INDEX

NAME: R. RAJESH KUMAR ROLLNO: SF2

STD: DIV./SEC: C SUBJECT: AFL

S. No.	Date	Title	Page No.	Teacher's Sign/Remarks
TOC				

String can start with ϵ [null]
Word cannot start with ϵ

Alphabets (Σ)	Language (L)
Ex:- $\Sigma = \{a, b\}$	Ex:- $L = \{ab, b\}$
$\Sigma^0 = \{\epsilon\}$	$L^0 = \{\epsilon\}$
$\Sigma^1 = \{a, b\}$	$L^1 = \{ab, b\}$
$\Sigma^2 = \{aa, ab, ba, bb\}$	$L^2 = \{abab, bab, bab, bb\}$
$\Sigma^* = \{\epsilon, a, b, \dots,\}$	$L^* = \{\epsilon, ab, b, \dots\}$
$\Sigma^+ = \{a, b, aa, \dots\}$	$L^+ = \{ab, b, \dots\}$

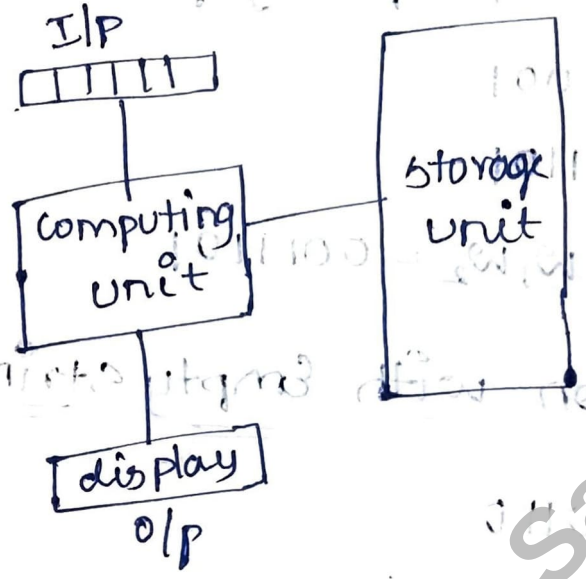
Automata Formal Language

Introduction to Formal langua

Ullman

Automata :-

→ It is an Abstract method of digital computer



Central Concepts of AFL :-

- **Symbol :-**

→ An abstract Entity which indicates a letter, a digit. Ex :- Machines : 0s, 1s

- **String :-**

→ It is a finite sequence of symbols.

Ex :- 001, 110, 010

→ **Length of String :-**

→ Total no of symbols in the string.

Ex :- w = 111100100

|w| = 9

→ Empty string :- (ε)

→ There is no symbols in a string.

⇒ Properties of string :-

1) Concatenation Property :

$w_1 = 001$

$w_2 = 110$

Concatenate $w_1 w_2 = 001,110$

2) Concatenation with Empty strings :-

$w = 0110$

concatenate $\varepsilon w = 0110$

concatenate $w\varepsilon = 0110$

$\therefore \varepsilon w = w\varepsilon$

• Alphabets :-

→ An alphabet is a finite set of symbols

→ It is denoted with Σ.

Ex :- $\Sigma = \{0, 1\}$

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

⇒ power of Alphabet :-

- $\Sigma^n = 2^n$ where n is a Symbols

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

Σ^n = Language over Symbols $\{0, 1\}$ with length of n.

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

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

- Language :-['L']

→ A certain Specified Set of strings of Symbols from alphabets is called the Language.

Ex :- A Language over $\Sigma = \{a, b\}$

$L_1 = \{a, b\}$ $L_2 = \{aab, bba\}$

$L_3 = \{ab, ba, bba, baa\}$

⇒ Properties on Language :-

1) Concatenation : $L_1 L_2 \neq L_2 L_1$

Languages $L_1 L_2$ are $\Sigma = \{a, b\}$.

$L_1 = \{ab, bb\}$ $L_2 = \{a, b\}$

Concaterate $L_1 L_2 = \{aba, abb, bba, bbb\}$

$L_2 L_1 = \{aab, abb, bab, bbb\}$

→ Star closure operation :- (Klleen: closure)

Ex:- Languages $\Sigma = \{0, 1\}$

$L^0, L^1, L^2, \ldots$ where L^0 is a null / Empty

$L^* = \{L^0 \cup L^1 \cup L^2 \cup \ldots\}$

→ Positive closure operation (L^+):

$L^+ = \{L^1 \cup L^2 \cup L^3 \cup \ldots\}$

$L^+ = L^* - \{\epsilon\}$

→ Star closure including with null string.

→ positive closure: Excludes null string.

Example :-

→ Define a word or string for a set of

$S = \{0, 1\}$.

Ans:- $w_1 = 0011$

$w_2 = 1100110$

→ Consider the strings $x = 110$ $y = 0110$.

find i) x^2 ii) ϵy

Ans:- i) $x^2 = xx = 110110$

ii) $\epsilon y = 0110$

→ Describe in the words the following over $\Sigma = \{a, b\}$

i) $L_1 = \{a, ab, ab^2, \ldots\}$

Ans:- A language L_1 over $\Sigma = \{a, b\}$ consist of all words begin with a followed by any number of b's.

ii) Language L of $\Sigma = \{a, b\}$

$L = \{a^n b^n$ when $n \geq 0\}$ → star closure

Ans:- A language over $\Sigma = \{a, b\}$ Begins with a (or) 0 (or) more no of a's followed by (or) more number of b's

iii) $L = \{a^n b^n$ where $n \geq 1\}$ → positive closure

Ans:- A language over $\Sigma(a, b)$ begins with any number of a's follows any number of b's.

Differences Blw:-

STAR CLOSURE	positive closure
① $L^* = \{L^0 \cup L^1 \cup L^2 \cup \ldots\}$	① $L^+ = \{L^1 \cup L^2 \cup \ldots\}$
② Included null string	② Excluded null string

Ex:- $\Sigma = \{a, b\}$

$L^0 = \epsilon$

$L^1 = \{a, b\}$

$L^2 = \{ab, ba, aa, bb\}$

$L^3 = \{aab, aba, \ldots\}$

$L^* = \{\epsilon, a, b, ab, ba, \ldots\}$

$L^+ = \{a, b, ab, ba, \ldots\}$

⇒ Given $\Sigma = \{a, b\}$. Obtain Σ^*, Hence check given $L = \{a^n b^n \mid n \geq 0\}$ and

i) aabb

ii) aaaabbbb

iii) abb

Ans:-

$\Sigma^* = \{\epsilon, a, b, aabb, aaabbb, aaaabbbb, \ldots\}$

i) Belongs

ii) Belongs

iii) Not Belongs

⇒ Given $L = \{a^n b^n \text{ where } n \geq 0\}$

obtain (1) L^2

(2) L^R

Ans:- i) $L^0 = \{\in\}$

$L^1 = ab$

$L^2 = \{aabb\}$

ii) $L^R = \{b^n a^n \text{ where } n \geq 0\}$

⇒ Given $L = \{ab, aa, baa\}$ which of the followings strings are in L^*.

i) ab|aa|baa|ab|aa — ✓

ii) aa|aa|baa|aa ✓

iii) baa|aa|baa|aab ✗

iv) baa|aa|baa ✓

Ans:-

i) Belongs

ii) Belongs

iii) Not Belongs

iv) Belongs

⇒ Given $L = \{a^n b^{n+1} \text{ where } n \geq 0\}$ is it know that $L^+ = L^*$ for given language of L

Ans:- $L^0 = b$

$\therefore$ No ϵ in L^*

$\therefore \boxed{L^+ = L^*}$

⇒ Given $u = a^2 b a^3 b^2$

$v = bab^2$

Obtain a) uv

b) $|uv|$

c) u^2

d) $|u^2|$

e) v^2

f) $|v^2|$

Ans:- a) $uv = a^2 b a^3 b^2 b a b^2$

b) $|uv| = 12$

c) $u^2 = a^2 b a^3 b^2 a^2 b a^3 b^2$

d) $|u^2| = 16$

e) $v^2 = bab^2 bab^2$

f) $|v^2| = 8$

⇒ for a Language $L = \{ab, c\}$ over $\Sigma = \{a, b, c\}$

find i) L^0

ii) L^1

iii) L^2

iv) L^{-2}

Ans:- i) $L^0 = \{\in\}$

ii) $L^1 = \{ab, c\}$

iii) $L^2 = \{abab, abc, cab, cc\}$

iv) L^{-2} = not Applicable

⇒ $L_1 = \{a, ab, a^2\}$ & $L_2 = \{b^2, aba\}$ over $\Sigma = \{a, b\}$

find i) $L_1 L_2$

ii) $L_2 L_1$

Ans:-

i) $L_1 L_2 = \{ab^2, aaba, abb^2, ababa, a^2b^2, a^2aba\}$

ii) $L_2 L_1 = \{b^2a, b^2ab, b^2a^2, abaa, abaab, abaa^2\}$

Finite Automata (NFA & DFA) :-

Finite state Machine :-

→ A mathematical model of system with discrete input and output Theory of automata is a useful design tools for these systems.

Applications :-

1) For designing and checking the behaviour of digital circuits.

2) In compilers to break i/p string into logical units.

3) In Languages, Grammers, it is used to define them.

4) It is used for communication protocol.

5) To scan large bodies of text such as collection of webs

FA

Deterministic FA

→ If the internal state i/p & contents of storage are known, it is possible to predict future behaviour of auto-machine. It is known as DFA

Non-deterministic FA

→ Otherwise

Deterministic Finite Automata (DFA)

↓ defined 5 tuple

$$M = (Q, \Sigma, \delta, f_0, F)$$

Where, M = Definite state machine

Q = finite non-Empty internal States

Σ = finite non-Empty alphabets/ symbols

δ = transition function

$$\delta : Q \times \Sigma^* \longrightarrow Q$$

f_0 = starting state $f_0 \in Q$

F = Final state $F \subseteq Q$

Represent the solution :-

(i) Transition Diagram :-

— Directed Graph

— vertices ⇒ states

Internal state = 0

Starting state = → 0

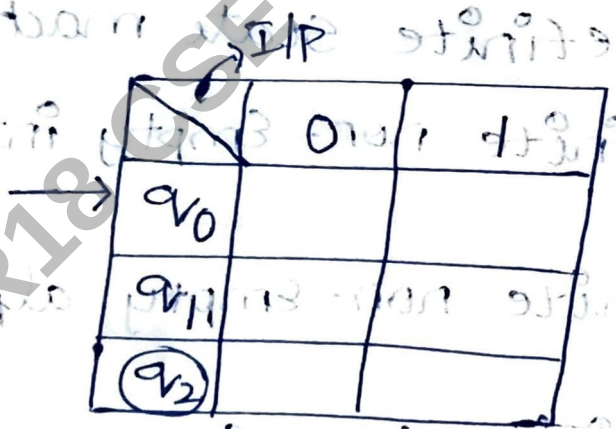
Final State = (0)

— Edge ⇒ Transition function

(ii) Transition Table :-

⇒ rows — states

⇒ columns — inputs



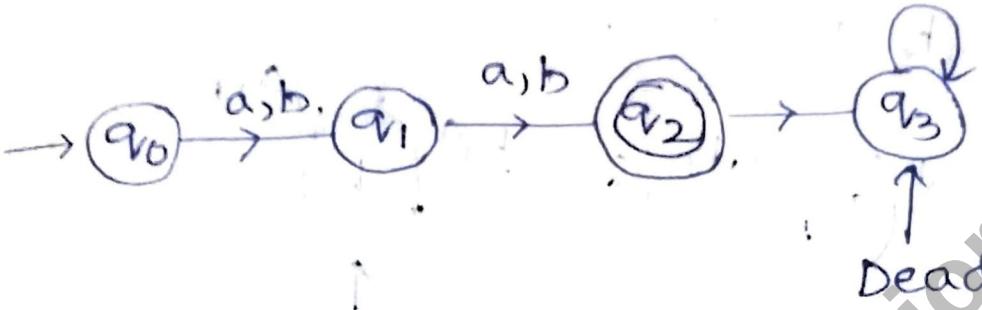
Problems:-

⇒ Construct DFA with $\Sigma = \{a, b\}$

$L = \{$ any a, b with length of string $w = 2\}$

Ans:-

$L = \{aa, ab, ba, bb\}$ $\quad |w| = 2$



For $|w| = n$

States required $= n + 2$

∴ Given $|w| = 2$

States required $= 2 + 2 = 4$

i/p / State	a	b
→ q_0	q_1	q_1
q_1	q_2	q_2
((q_2))	q_3	q_3
q_3	q_3	q_3

$\delta = Q \times \Sigma^* \longrightarrow Q$

$\delta_1 : q_0 \times a \longrightarrow q_1$

$\delta_2 : q_0 \times b \longrightarrow q_1$

$\delta_3 : q_1 \times a \rightarrow q_2$

$\delta_4 : q_1 \times b \rightarrow q_2$

$\delta_5 : q_2 \times a \rightarrow q_3$

$\delta_6 : q_2 \times b \rightarrow q_3$

$\delta_7 : q_3 \times a \rightarrow q$

$\delta_8 : q_3 \times b \rightarrow q$

⇒ Construct a DFA to accept length of string

i) $|w| \leq 2$

ii) $|w| = 2$ ✓

iii) $|w| \geq 2$

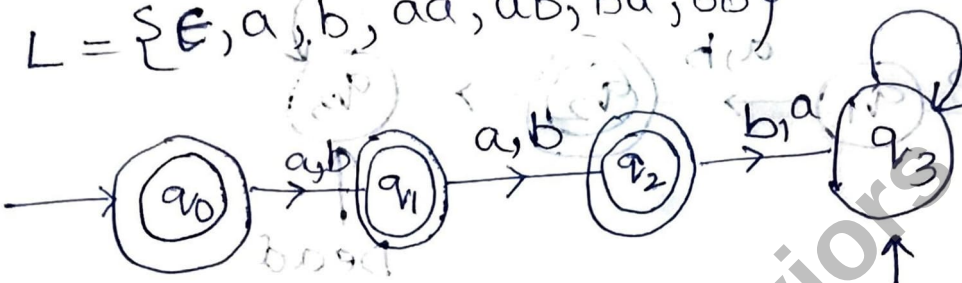
$L = \{aa, ab, ba, bb\}$ } TYPE-2

$|w| = 2$

Ans:-

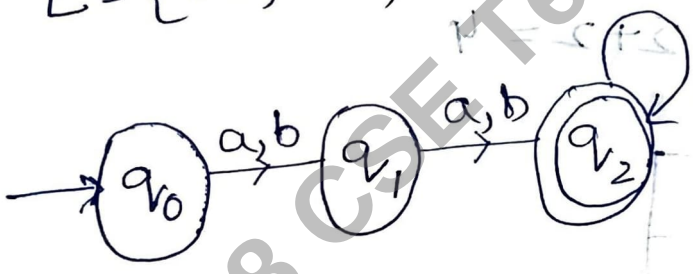
i) $|w| \leq 2$

$L = \{\epsilon, a, b, aa, ab, ba, bb\}$



iii) $|w| \geq 2$

$L = \{aa, ab, ba, bb, aaa, aab, bba, \ldots\}$



⇒ Construct DFA accept a string starting with over $w = \{a, b\}$

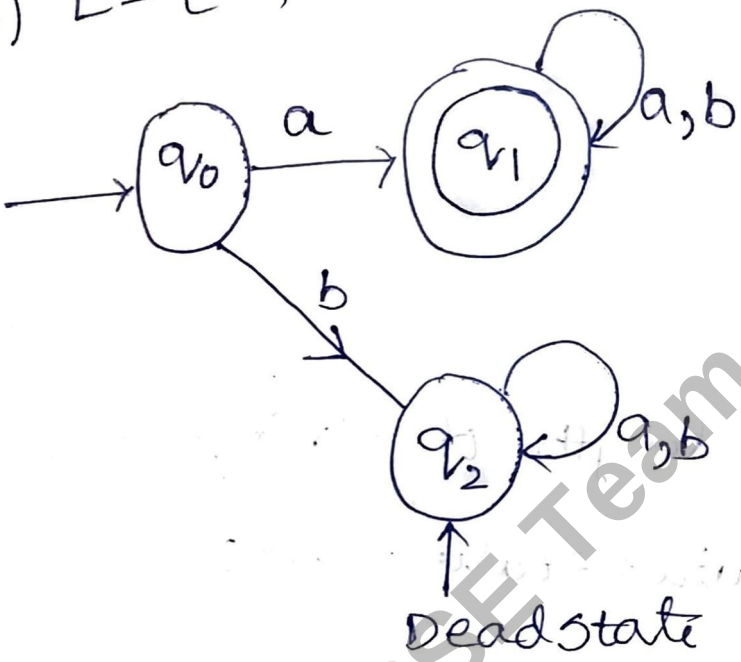
i) $w = a$

ii) $w = ba$

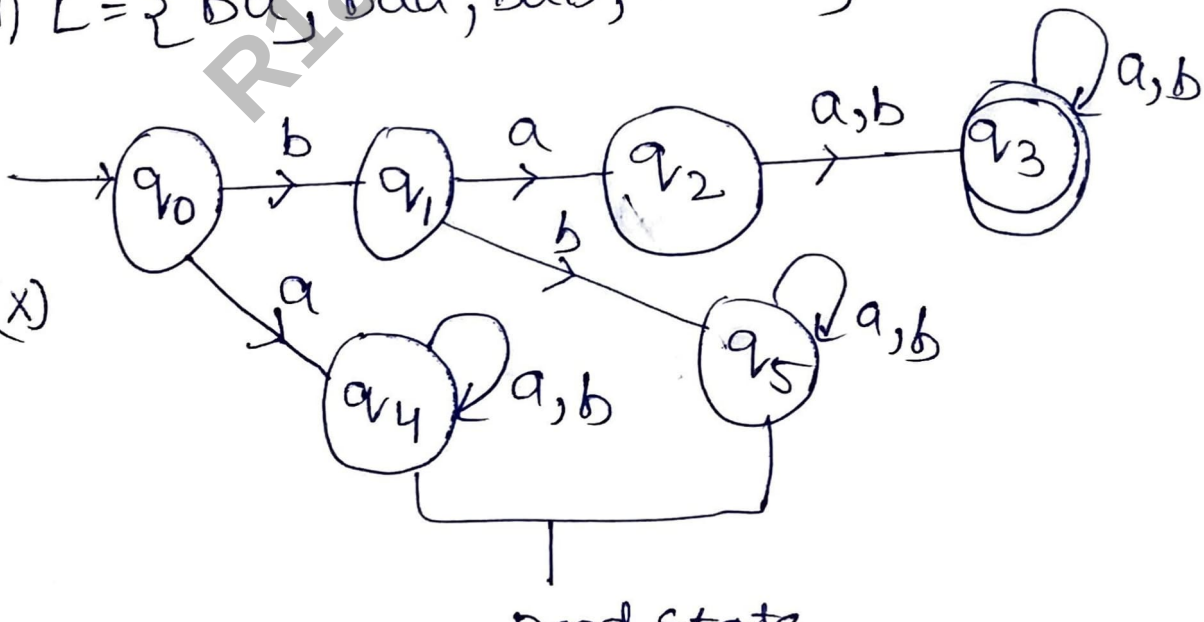
iii) $w = abb$

Ansr

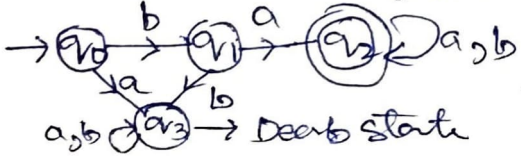
i) $L = \{a, ab, ab, aaa, aab, aba, abb, \ldots\}$



