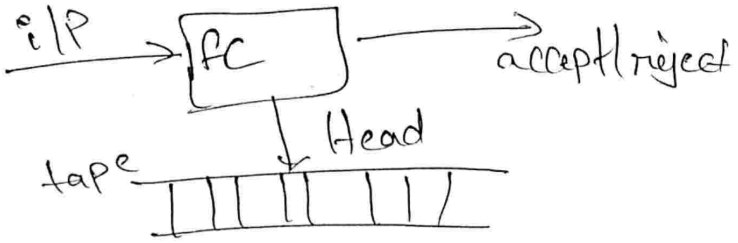


It is divided into small squares formed dive to corresponding rows & collems

→ Turing machine with 1-D tape is equally powerful to that of 2-D tape

→ The head of 2-D tape moves one square updown left right

(4)



Turing machine that have one finite control & one tape which extends infinetly both directions. This type of turing machine is powerful as one-tape TM whose tape has left end.

(6) Similar to NFA, for any state & any i/p symbol. It can take any action from a set rather than a definite Pre-determined action even in some situations it may take different actions at different times.

Eg: Turing machine which accepts language L

$L = \{ww \mid w \in (a+b)^*\}$ is non-deterministic TM

19. a)

Church's Hypothesis:

The assumption that the intuitive notion of computable functions can be identified with partial recursive functions

→ However this hypothesis cannot be proved

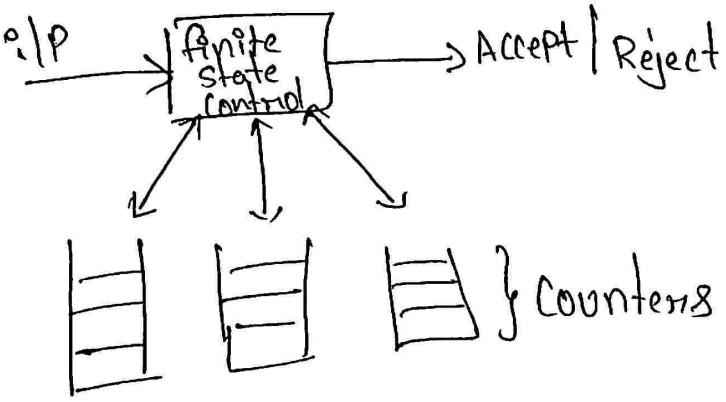
→ Computability of recursive functions is based on the following assumption

i) Each elementary function is computable.

ii) let f be computable function & A be another function which can be obtained by applying an elementary operations to F, then A becomes computable function.

iii) Any function becomes computable if it is obtained by rule i & ii

b) Counter Machine:



Counter Machine

It is similar to multi stack turing machine but the only the difference between this is in place of each stack, there is a counter, which contains non -ve integers each move of counter machine depends on its state, current i/p symbol.

In one move of counter machine, CM can

a) Change state
b) add or subtract one from any one of its counters.

-ve counters are not allowed at all

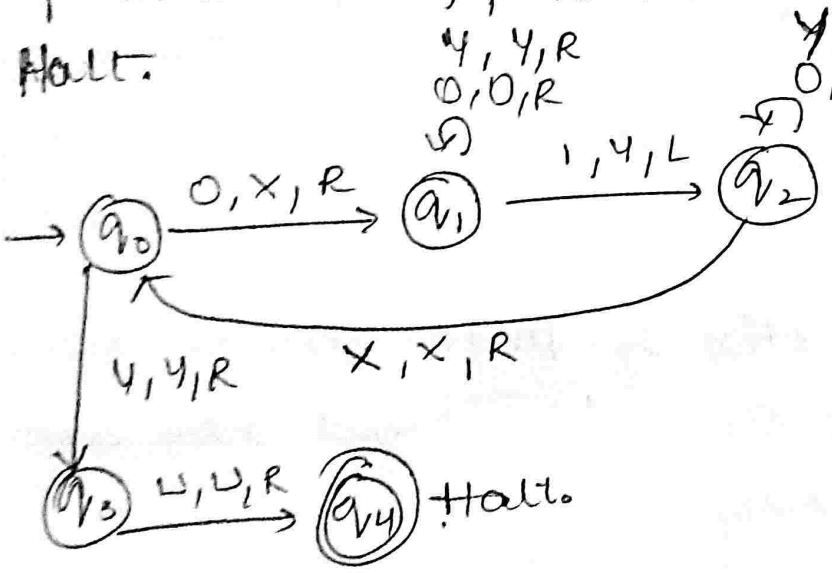
→ Counter Machine is similar to restricted multi stack Machine which following restrictions.

i) There are only two stack symbols Z_0, X
ii) Z_0 is bottom of stack marker. It is initially on each stack.
iii) Replace Z_0 only by string of the form $X^i Z_0$; $i \geq 0$
iv) Replace X only by X^i, $i \geq 0$ i.e Z_0, appears on the bottom of each stack & all other stack symbols are X
v) Every language accepted by CM is recursively enumerable.

(20) Construct TM for $L = \{0^n 1^n \mid n \geq 1\}$, Give the Transition diagram moves for string 000111

1) Procedure:

- Head starts on left most end.
- The first '0' is updated with X;
- Head moves right until '1' is hit.
- The '1' is updated to 'Y';
- Head move left until 'X' is hit, head move one step Right
- If head hits '0', repeat from step (2);
- If head hits 'Y', head moves to right until '␣' hit.
- Halt.



Transition Table:

	0	1	X	Y	␣
q_0	(q_1, X, R)	—	—	(q_3, Y, R)	—
q_1	$(q_1, 0, R)$	(q_2, Y, L)	—	(q_1, Y, R)	—
q_2	$(q_2, 0, L)$	—	(q_0, X, R)	(q_2, Y, L)	
q_3	—	—	—	—	$(q_4, ␣, R)$
q_4	—	—	—	—	—

(4) Given string, 000 | | | UUU ...

- '0' is update to X,
- Head moves right until '|' is hit.
- '|' is update to 'Y'.
- Head moves left, until 'X' is hit.
- Head move one step right, onto '0'
- Repeat above steps; until all '0' are 'X', all '|' are 'Y'.
- Head move to right until 'U' hit.
- Halt.

- PART - C :

All answers to question in part c can be answered via part B. The ones which cannot have been answered down here.

(9) Describe moves in TM

A) At each step of its operation, the head reads the symbol in its cell.
Then, based on the symbol and the machine's current state, the machine writes a symbol into the same cell, or updates it with some other symbol.
Then, the head moves one step to the left or right (or halts) computation.

(10) Define an Instantaneous description of a TM.

A) "All symbols to left of head, State of Machine, symbol head is scanning and all symbols to right of the head"

— These define an Instantaneous description of a TM.

(21) Describe TM for 1's Complement for binary numbers.

A) Procedure:

- Scanning from left to right.
- All '0' are converted to '1'.
- All '1' are converted to '0'.
- Halt when blanks reached.

	0	1	⊔
q_0	$(q_0, 1, R)$	$(q_0, 0, R)$	$(q_1, ⊔, R)$
q_1	—	—	—

→ Unsolved Questions:-

i) PART A: (4)

ii) Part B: (5)