ii) $L = \{ba, baa, bab, \ldots\}$

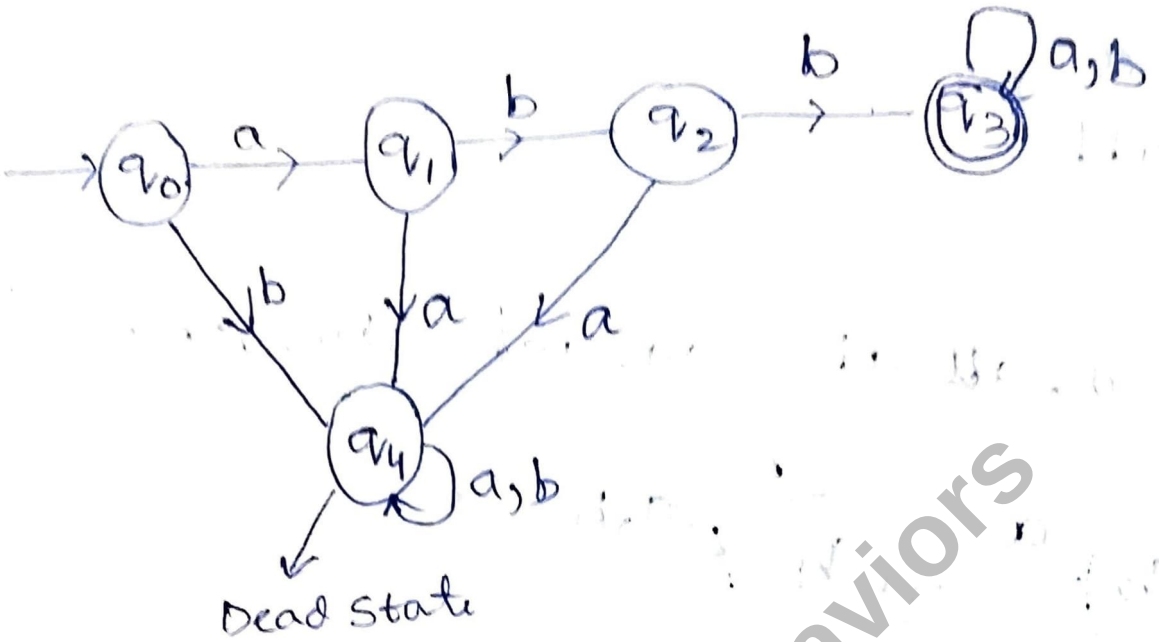


(v)



iii) $w = abb$

$L = \{abb, abba, abbb, \ldots\}$



NOTE :-

→ For starting n lengths of strings then

DFA required states = $n+2$

Type-3

⇒ Design DFA over $\Sigma = \{a, b\}$ ending with

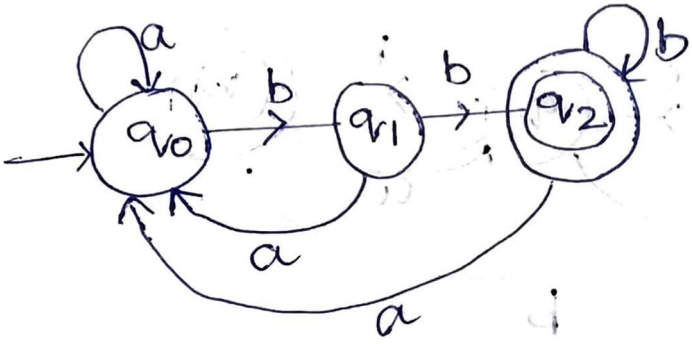
i) bb

ii) ab

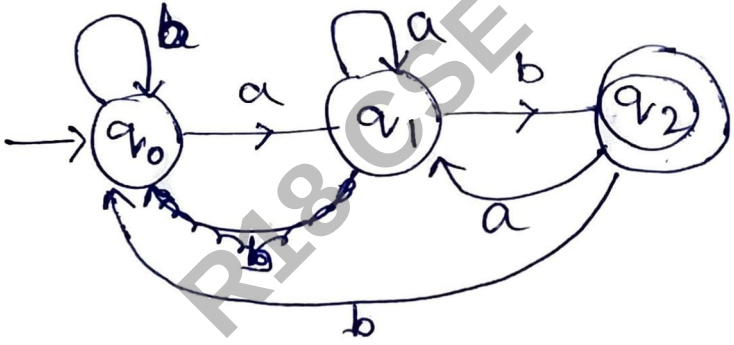
iii) bab

Ans:-

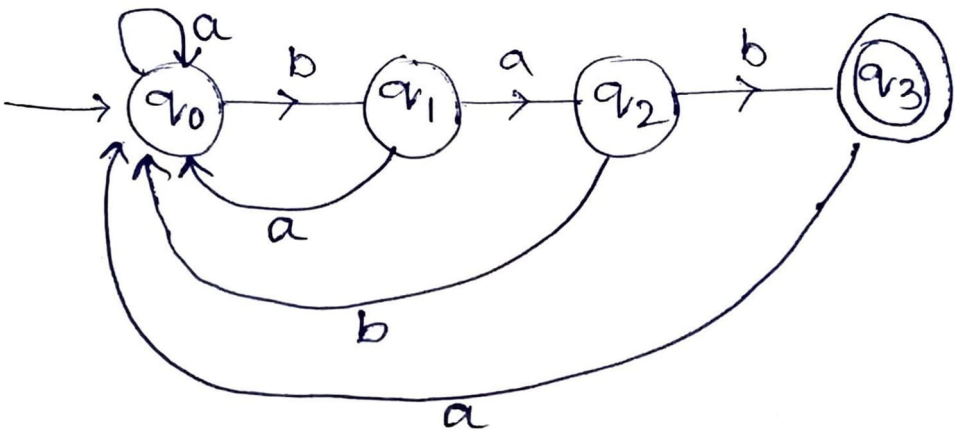
i) $L = \{bb, abb, bbb, aabb, abbbb, babb, bbbb, \ldots\}$



ii) $L = \{ab, aab, bab, \ldots\}$



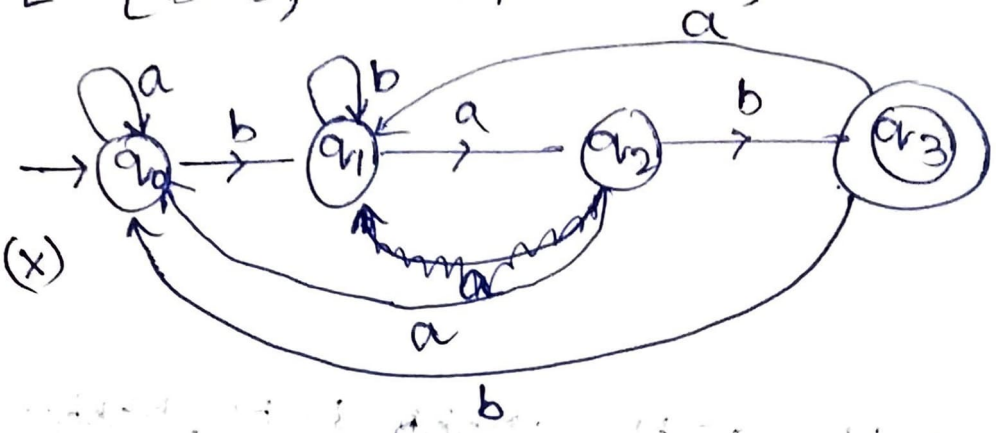
iii) $L = \{bab, abab, bbab, \ldots\}$



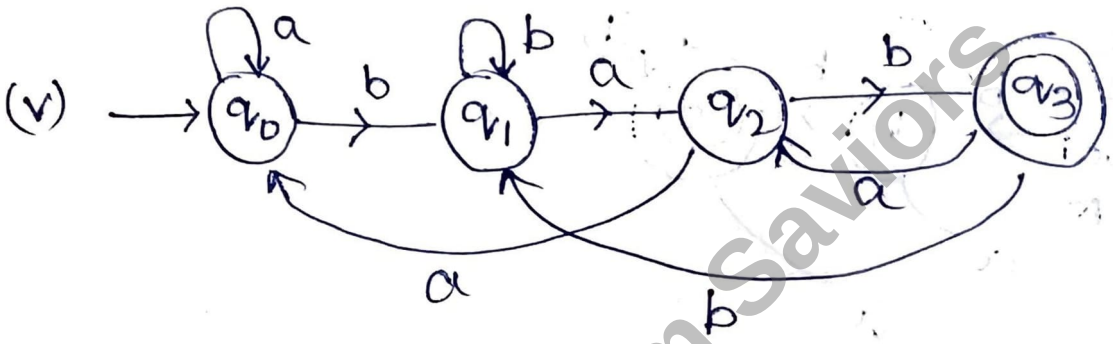
iii) bab

$L = \{bab, abab, \ldots\}$

(x)



(v)



NOTE:-

⇒ Ending with n length requires n+1 states

DFA

TYPE - 4

→ construct DFA over $\Sigma = \{a, b\}$ containing

i) aa

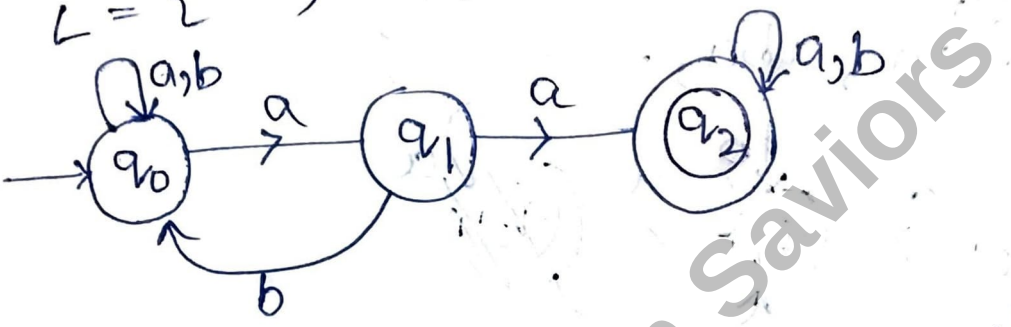
ii) ba

iii) abb

Ans:-

i) $L = \{$ any a's & b's + string + any a's & b's $\}$

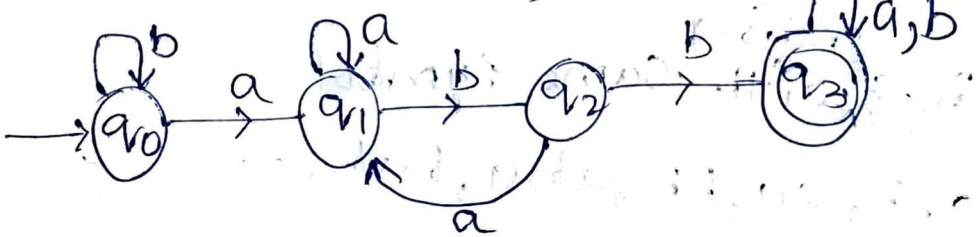
$L = \{aa, baa, aaa, aab, \ldots\ldots\}$



ii) $L = \{$ any a's & b's + string + any a's & b's $\}$

$L = \{ba, \ldots\ldots\}$

iii) $L = \{abb, \ldots\ldots\}$



NOTE:-

→ 'n' length strings requires n+1 states in DFA containing string

TYPE-5 :-

⇒ Construct DFA for a string starting & Ending with.

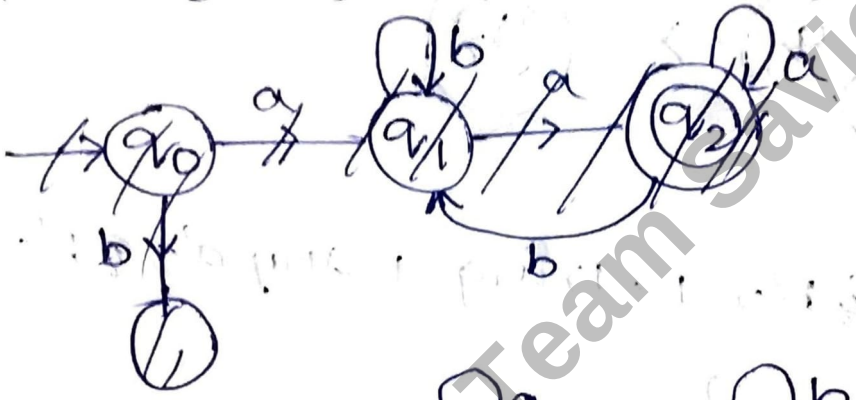
i) a

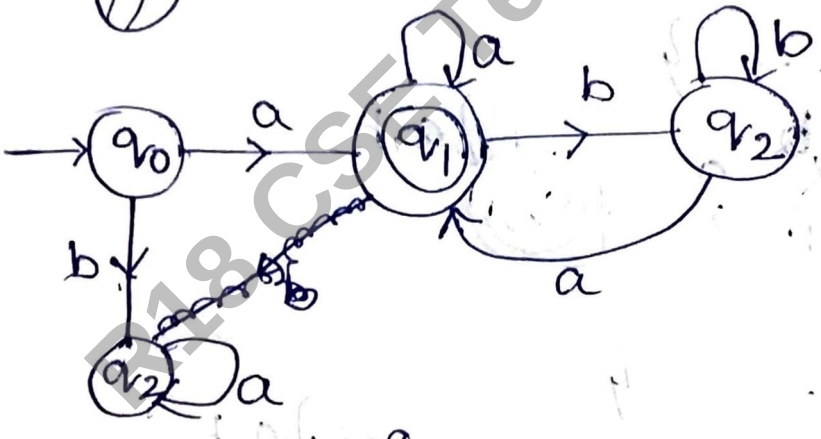
ii) sample symbol

iii) different symbol.

Ans:

i) $L = \{a, aa, aba, \ldots\}$

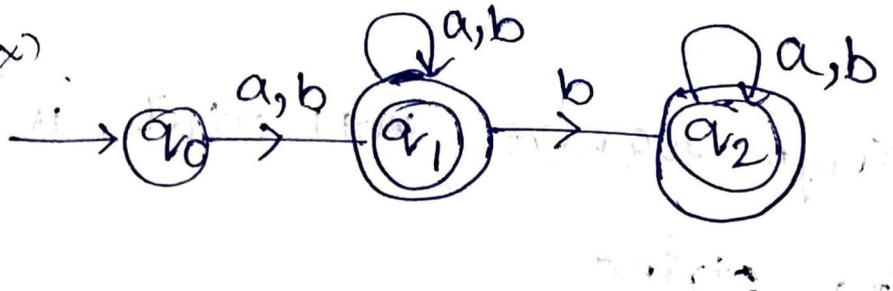




Ending:

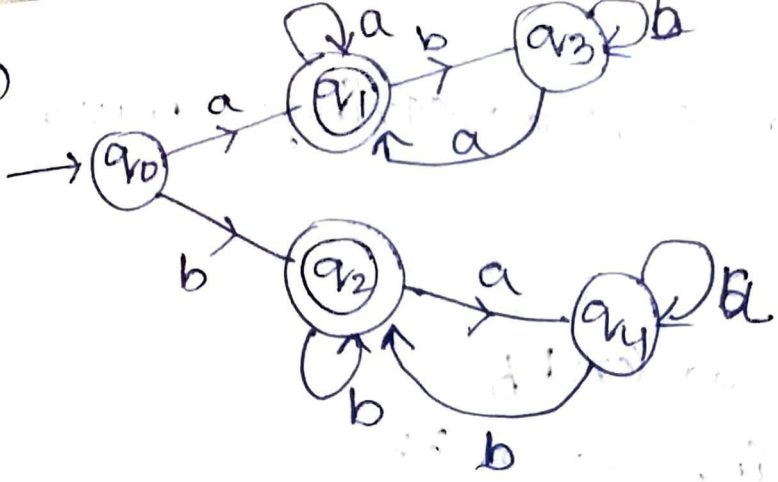
ii) starting & with same symbol

$L = \{a, b, aa, bb, aba, bab, \ldots\}$



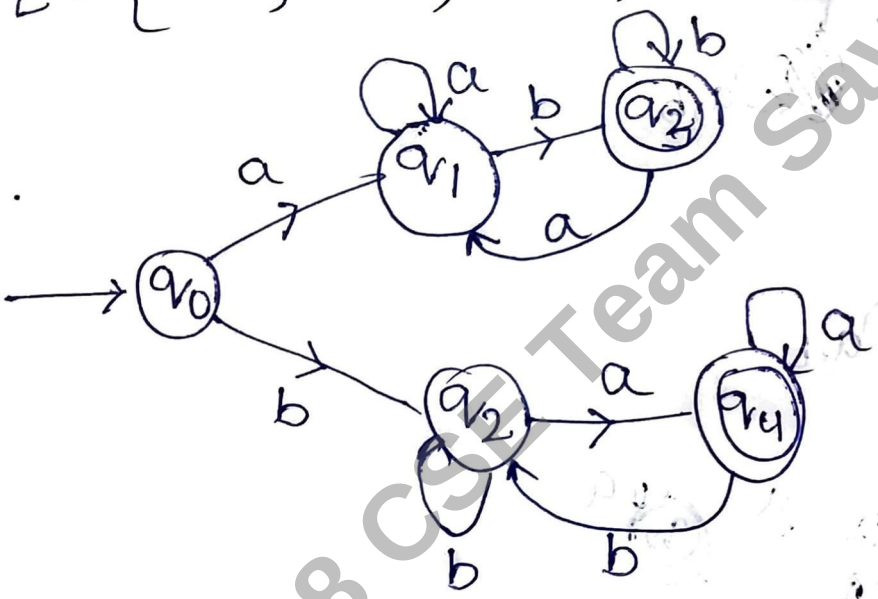
(x)

(v)



iii) Starting & Ending with different symbols

$L = \{ab, abb, aab, baa, bba, \ldots\}$



TYPE - 6 :-

⇒ Construct DFA to accept all strings must start with

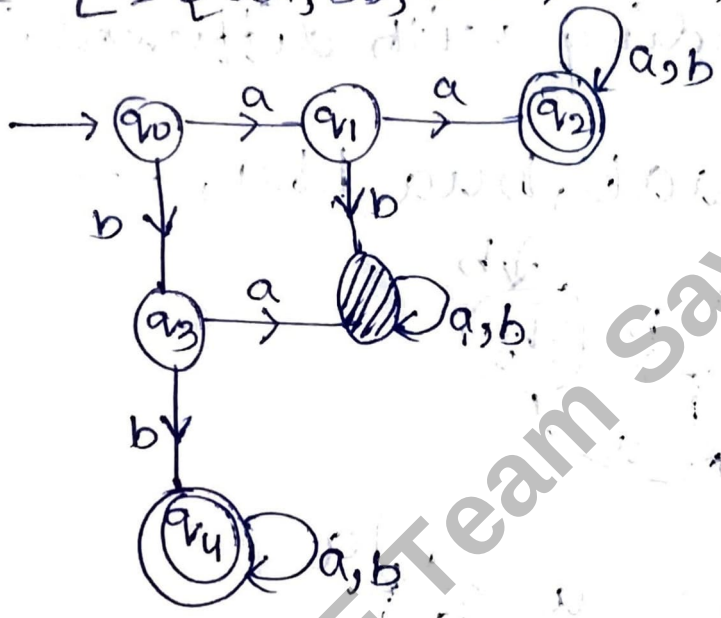
i) aa (or) bb

ii) start with aa (or) bb

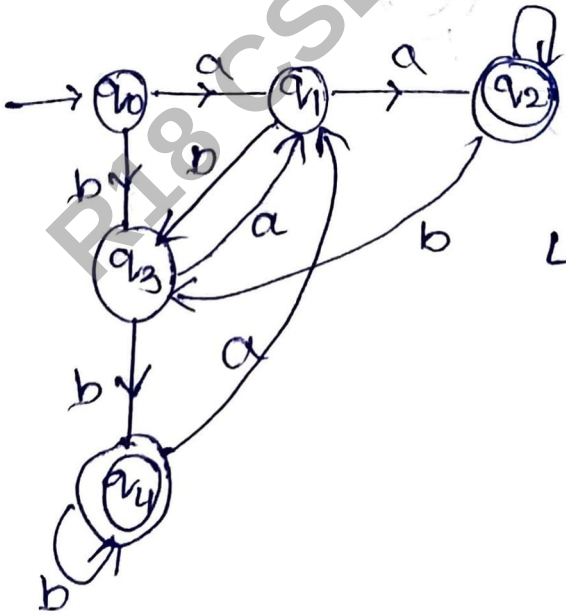
iii) must end with aa (or) bb

Ans: $L = \{aa, bb, aab, aaaa, \dots\}$

i)

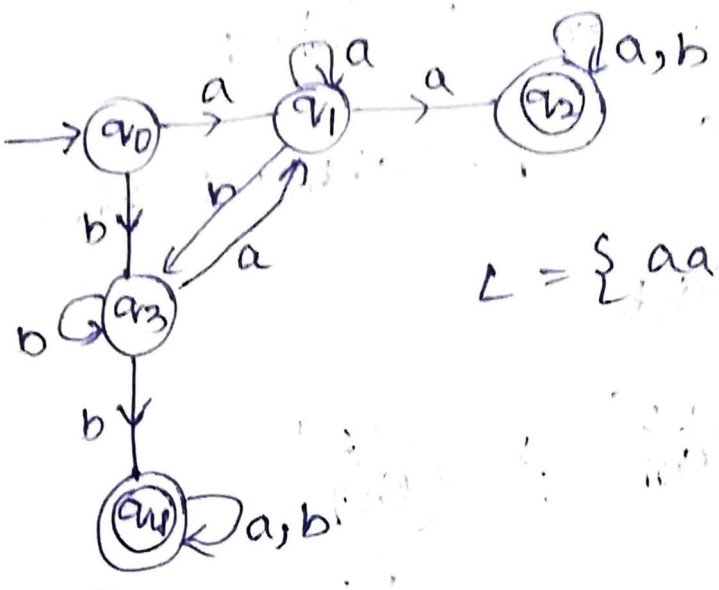


ii)



$L = \{aa, bb, aaa, bbb, \dots\}$

iii)



$L = \{aa, bb, baab, \ldots\}$

TYPE – 7

⇒ Design DFA which accepts

i) Every a followed by b

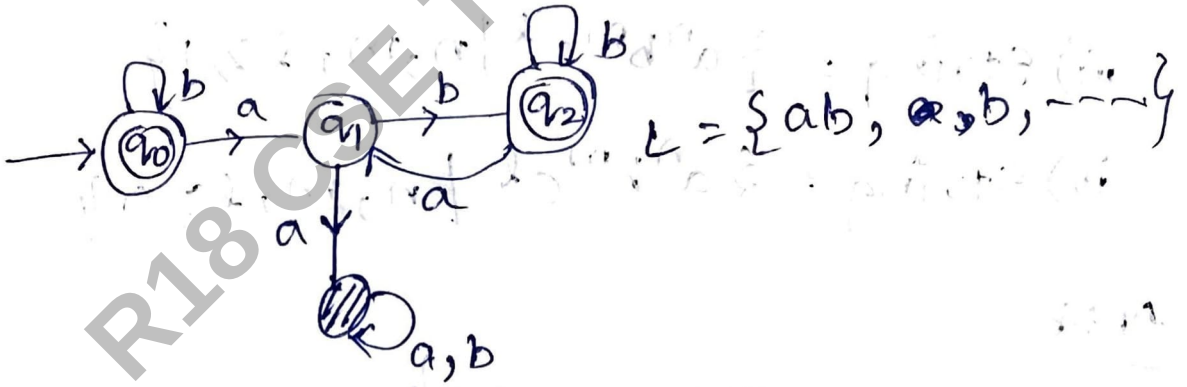
ii) Every a followed by bb

iii) Every a never followed by b

iv) Every a never followed by bb.

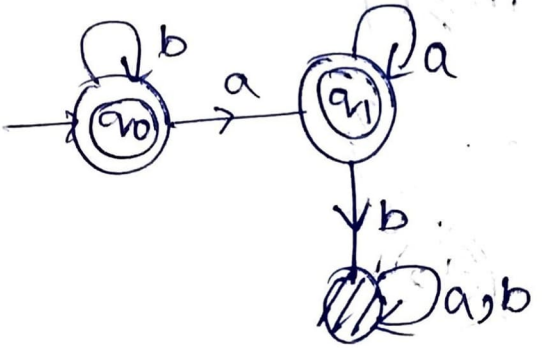
Ansb-

i)



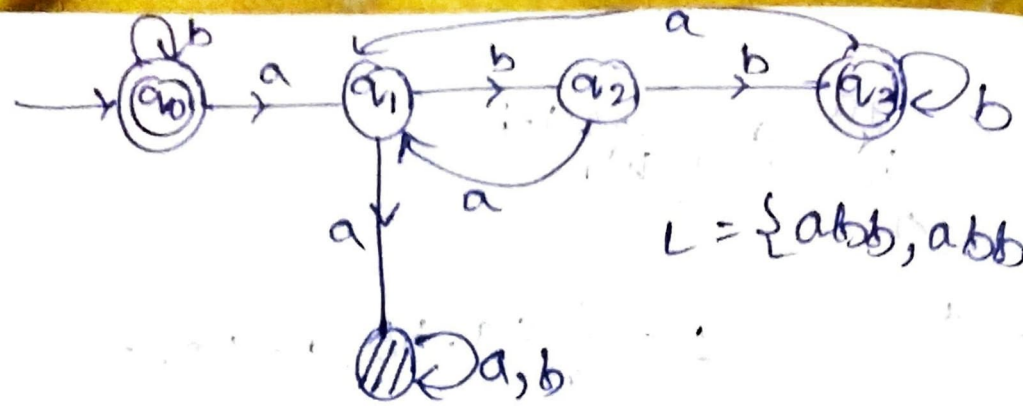
$L = \{ab, a, b, \ldots\}$

iii)



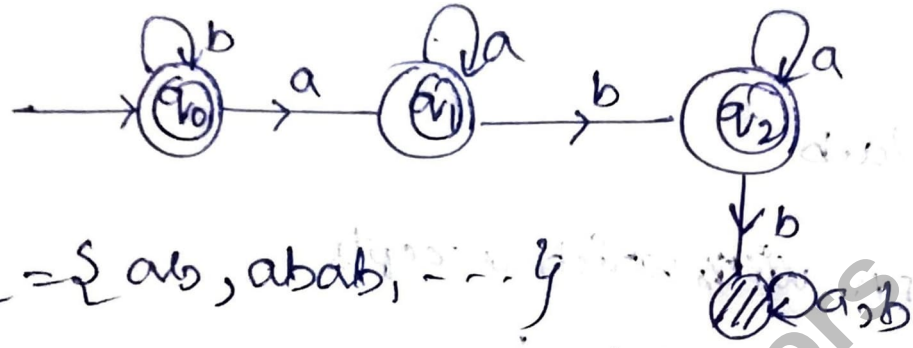
$L = \{aa, aaa, bbb, \ldots\}$

ii)



$L = \{abb, abbabb, \ldots\}$

iv)



$L = \{ab, abab, \ldots\}$

Type-8 :-

⇒ Design DFA to accept a i) string $= \{a^n b^m / n, m \geq 1\}$

ii) string $= \{a^n b^m / n, m \geq 0\}$

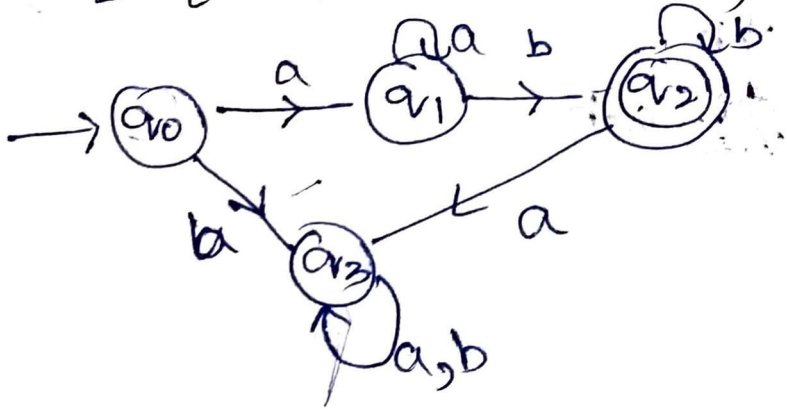
iii) String $= \{a^n b^m c^l / n, m, l \geq 1\}$

iv) String $= \{a^n b^m c^l / n, m, l \geq 0\}$

Ans:-

i) $L = \{a^n b^m / n, m \geq 1\}$

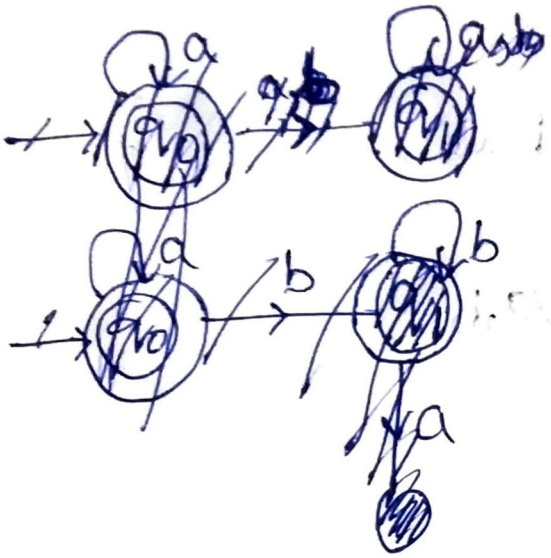
$L = \{ab, aabb, \ldots\}$



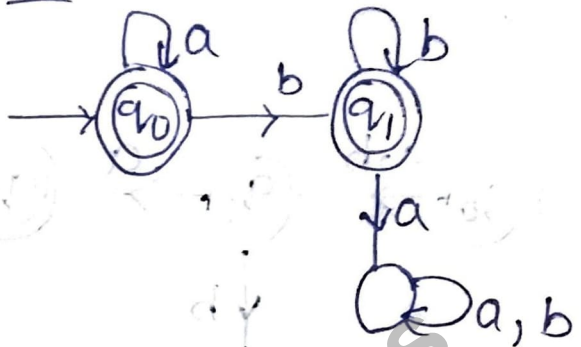
ii) $L = \{ a^n b^m \mid n, m \geq 0 \}$

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

$L = \{ \epsilon, ab, aabb, \ldots \}$

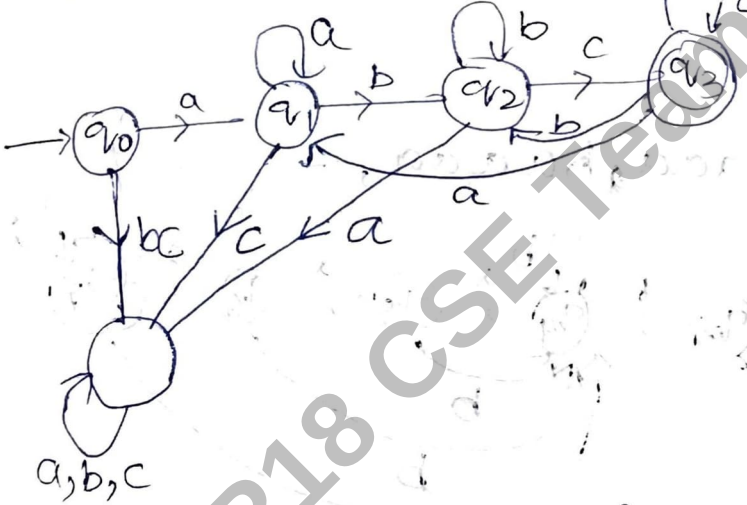


Ans :-

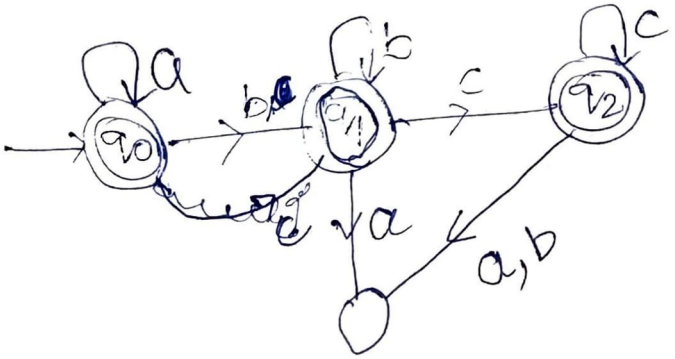


iii)

$L = \{ abc, \ldots \}$



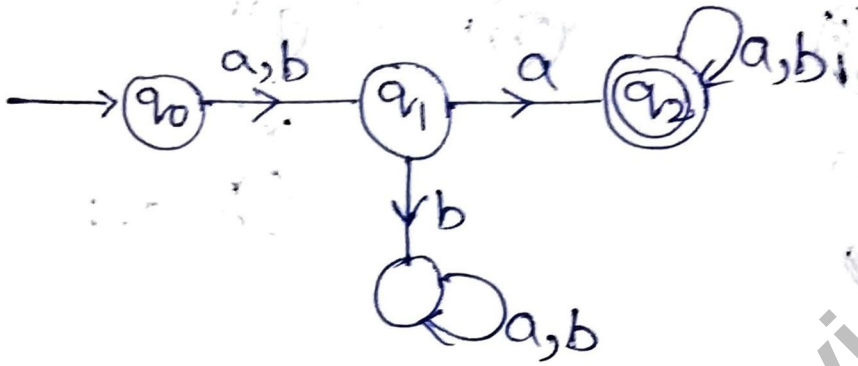
iv) $L = \{ \epsilon, abc, \ldots \}$



⇒ find DFA which accepts strings such that 2nd symbol from LHS is a

Ans:

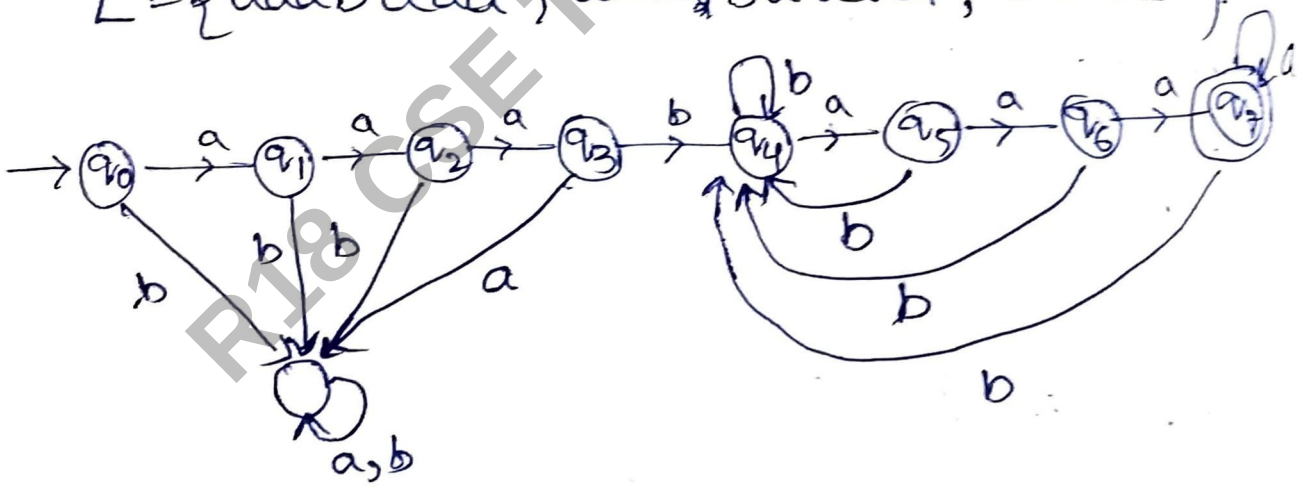
$L = \{aa, ba, aaa, aab, bab, \ldots\}$



⇒ Design DFA for $L = \{a^3 b w a^3 / w \in \{ab\}^*\}$

Ans:

$L = \{aaabaaa, aaaabaaaa, \ldots\}$



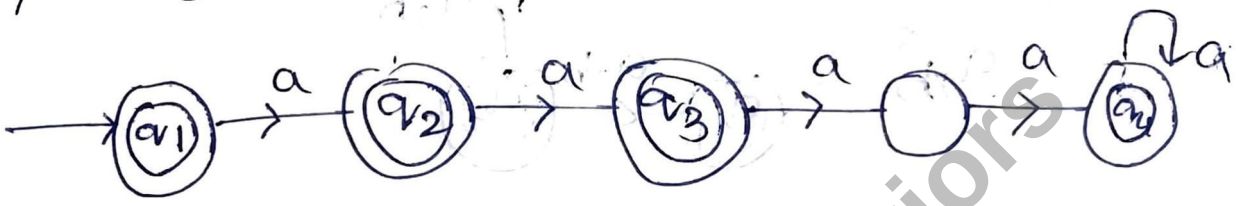
⇒ Construct Minimum DFA over $\Sigma = \{a\}$ to accept

i) $L = \{a^n / n \geq 0, n \neq 3\}$

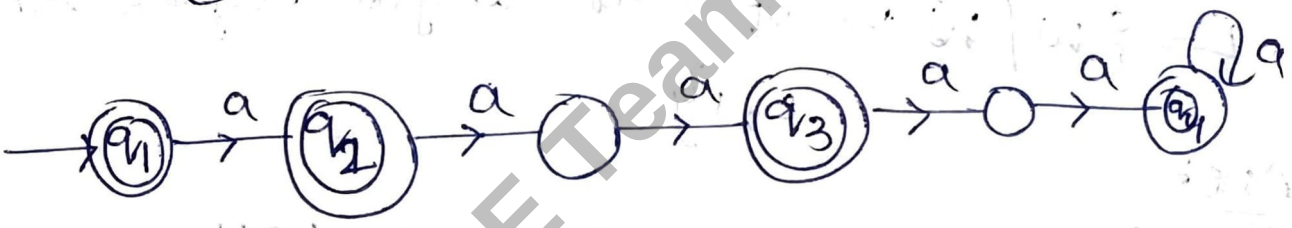
ii) $L = \{a^n / n \geq 0, n \neq 2, n \neq 4\}$

Ans:-

i) $L = \{\epsilon, a, a^2, a^4, a^5, \ldots\}$



ii) $L = \{\epsilon, a, a^3, a^5, \ldots\}$



⇒ Design min DFA over $\Sigma = \{a, b\}$ such that Every string Length must be

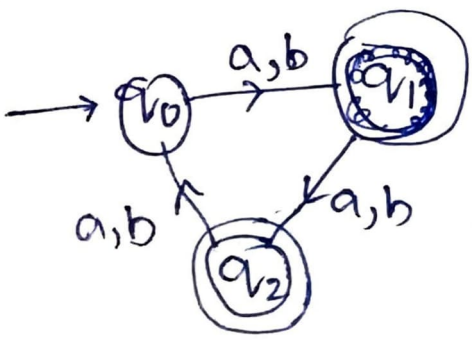
i) $|w| = 2 \pmod 3$

ii) $|w| = 3 \pmod 4$

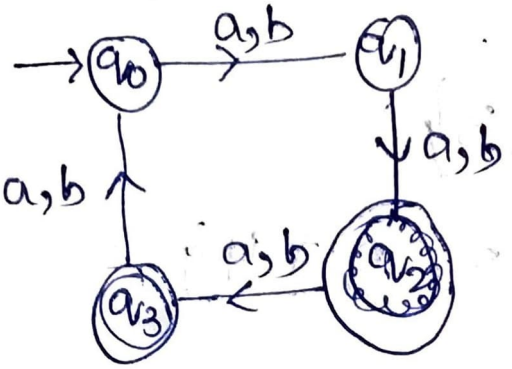
iii) $|w| = 4 \pmod 5$

Ans:-

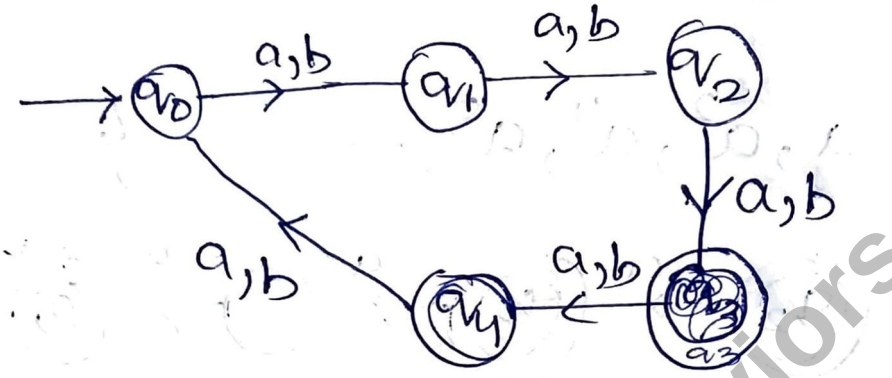
i)



(ii)



(iii)

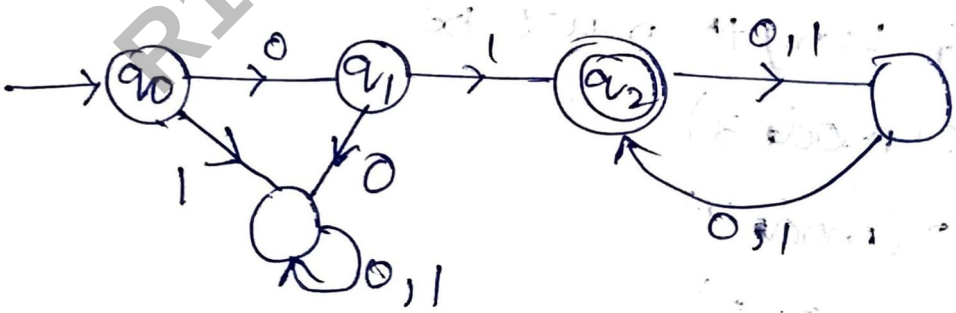


⇒ Design DFA to accept a language

$L = \{w / w$ is of even length begins with $0,1\}$

Ans:

$L = \{01, 10, 0100, 0111, 1000, 1011, \cdots\}$



⇒ How many diff DFA can be design with fixed initial state over $\Sigma = \{0,1\}$ and no of states are 2.

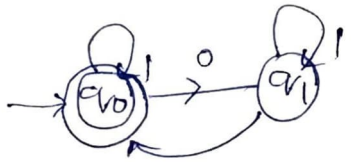
Ansk $2 \times 2^4 \times 4 = 128$

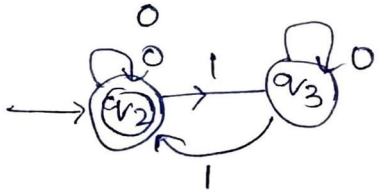
⇒ TYPE – 9

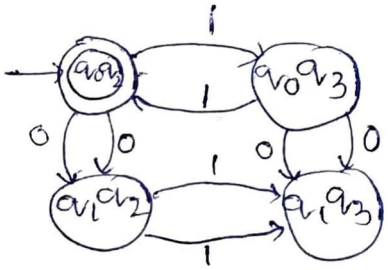
⇒ Draw a DFA a string over $\Sigma = \{0,1\}$ consistency.

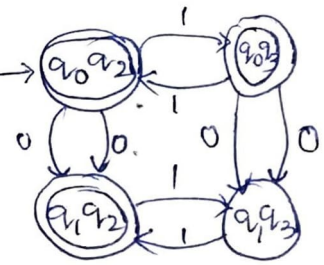
i) Even no of 0's and Even no of 1's
ii) Even no of 0's and odd no of 1's
iii) odd no of 0's and Even no of 1's
iv) odd no of 0's and odd no of 1's
v) Even no of 0's (or) Even no of 1's
vi) Even no of 0's or ~~Odd~~ no of 1's
vii) odd no of 0's or Even no of 1's
viii) odd no of 0's or odd no of 1's

Ansk

i) Even no of 0's ⇒ 

Even no of 1's ⇒ 

&& ⇒ 

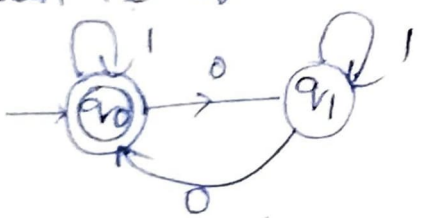
or ⇒ v) 

NOTE:-

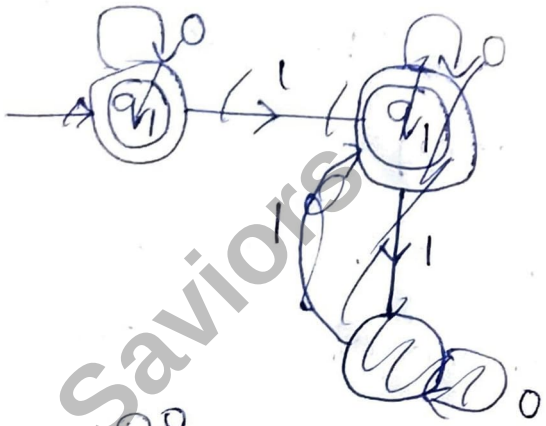
For : && — Final state whose BOTH. state are final.

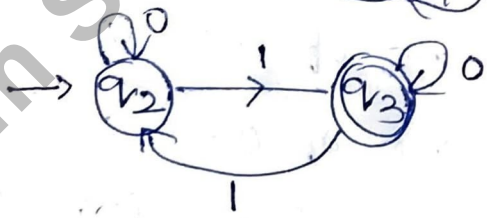
(or) — Final state only atleast one state final.

ii) Even no of 0's

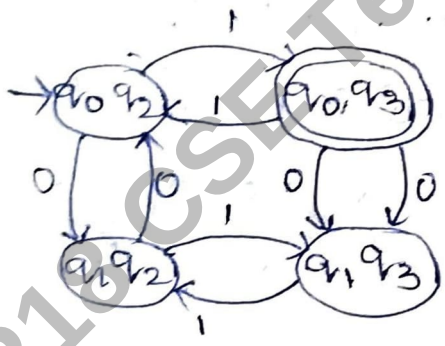


odd no of 1's

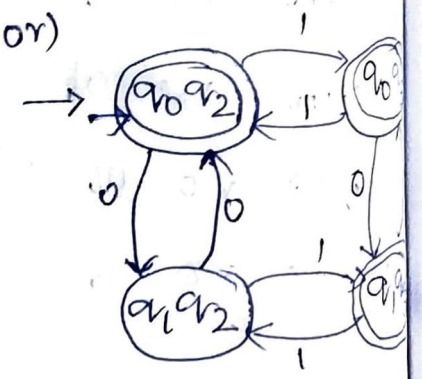




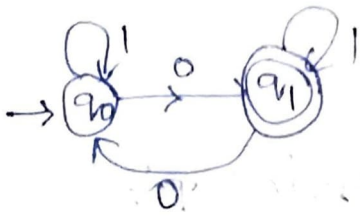
&& →

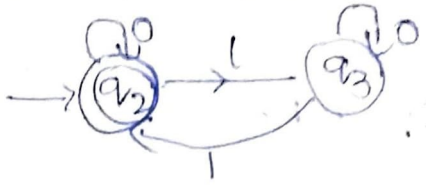


vi) or)

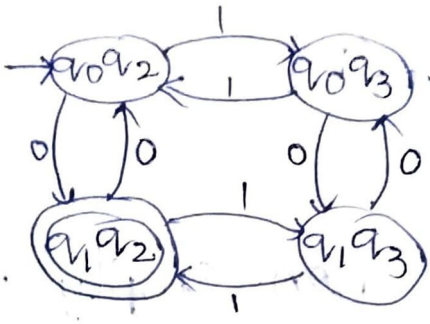


iii) odd no of 0's . Even no of 1's



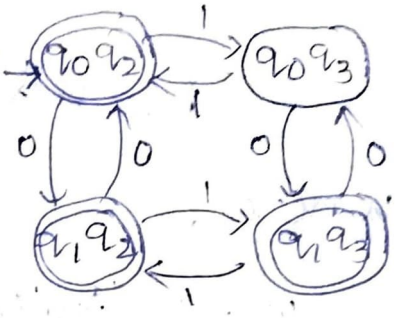


&

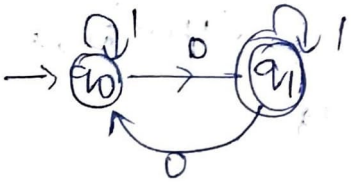


or

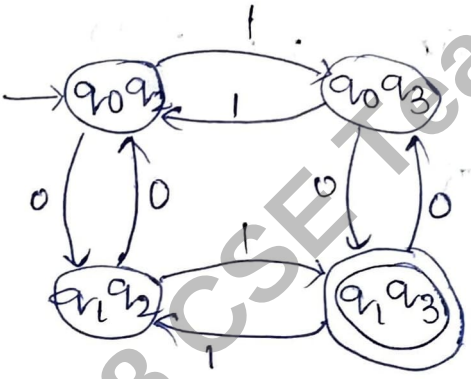
vii)



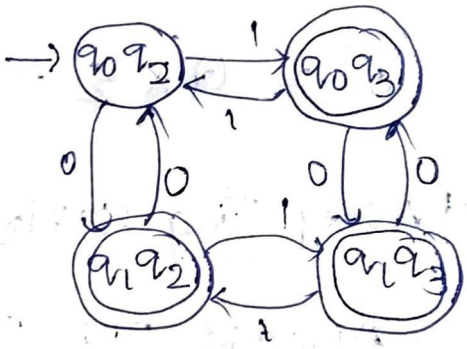
iv) odd no of 0's . odd no of 1's



&



viii) OR



Non-Determine finite Automata :- (NFA)

Notation of NFA :-

→ It is 5-tuple notation (Quintuple)

$$M = (Q, \Sigma, \delta', q_0, F)$$

where,

$Q \rightarrow$ finite set of non-empty internal nodes

$\Sigma \rightarrow$ finite non Empty set of input symbols

$q_0 \rightarrow$ Initial State, $q_0 \in Q$

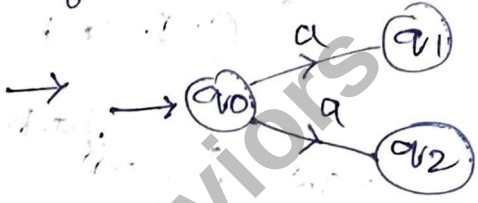
$F \rightarrow$ final state, $F \subseteq Q$

$$\delta' = Q \times \Sigma \rightarrow 2^Q$$

* Accepting String :-

→ Atleast one transition path reach final State from Initial State then String is accepted.

* Differences:

DFA	NFA
→ From a state reading any input symbol it must define a single path to any state in the design	→ A state may contain more than one transition from same input symbol.
→ Transition function in DFA. $\delta: \Sigma \times Q \rightarrow Q$	→ Transition function in NFA. $\delta: \Sigma \times Q \rightarrow 2^Q$
	

NOTE:-

1) Design NFA is easier than DFA

2) Capabilities of NFA & DFA are same.

3) Both NFA & DFA accepts the same string.

4) There is no concept of dead state in NFA.

5) In NFA, there is no need to address all input symbols.

6) Every NFA is converted to equivalent DFA

7) Every DFA is NFA.

⇒ Construct DFA & NFA which accepts a string starting with

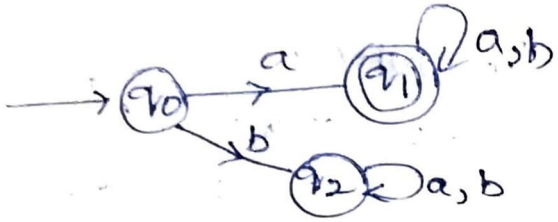
i) a

ii) ab

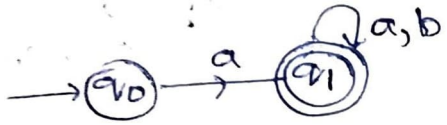
Ans:

i) $L = \{a, aa, ab, \ldots\}$

DFA:

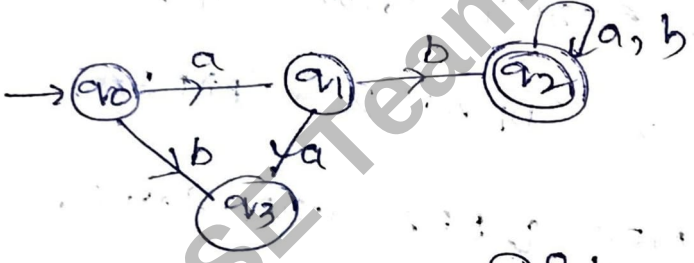


NFA :-

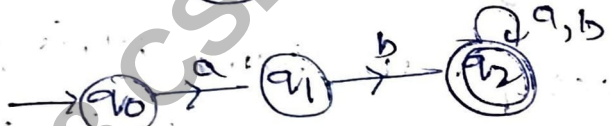


ii) $L = \{ab, aba, \ldots\}$

DFA:



NFA:



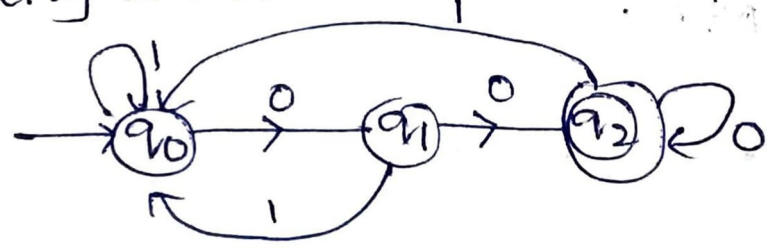
NOTE:

→ For DFA $|w| = n$, we require $(n+2)$ states in DFA.

→ For DFA $|w| = n$, we require $(n+1)$ states in NFA.

⇒ Construct NFA which accepts a string ending with 00.

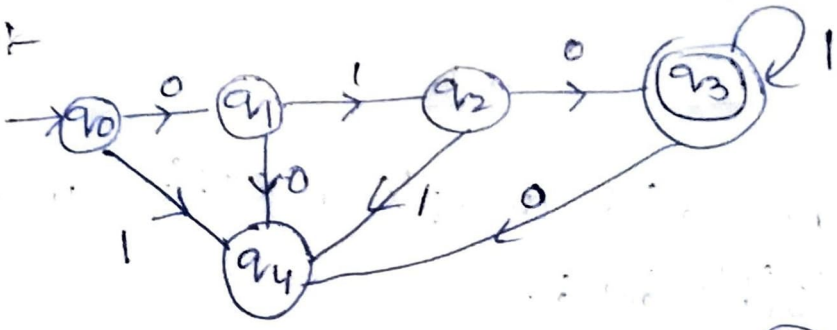
Ans:



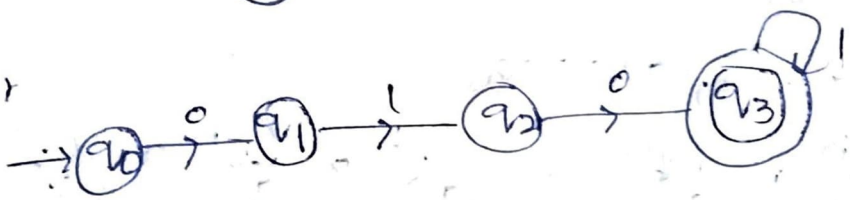
⇒ Construct NFA for the language $L = \{010^n / n \geq 0\}$

Ans: $L = \{010, 0101, 01011, \ldots\}$

DFA:-



NFA:

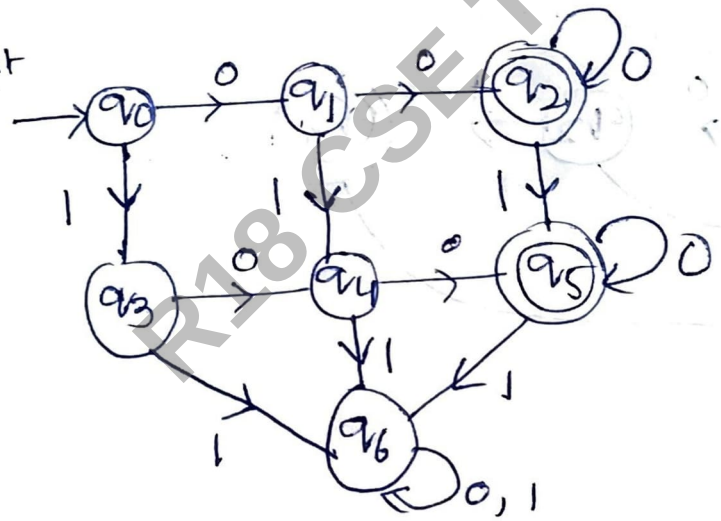


⇒ Construct DFA & NFA for

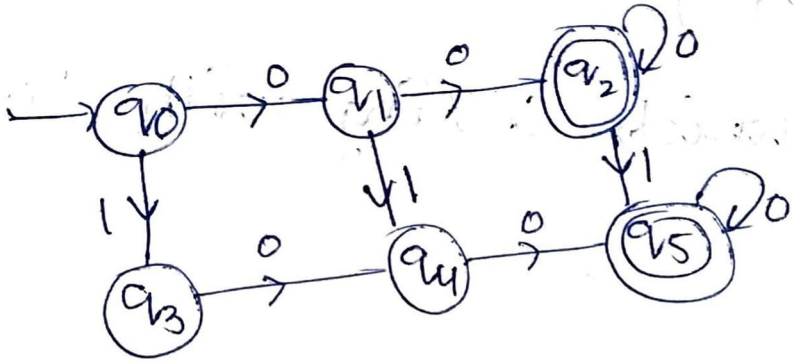
$L = \{w \in \{0,1\}^* \text{ where } w \text{ contains atleast two 0's \& atmost one 1's}\}$

Ans: $L = \{00, 100, 010, \ldots\}$

DFA:-

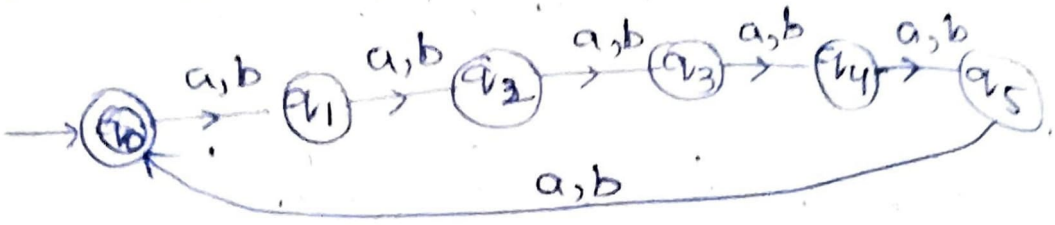


NFA:-



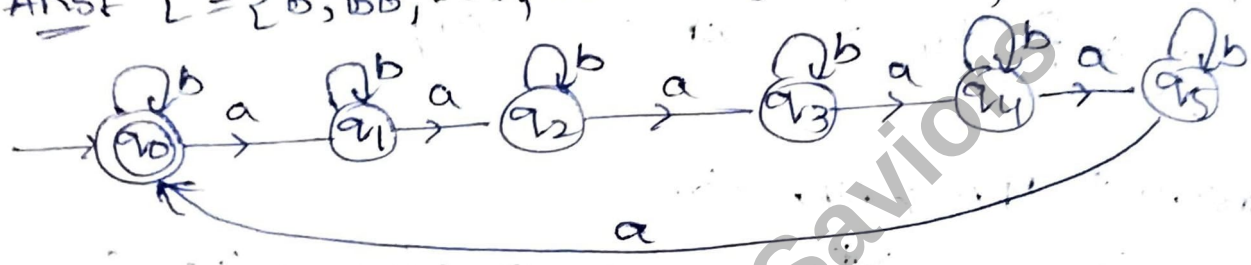
⇒ Construct DFA & NFA which accepts a string whose length is divisible by 6.

Ans:

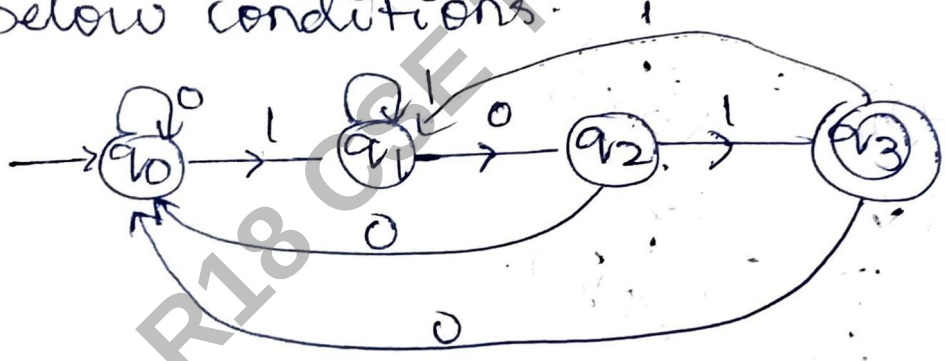


⇒ construct NFA which accepts a string in which no of a's divisible by 6.

Ans: $L = \{b, bb, \ldots, aaaaaa, \ldots\}$



⇒ Find a string which satisfies the below conditions.



Ans:

$L = \{0, 1\}$

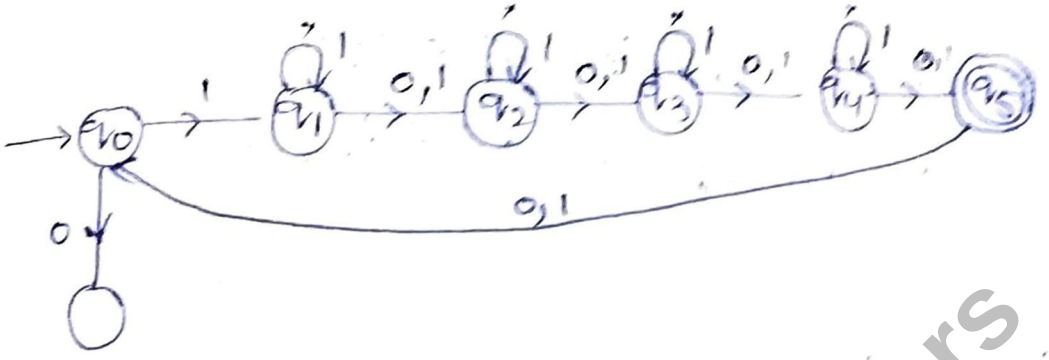
String in which Ending with 101

⇒ Construct NFA for the set of all strings over the $\Sigma = \{0, 1\}$ where 5th symbol from right is 1.

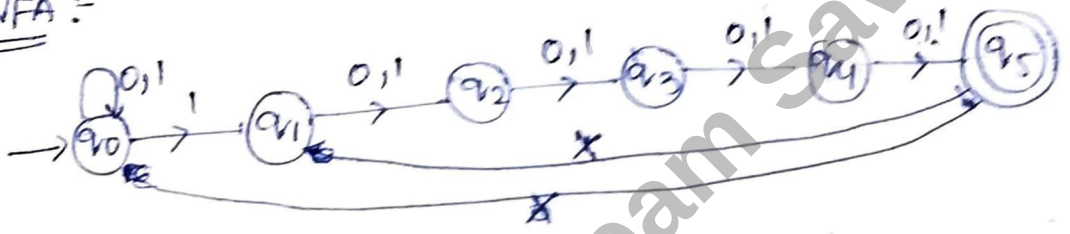
Ans:-

$$L = \{xx1000, xx\ 11000, \ldots\}$$

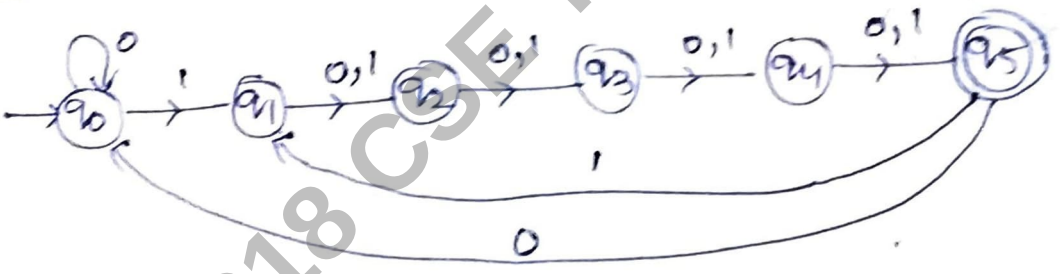
(x)



NFA:-



DFA:-



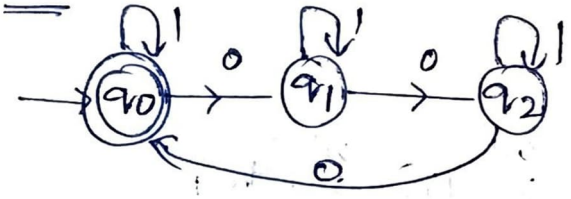
⇒ Find NFA which accepts the set of all strings over $\Sigma = \{0,1\}$ in which number of occurances of 0 is divisible by 3 and the number of occurances of 1 is divisible by 2.

Ans:-

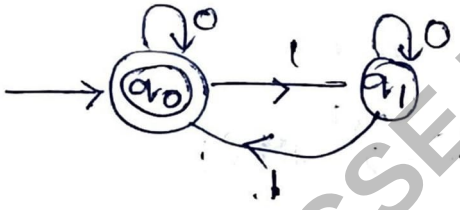
Cond 1 :- no of 0's divisible 3.

cond 2 :- no of 1's divisible 2

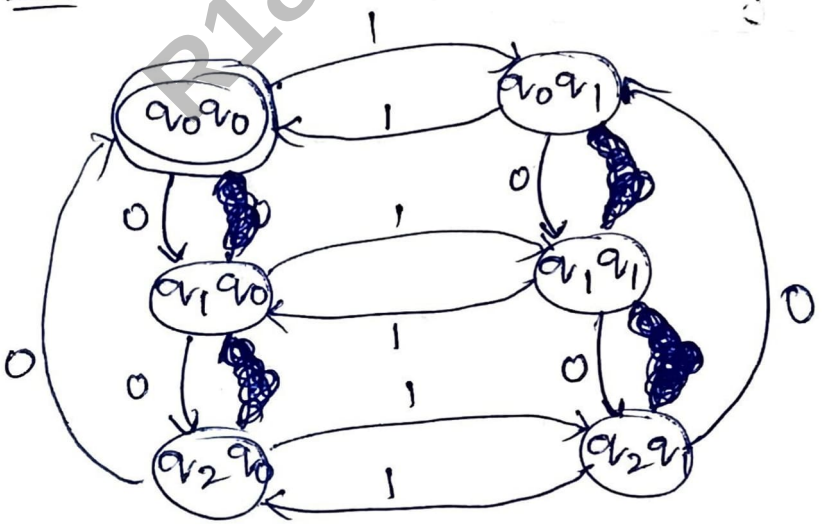
cond 1 :-



Cond 2 :-



&& :-



⇒ Draw a transition diagram for a given Automata finite data.

$$M = \{(A,B,C,D), \{0,1\}, \delta, C, (A,C)\}$$

(A,B,C,D) → Q; {0,1} → Σ; C → initial; (A,C) → final

$\delta(A,0) = \delta(A,1) = \{A,B,C\}$

$\delta(B,0) = B$, $\delta(B,1) = \{A,C\}$

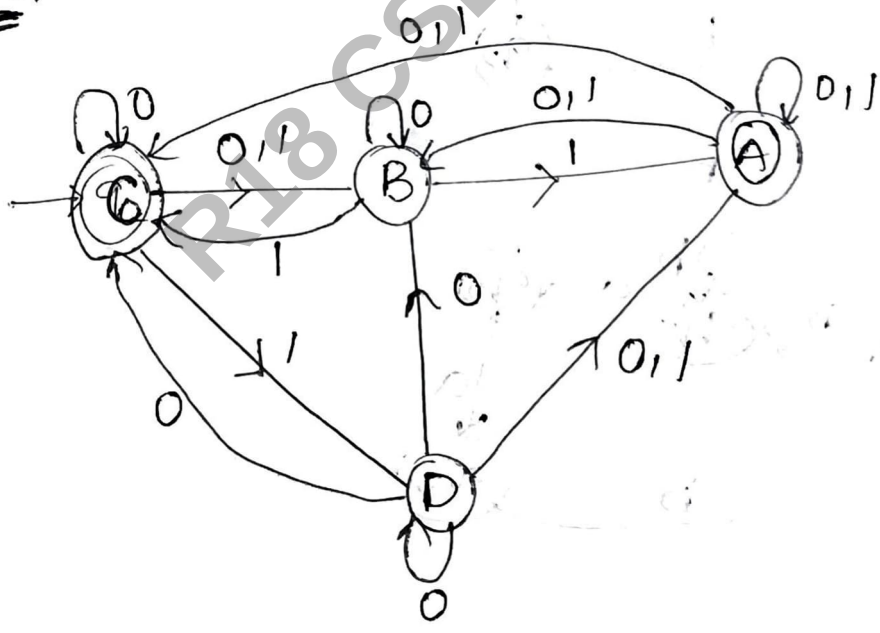
$\delta(C,0) = \{B,C\}$

$\delta(C,1) = \{B,D\}$

$\delta(D,0) = \{A,B,C,D\}$

$\delta(D,1) = \{A\}$

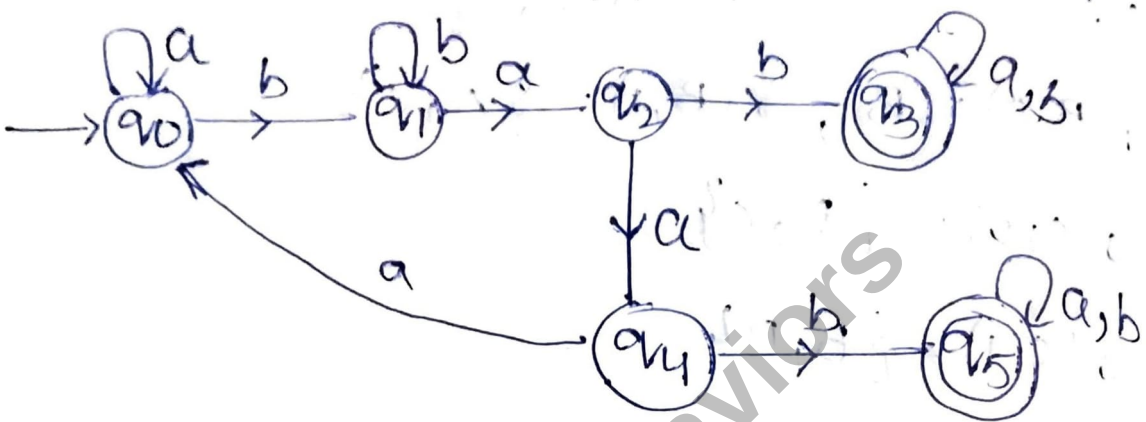
Ans:-



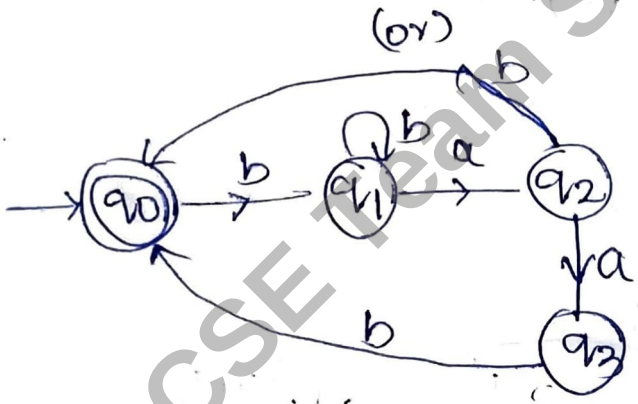
⇒ Design NFA accepts the set of all strings containing an occurance of pattern baB (or) baaB.

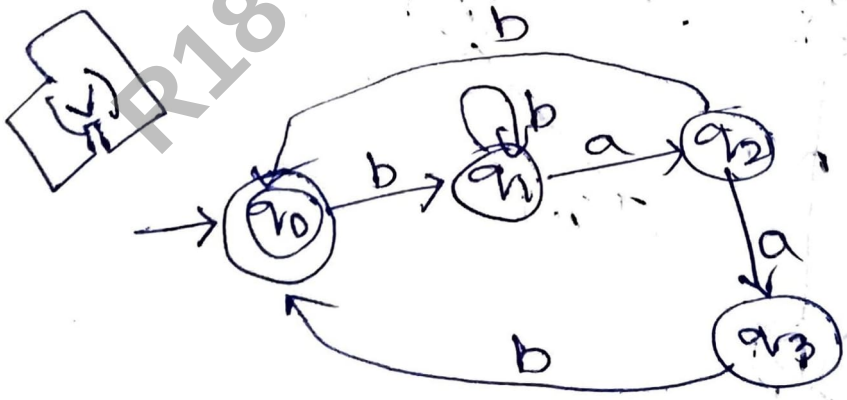
Ans:-

$L = \{ baB, baaB, \cdots \}$



(or)





NFA with ∈-moves :-

Notation in 5-tuple

$$M = (Q, \Sigma, \delta, q_0, F)$$

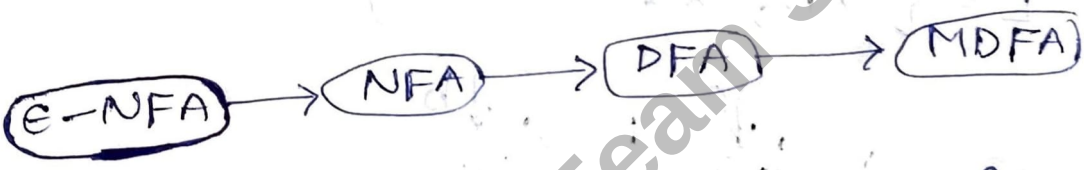
Q = finite non-Empty set internal nodes

Σ = finite non-Empty set i/p symbols

q_0 = initial state, $q_0 \in Q$

F = final state, $F \subseteq Q$

$$\delta = Q \times \Sigma^{\cup\{\epsilon\}} \longrightarrow 2^Q$$



NFA

1) $\delta : Q \times \Sigma \longrightarrow 2^Q$

2) More than one state can be possible on the same input symbol.

NFA with ∈-moves

1) $\delta : Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$

2) for given input there would be more than one transition including ∈ from a state.

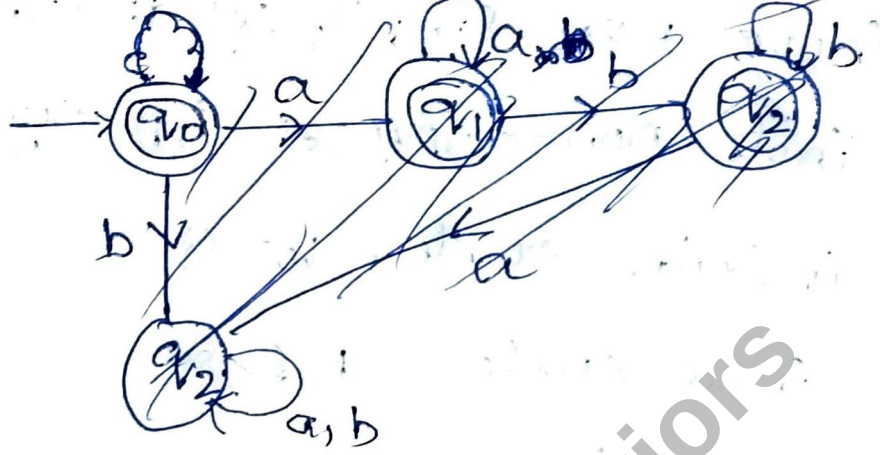
⇒ Draw an NFA with ε-moves for

$$L = \{a^n b^m / n, m \geq 0\}$$

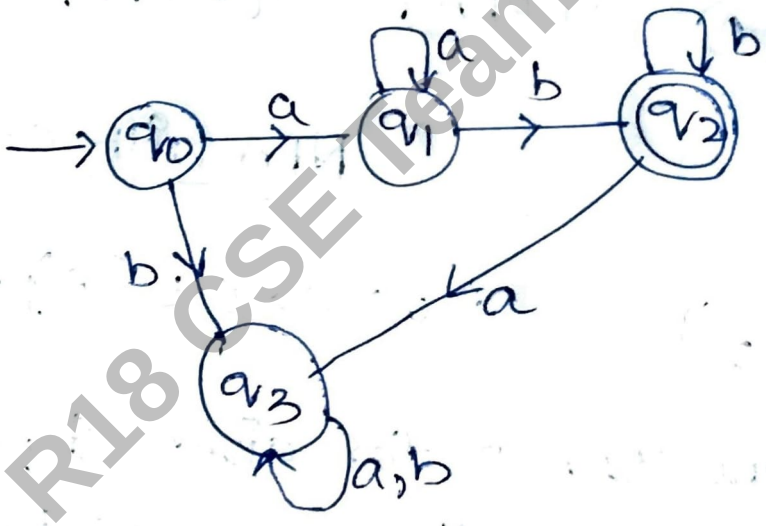
Ans:

$L = \{\epsilon, aaa, ba, b, \ldots\}$

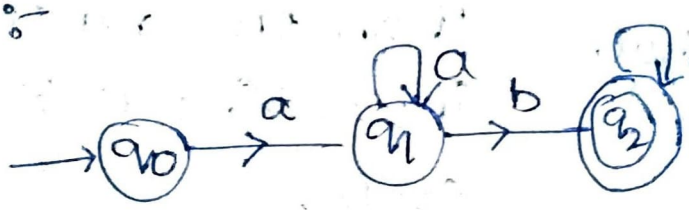
DFA:



DFA :- $L = \{ab, a^2b^2, \ldots\}$



NFA :-

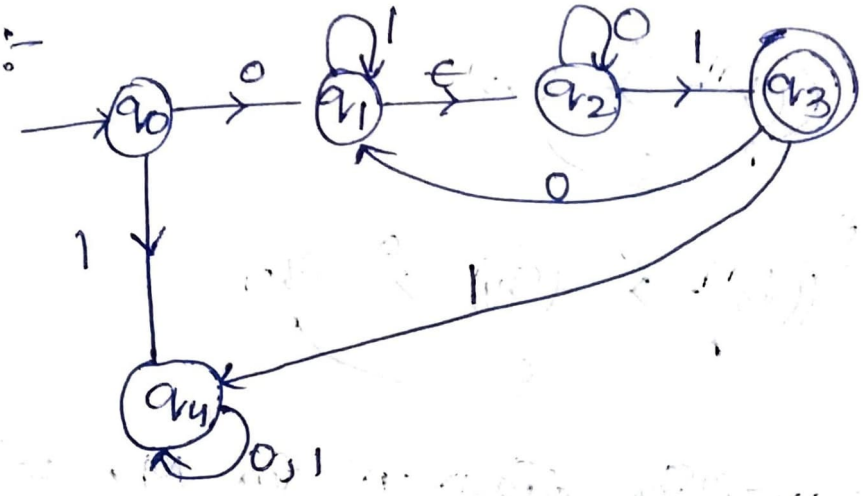


ε-NFA:

⇒ Design NFA with ϵ-moves for a

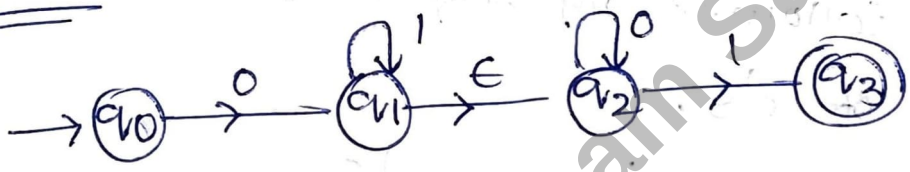
$$L(M) = 01^*0^*1$$

Ans:-



DFA

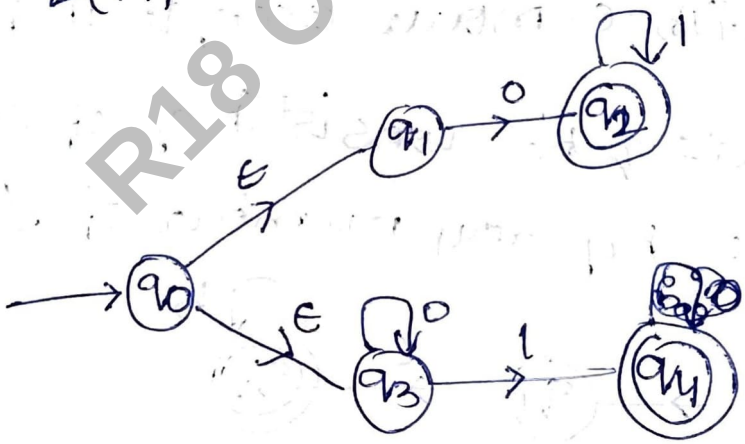
NFA — ϵ



⇒ Design NFA with ϵ-moves for a

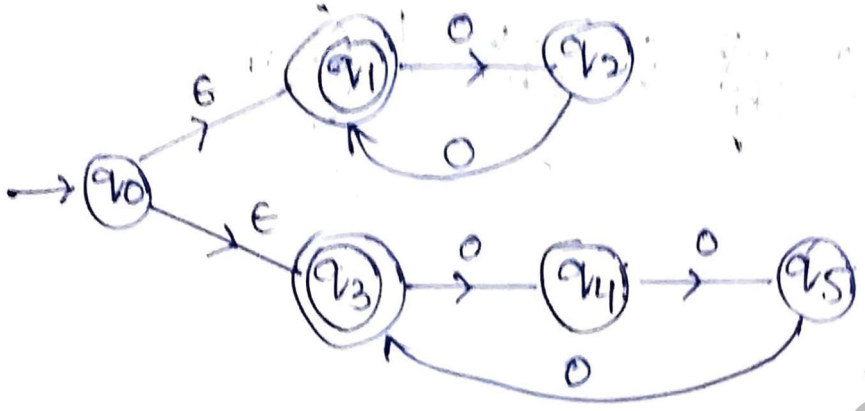
$$L(M) = 01^*/0^*1$$

Ans:-



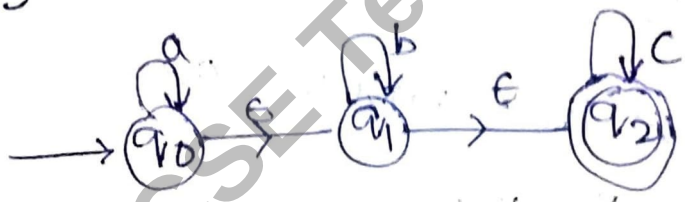
⇒ Design ε-NFA for a $L = \{0^K / K$ is multiple of 2 or 3$\}$

Ans:-



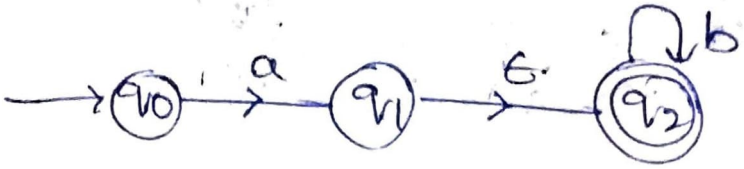
⇒ Draw a transition diagram for NFA with ε moves & accepts a language consisting any no of a's followed by any no of b's followed by any no of c's.

Ans:-



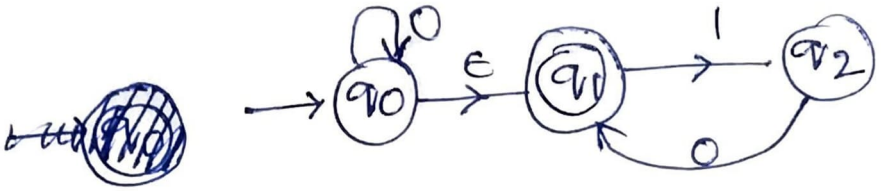
⇒ Design NFA with ε-moves accepts input string with a's & b's consisting of single a followed by any number of b's.

Ans:-



⇒ Design NFA with ϵ-moves for a language consisting of $L = 0^*(10)^*$

Ans:-

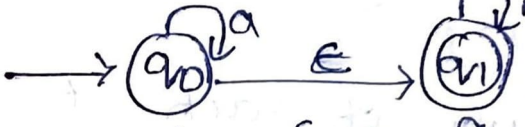


UNIT-II

CONVERT ∈-NFA to NFA without ∈ :-

∈-closure :-

→ ∈-closure of state defined as, it is a set of all states p such that there is a path from q to p labelled with ∈ including itself.



Ex :- ∈-closure(q_0) = {q_0, q_1}

∈-closure(q_1) = {q_1}

Procedure :-

Step ① :

→ calculate ∈-closures of all states.

Step ② :

→ calculate Extended transition function ($\hat{\delta}$)

$$\hat{\delta}(q, a) = \text{∈-closure}(\delta(\hat{\delta}(q, \epsilon), a))$$

$$\hat{\delta}(q, \epsilon) = \text{∈-closure}(q)$$

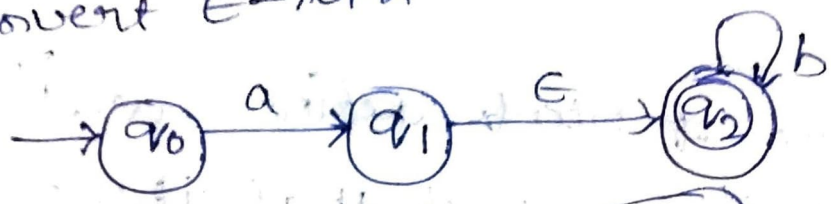
Step ③ :

→ F' is a set of all states where ∈-closure contains a final state F.

Ex:-

Problems on conversion of ϵ-NFA to NFA without ϵ.

⇒ Convert ϵ-NFA to NFA



$\epsilon^+ \cdot a \cdot \epsilon^+$

Ans:-

Step-①:

ϵ-closure of each states

$\epsilon\text{-closure}(q_0) = \{q_0\}$

$\epsilon\text{-closure}(q_1) = \{q_1, q_2\}$

$\epsilon\text{-closure}(q_2) = \{q_2\}$

Step-②:

calculate transition function

$\hat{\delta}(q_0, a) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon), a))$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), a))$

$= \epsilon\text{-closure}(\delta(q_0, a))$

$= \epsilon\text{-closure}(q_1)$

$= \{q_1, q_2\}$

$\hat{\delta}(q_0, b) = \epsilon\text{-closure}(\delta(q_0, b))$

$= \epsilon\text{-closure}(\phi)$

$= \{\phi\}$

$\hat{\delta}(q_1, a) = \epsilon\text{-closure}(\delta(q_1, a))$

$= \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a))$

$= \{\phi\}$

$\hat{\delta}(q_1, b) = \epsilon\text{-closure}(\delta(q_1, b))$

$= \epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b))$

$= \{q_2\}$

$\hat{\delta}(q_2, a) = \epsilon\text{-closure}(\delta(q_2, a))$

$= \epsilon\text{-closure}(\phi)$

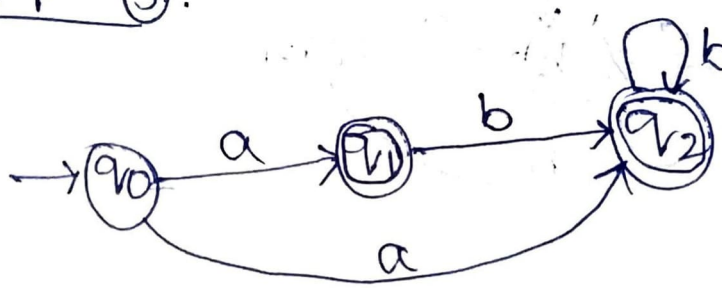
$= \{\phi\}$

$\hat{\delta}(q_2, b) = \epsilon\text{-closure}(\delta(q_2, b))$

$= \epsilon\text{-closure}(q_2)$

$= \{q_2\}$

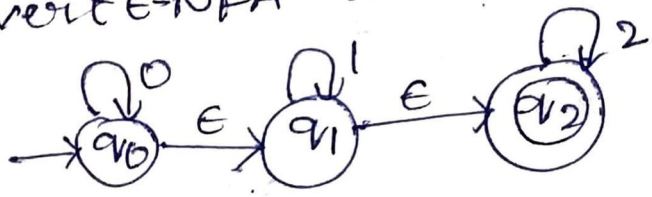
Step – (3) :-



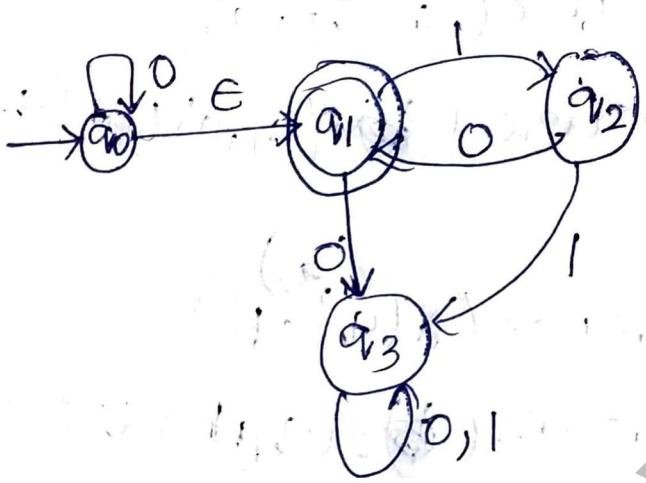
Problems:

⇒ Convert ε-NFA to NFA

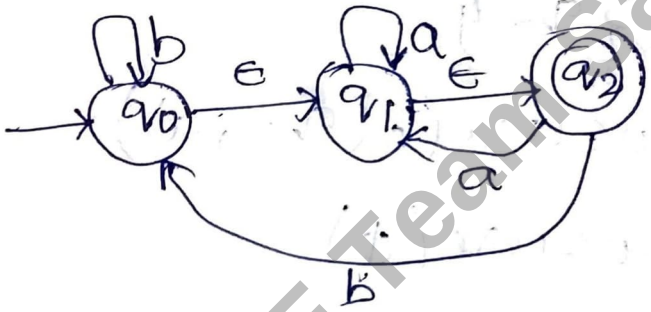
1)



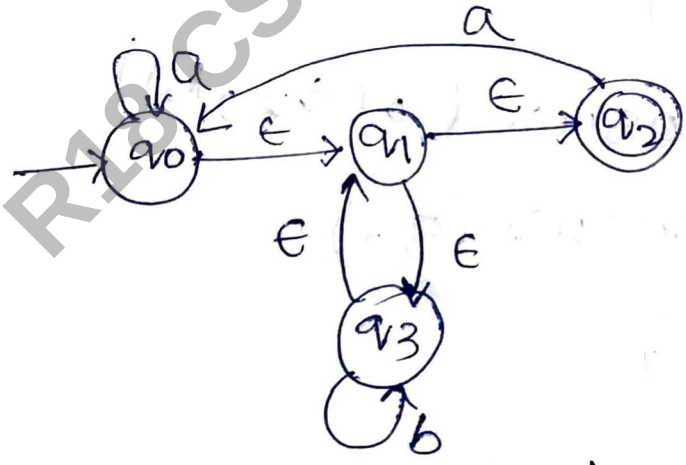
2)



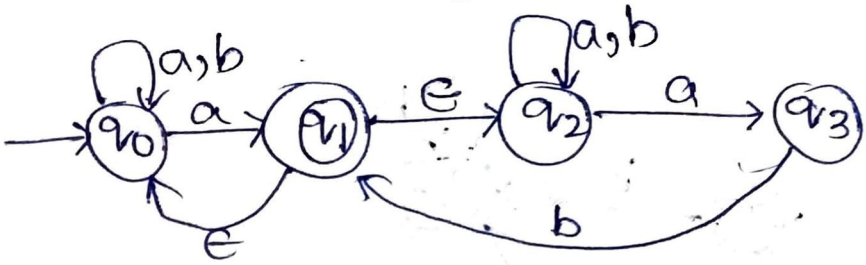
3)



4)



5)



Answers

1) step ① :-

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

step ② :-

a) $\hat{\delta}(q_0, 0) = \epsilon$-closure $(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

$= \epsilon$-closure (q_0)

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

$\hat{\delta}(q_0, 1) = \epsilon$-closure $(\delta(q_0, 1) \cup \delta(q_1, \phi) \cup \delta(q_2, \phi))$

$= \epsilon$-closure (q_1)

$= \{q_1, q_2\}$

$\hat{\delta}(q_0, 2) = \epsilon$-closure $(\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2))$

$= \epsilon$-closure (q_2)

$= \{q_2\}$

$\hat{\delta}(q_1, 0) = \epsilon$-closure $(\delta(q_1, 0) \cup \delta(q_2, 0))$

$= \phi$

$\hat{\delta}(q_1, 1) = \epsilon$-closure $(\delta(q_1, 1) \cup \delta(q_2, 1))$

$= \{q_1, q_2\}$

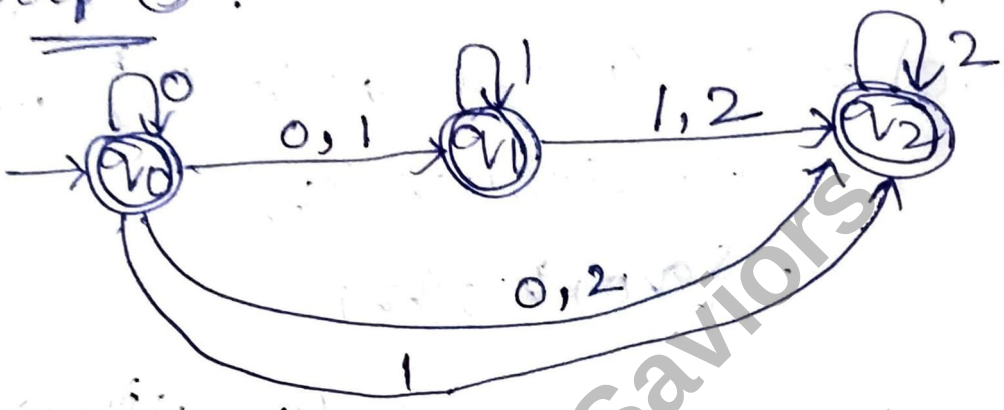
$\hat{\delta}(q_1, 2) = \{q_2\}$

$\hat{\delta}(q_2, 0) = \{\phi\}$

$\hat{\delta}(q_2, 1) = \{\phi\}$

$\hat{\delta}(q_2, 2) = \{q_2\}$

Step ③ :



2) Answer

Step ① :

ϵ-closure $(q_0) = \{q_0, q_1\}$

ϵ-closure $(q_1) = \{q_1\}$

ϵ-closure $(q_2) = \{q_2\}$

ϵ-closure $(q_3) = \{q_3\}$

Step ② :

$\hat{\delta}(q_0, 0) = \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0))$

$= \epsilon\text{-closure}(q_0 \cup q_3)$

$= \{q_0, q_1, q_3\}$

$\hat{\delta}(q_0, 1) = \epsilon\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1))$

$= \epsilon\text{-closure}(\phi \cup q_2)$

$= \{q_2\}$

$\hat{\delta}(q_1,0) = \epsilon\text{-closure}(\delta(q_1,0))$

$= \epsilon\text{-closure}(q_3)$

$= \{q_3\}$

$\hat{\delta}(q_1,1) = \epsilon\text{-closure}(\delta(q_1,1))$

$= \epsilon\text{-closure}(q_2)$

$= \{q_2\}$

$\hat{\delta}(q_2,0) = \epsilon\text{-closure}(\delta(q_2,0))$

$= \epsilon\text{-closure}(q_1)$

$= \{q_1\}$

$\hat{\delta}(q_2,1) = \epsilon\text{-closure}(\delta(q_2,1))$

$= \epsilon\text{-closure}(q_3)$

$= \{q_3\}$

$\hat{\delta}(q_3,0) = \epsilon\text{-closure}(\delta(q_3,0))$

$= \epsilon\text{-closure}(q_3)$

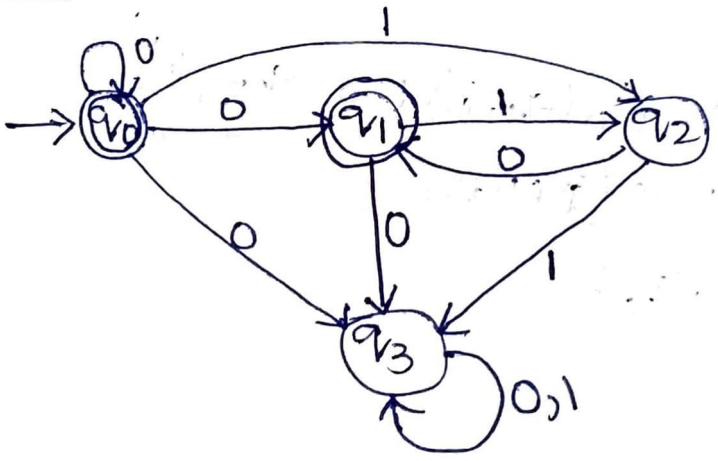
$= \{q_3\}$

$\hat{\delta}(q_3,1) = \epsilon\text{-closure}(\delta(q_3,1))$

$= \epsilon\text{-closure}(q_3)$

$= \{q_3\}$

Step-③ :-



3) Answer

Step ① :-

$$\epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$$

$$\epsilon\text{-closure}(q_1) = \{q_1, q_2\}$$

$$\epsilon\text{-closure}(q_2) = \{q_2\}$$

Step - ② :-

$$\hat{\delta}(q_0, a) = \epsilon\text{-closure}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$$

$$= \epsilon\text{-closure}(\phi \cup q_1 \cup \phi)$$

$$= \epsilon\text{-closure}(q_1)$$

$$= \{q_1, q_2\}$$

$$\hat{\delta}(q_0, b) = \epsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b))$$

$$= \epsilon\text{-closure}(q_0 \cup \phi \cup \phi)$$

$$= \epsilon\text{-closure}(q_0)$$

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

$$\hat{\delta}(q_1, a) = \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a))$$

$$= \epsilon\text{-closure}(q_1)$$

$$= \{q_1, q_2\}$$

$$\hat{\delta}(q_1, b) = \epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b))$$

$$= \epsilon\text{-closure}(\phi \cup q_0)$$

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

$\delta(q_2, a) = \epsilon\text{-closure}(\delta(q_2, a))$

$= \epsilon\text{-closure}(q_1)$

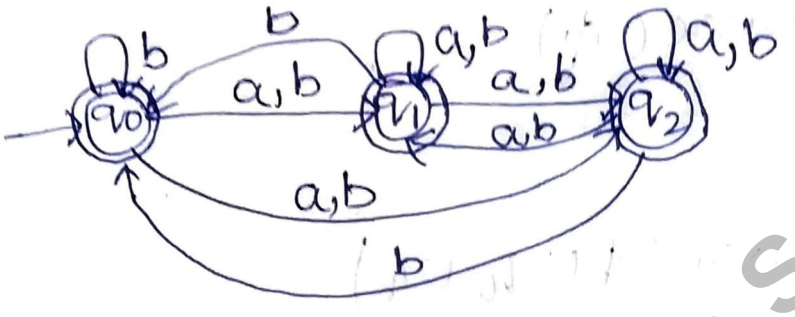
$= \{q_1, q_2\}$

$\delta(q_2, b) = \epsilon\text{-closure}(\delta(q_2, b))$

$= \epsilon\text{-closure}(q_0)$

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

step-3 :-



4th ANSWER

step ① :-

$\epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2, q_3\}$

$\epsilon\text{-closure}(q_1) = \{q_1, q_2, q_3\}$

$\epsilon\text{-closure}(q_2) = \{q_2\}$

$\epsilon\text{-closure}(q_3) = \{q_1, q_3, q_2\}$

Step-② :-

$\hat{\delta}(q_0, a) = \epsilon\text{-closure}(q_0 \cup \phi \cup q_0 \cup \phi)$

$= \epsilon\text{-closure}(q_0)$

$= \{q_0, q_1, q_2, q_3\}$

$\hat{\delta}(q_0, b) = \epsilon\text{-closure}(\phi \cup \phi \cup \phi \cup q_3)$

$= \{q_1, q_2, q_3\}$

$\hat{\delta}(q_1, a) = \epsilon\text{-closure}(\phi \cup q_0 \cup \phi)$

$= \{q_0, q_1, q_2, q_3\}$

$\hat{\delta}(q_1, b) = \epsilon\text{-closure}(\phi \cup \phi \cup q_3)$

$= \{q_1, q_2, q_3\}$

$\hat{\delta}(q_2, a) = \epsilon\text{-closure}(q_0)$

$= \{q_0, q_1, q_2, q_3\}$

$\hat{\delta}(q_2, b) = \epsilon\text{-closure}(\phi)$

$= \{\phi\}$

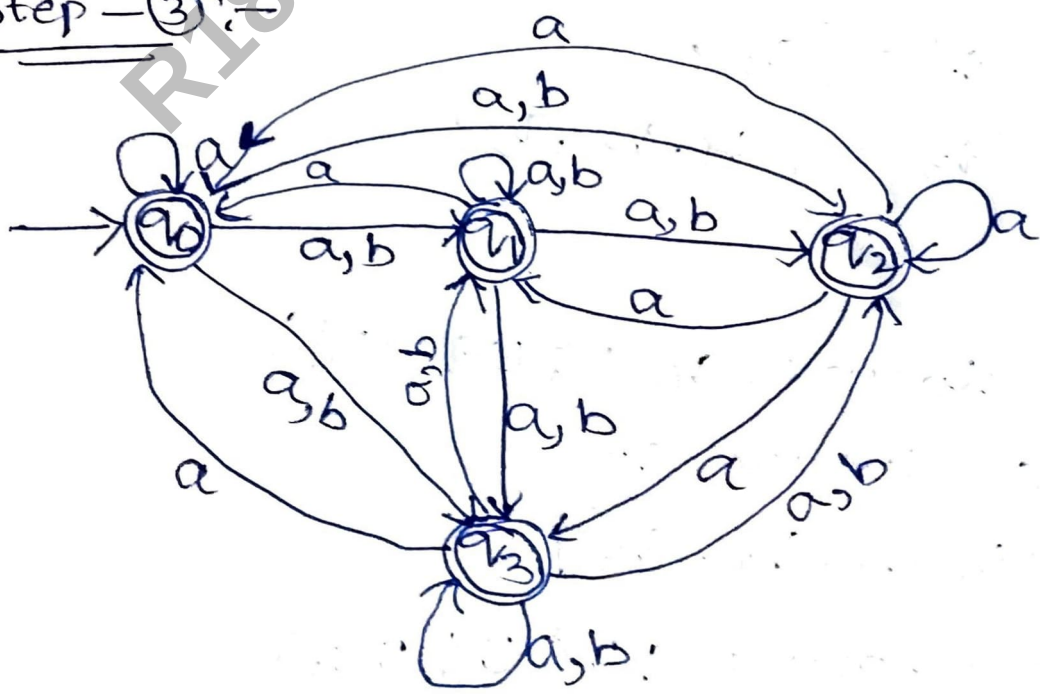
$\hat{\delta}(q_3, a) = \epsilon\text{-closure}(\phi \cup q_0 \cup \phi)$

$= \{q_0, q_1, q_2, q_3\}$

$\hat{\delta}(q_3, b) = \epsilon\text{-closure}(\phi \cup q_3 \cup \phi)$

$= \{q_1, q_2, q_3\}$

Step – (3) :–



5th Answer

Step – ① :-

ϵ-closure $(q_0) = \{q_0\}$

ϵ-closure $(q_1) = \{q_1, q_2, q_3, q_0\}$

ϵ-closure $(q_2) = \{q_2\}$

ϵ-closure $(q_3) = \{q_3\}$

Step – ② :-

$\hat{\delta}(q_0, a) = \epsilon$-closure $(q_0 \cup q_1)$

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

$\hat{\delta}(q_0, b) = \epsilon$-closure (q_0)

$= \{q_0\}$

$\hat{\delta}(q_1, a) = \epsilon$-closure $(q_0 \cup q_1 \cup q_2 \cup q_3)$

$= \{q_0, q_1, q_2, q_3\}$

$\hat{\delta}(q_1, b) = \epsilon$-closure $(q_0 \cup q_2)$

$= \{q_0, q_2\}$

$\hat{\delta}(q_2, a) = \epsilon$-closure $(q_2 \cup q_3)$

$= \{q_2, q_3\}$

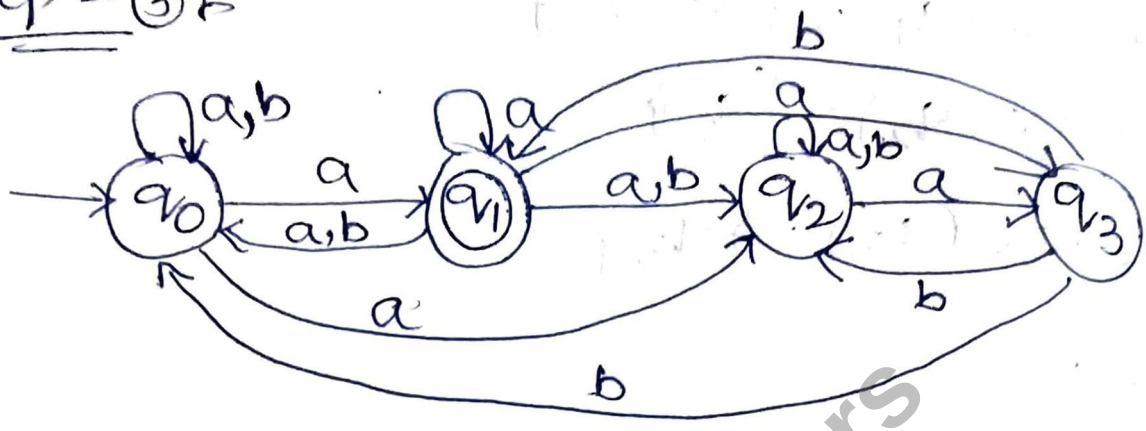
$\hat{\delta}(q_2, b) = \epsilon$-closure (q_2)

$= \{q_2\}$

$\hat{\delta}(q_3, a) = \epsilon\text{-closure}(\phi) = \phi$

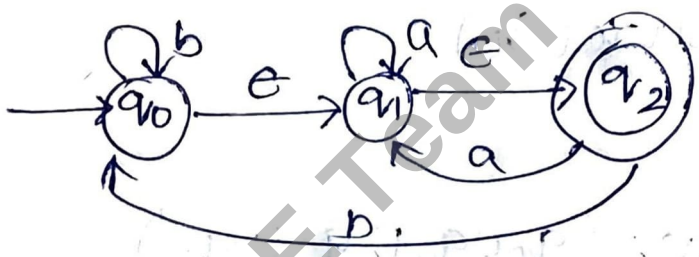
$\hat{\delta}(q_3, b) = \epsilon\text{-closure}(q_1) = \{q_0, q_1, q_2\}$

Step - (3) :-



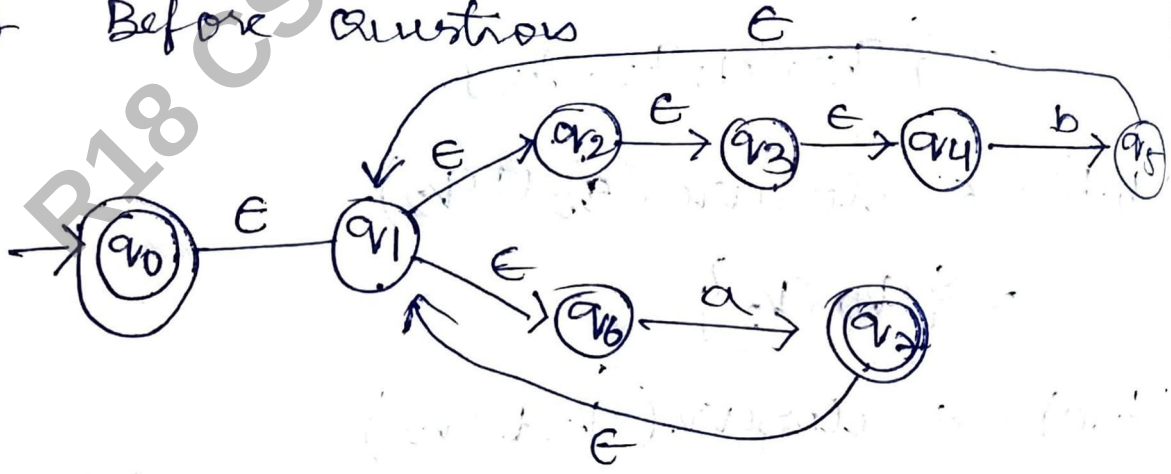
⇒ Convert ϵ-NFA to NFA without ϵ for a given problem

i)



Ans:- Before Question

ii)



Ans:-

Step-(1)

ϵ-closure$(q_0) = \{q_0, q_1, q_2, q_3, q_4, q_6\}$

ϵ-closure$(q_1) = \{q_1, q_2, q_3, q_4, q_6\}$

$(q_2) = \{q_2, q_3, q_4\}$

$(q_3) = \{q_3, q_4\}$

$(q_4) = \{q_4\}$

$(q_5) = \{q_5, q_1, q_2, q_3, q_4, q_6\}$

$(q_6) = \{q_6\}$

$(q_7) = \{q_7, q_1, q_6, q_2, q_3, q_4\}$

Step-(2)

$\delta(q_0, a) = \epsilon$-closure$(q_7)$

$= \{q_1, q_2, q_3, q_4, q_6, q_7\}$

$\delta(q_0, b) = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\delta(q_1, a) = \{q_1, q_2, q_3, q_4, q_6, q_7\}$

$\delta(q_1, b) = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\delta(q_2, a) = \{\phi\}$

$\delta(q_2, b) = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\hat{\delta}(q_3, a) = \{\phi\}$

$\hat{\delta}(q_3, b) = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\hat{\delta}(q_4, a) = \{\phi\}$

$\hat{\delta}(q_4, b) = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\hat{\delta}(q_5, a) = \{q_1, q_2, q_3, q_4, q_6, q_7\}$

$\hat{\delta}(q_5, b) = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

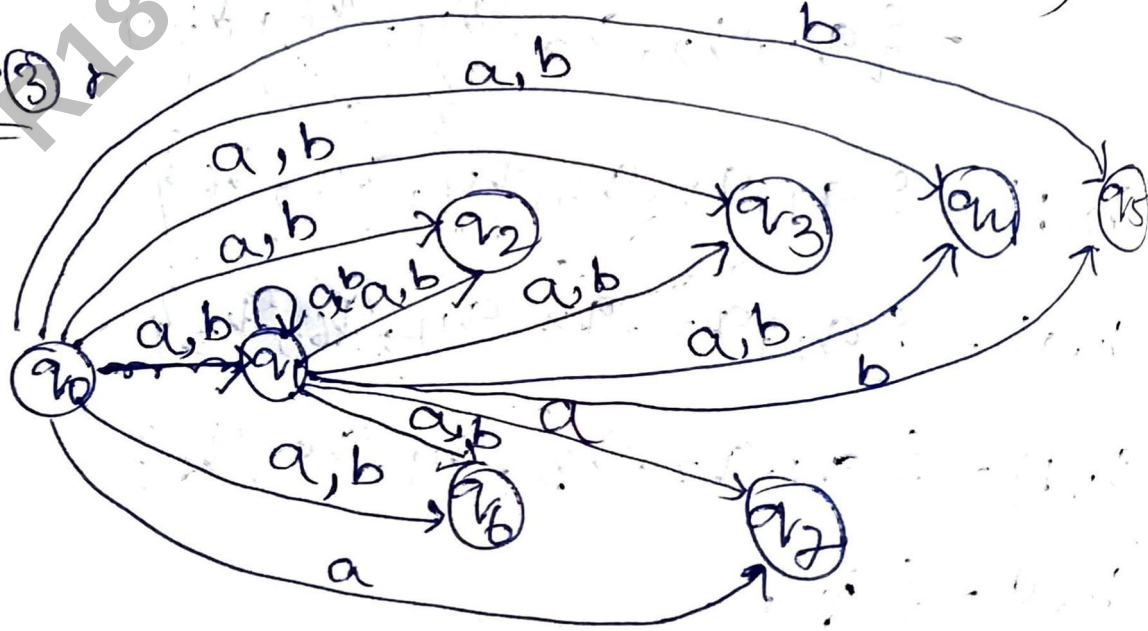
$\hat{\delta}(q_6, a) = \{q_1, q_2, q_3, q_4, q_6, q_7\}$

$\hat{\delta}(q_6, b) = \{\phi\}$

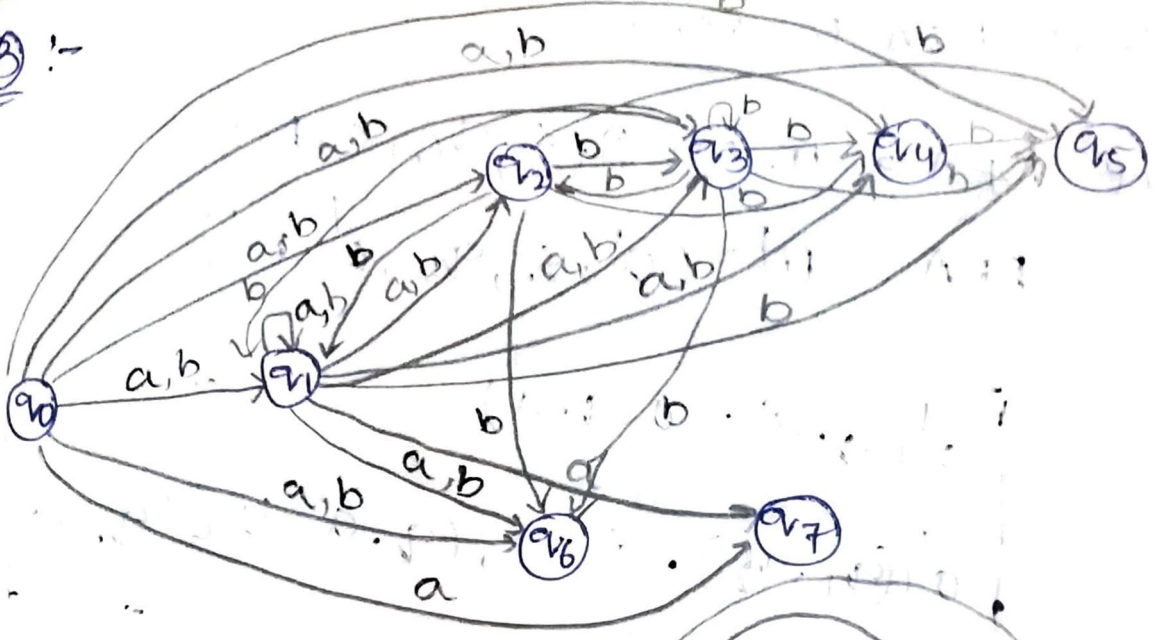
$\hat{\delta}(q_7, a) = \{q_1, q_2, q_3, q_4, q_6, q_7\}$

$\hat{\delta}(q_7, b) = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

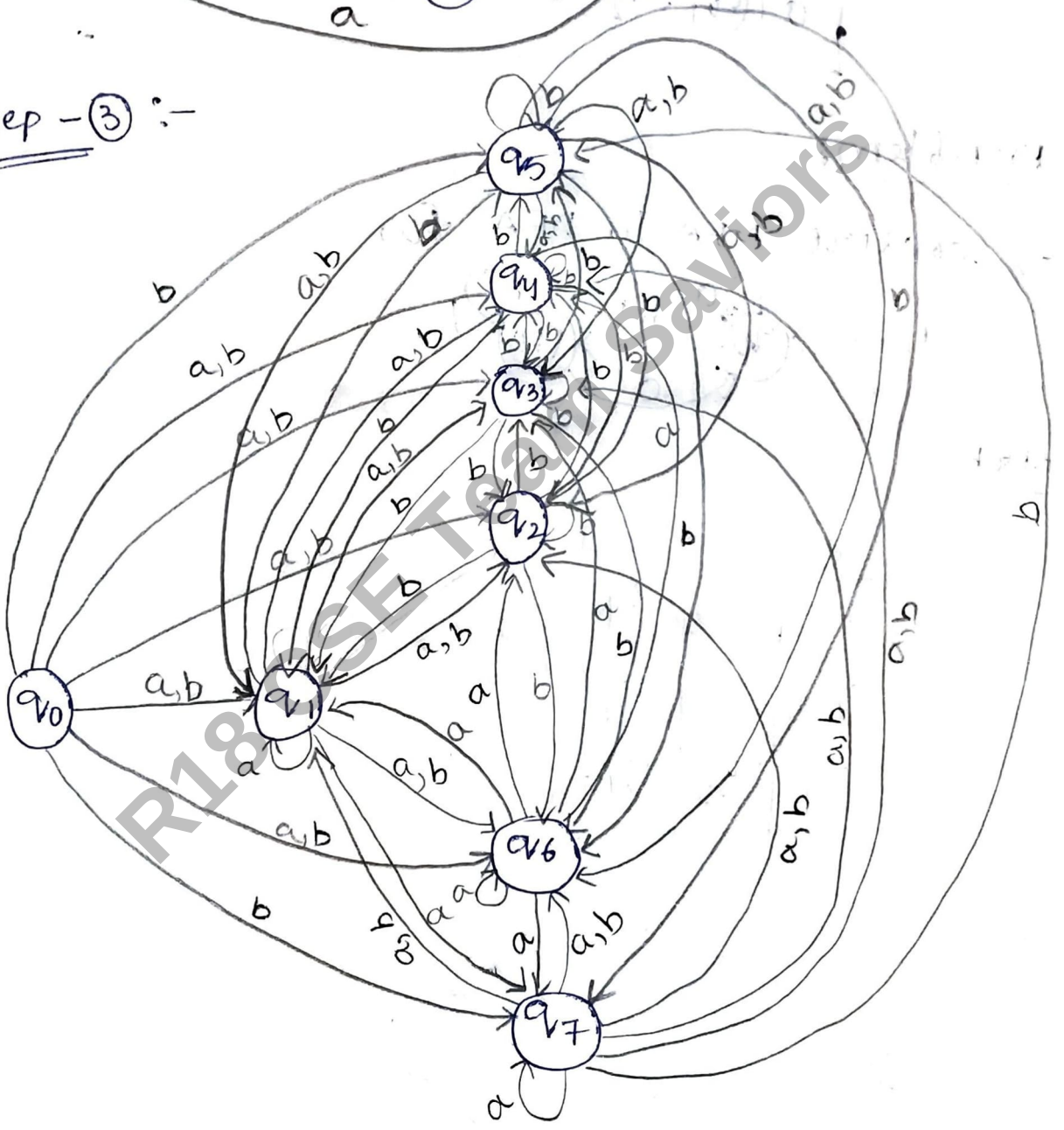
Step (3) :



(3) :-



Step – (3) :-



Convert NFA to DFA :-

Let NFA $\Rightarrow$ $M = (Q, \Sigma, q_0, \delta, F)$

DFA $M' = (Q', \Sigma, q_0, \delta', F')$

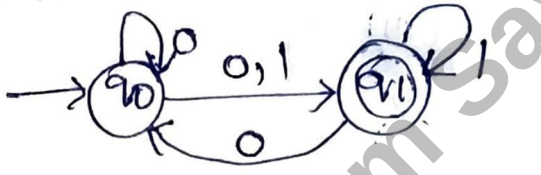
$$\delta'(q, a) = \delta(q, a)$$

$$\delta'((q_1, q_2), a) = \delta(q, a) \cup \delta(q,$$

Problem:

$\Rightarrow$ Convert NFA to DFA for

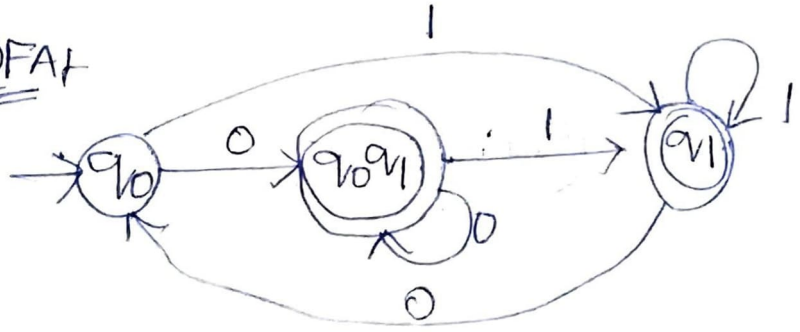
i)

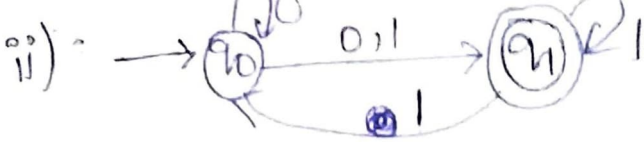


Ans:

δ'	0	1
$\rightarrow q_0$	$[q_0 q_1]$	q_1
$[q_0 q_1]$	$[q_0 q_1]$	q_1
q_1	q_0	q_1

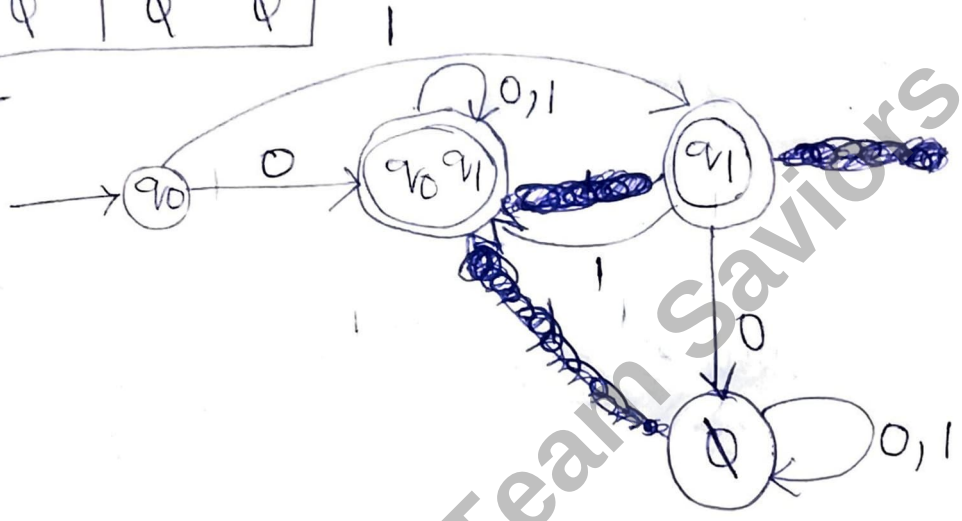
DFA:



ii) 

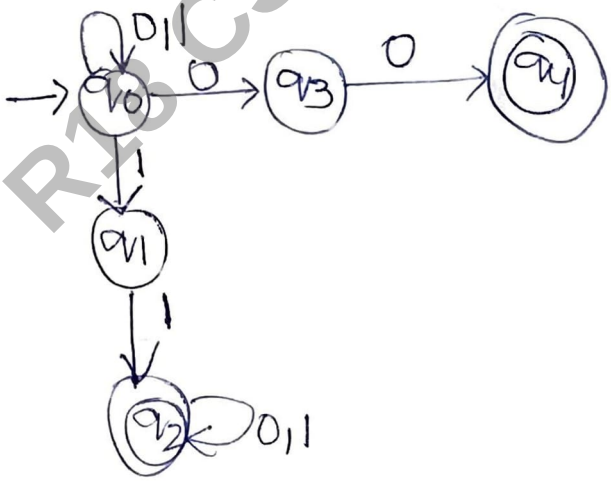
δ'	0	1
$\to q_0$	$[q_0q_1]$	q_1
$[q_0q_1]$	$[q_0q_1]$	$[q_0q_1]$
q_1	ϕ	$[q_0q_1]$
ϕ	ϕ	ϕ

DFA:-

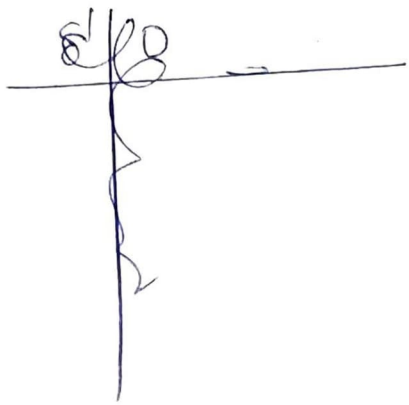


⇒ Convert NFA to DFA for (x)

(x)



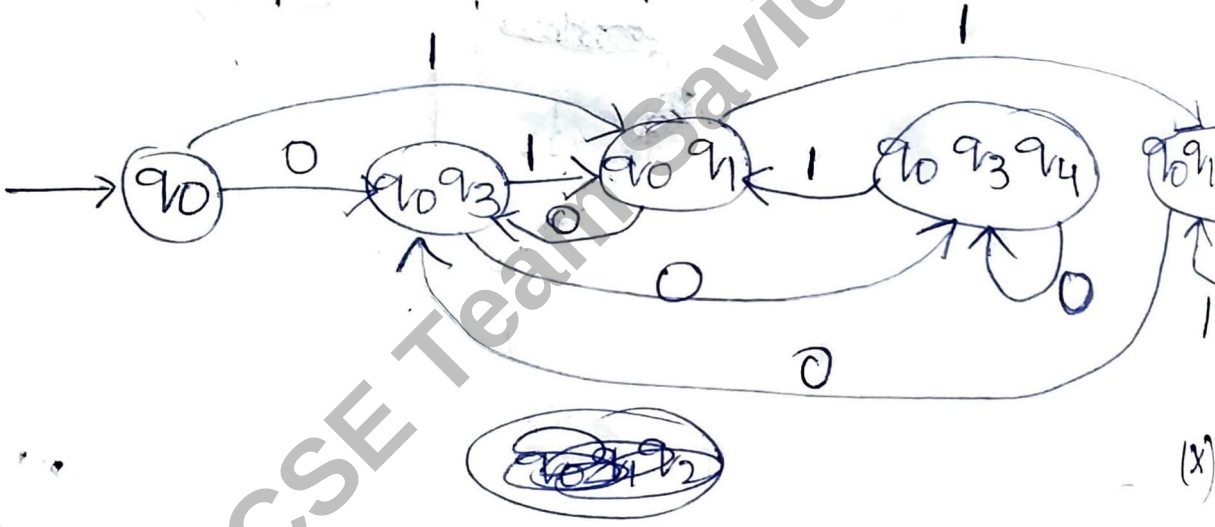
Ans



(x) (x)

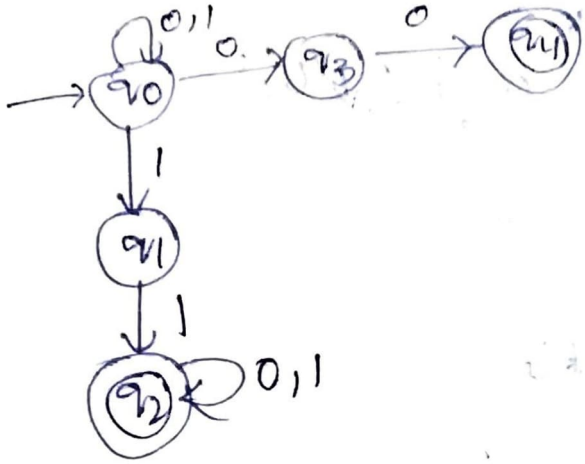
Ans: (X)

δ'	0	1
$\rightarrow q_0$	$[q_0 q_3]$	$[q_0 q_1]$
$[q_0 q_3]$	$[q_0 q_3 q_4]$	$[q_0 q_1]$
$[q_0 q_1]$	$[q_0 q_3]$	$[q_0 q_1 q_2]$
$[q_0 q_3 q_4]$	$[q_0 q_3 q_4]$	$[q_0 q_1]$
$[q_0 q_1 q_2]$	$[q_0 q_3]$	$[q_0 q_1 q_2]$



(X)

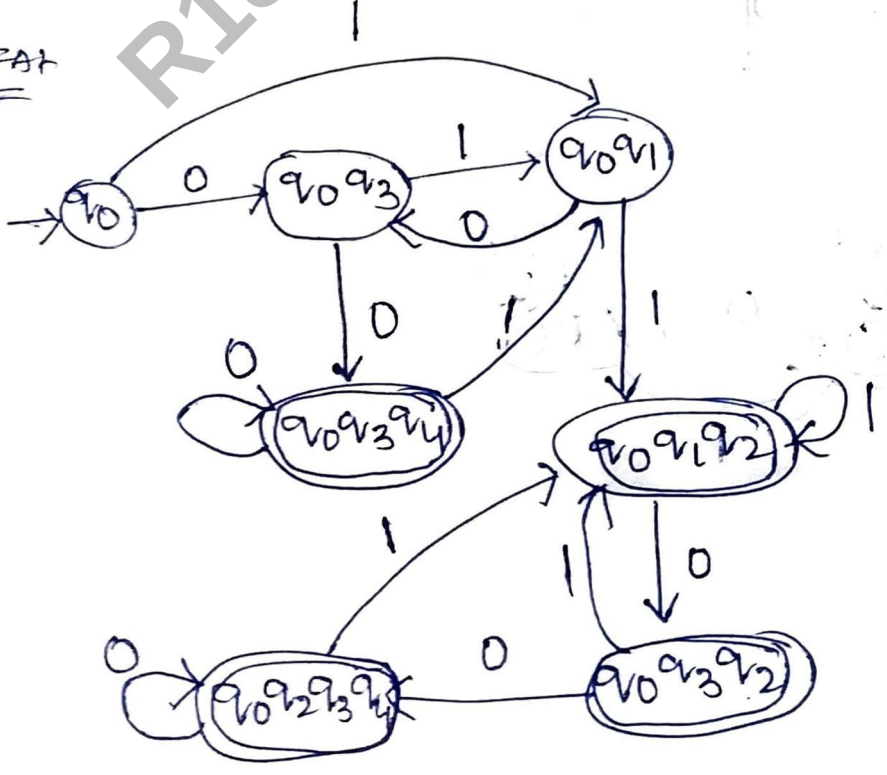
⇒ convert NFA to DFA



Ans:

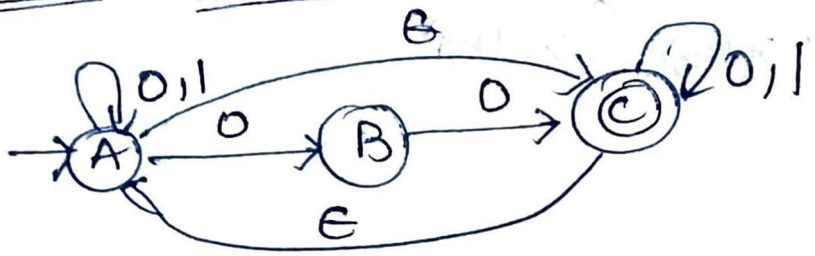
δ	0	1
→ q_0	$[q_0q_3]$	$[q_0q_1]$
$[q_0q_3]$	$[q_0q_3q_4]$	$[q_0q_1]$
$[q_0q_1]$	$[q_0q_3]$	$[q_0q_1q_2]$
$[q_0q_3q_4]$	$[q_0q_3q_4]$	$[q_0q_1]$
$[q_0q_1q_2]$	$[q_0q_3q_2]$	$[q_0q_1q_2]$
$[q_0q_3q_2]$	$[q_0q_2q_3q_4]$	$[q_0q_1q_2]$
$[q_0q_2q_3q_4]$	$[q_0q_2q_3q_4]$	$[q_0q_1q_2]$

DFA:



⇒ Convert e-NFA to DFA :-

⇒



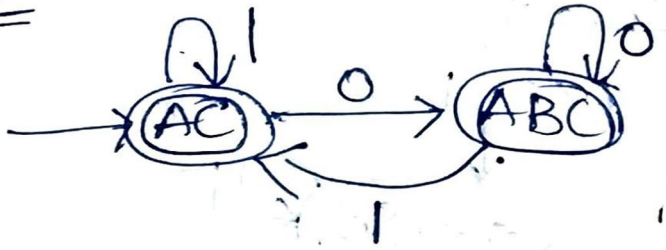
Ans:

Transitions table

δ	0	1	e-closure
→ A	AB	A	{A, C}
B	C	ϕ	{B}
(C)	C	ϕ	{A, C}

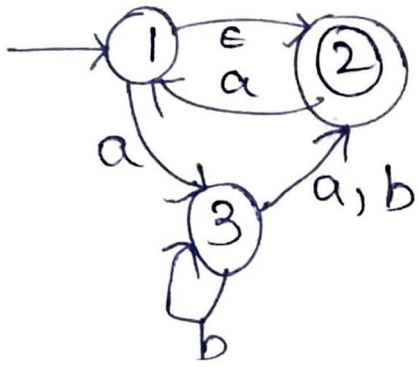
δ'	0	1
→ [AC]	[ABC]	[AC]
[ABC]	[ABC]	[AC]

DFA



ϵ-NFA to DFA

⇒

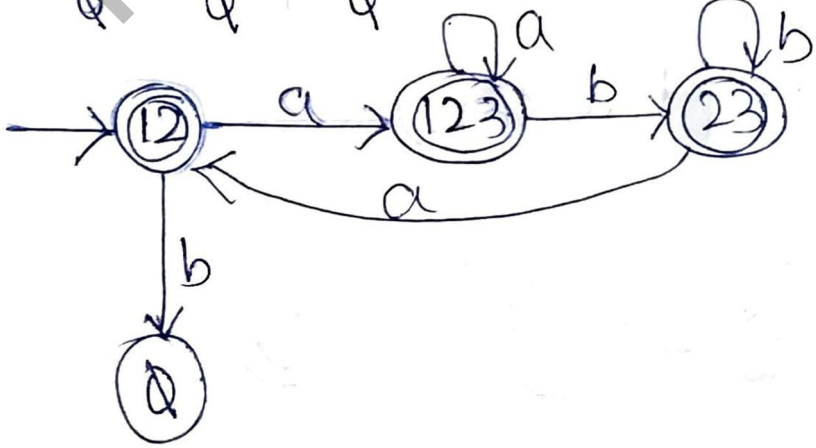


Ans:

δ	a	b	ϵ-closure
1	3	ϕ	{1,2}
(2)	1	ϕ	{2}
3	2	{2,3}	{3}

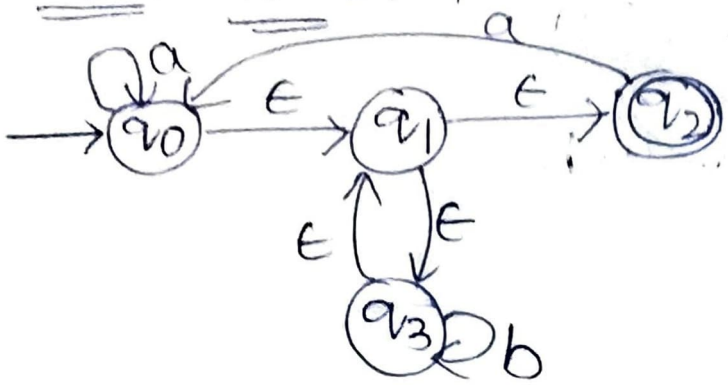
δ^+	a	b
→ [12]	[123]	ϕ
[123]	[123]	[23]
[23]	[12]	[23]
ϕ	ϕ	ϕ

DFA:



⇒ ϵ-NFA to DFA

1)



2)

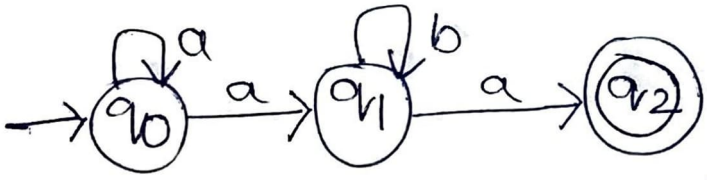
δ	0	1	ϵ
→ p	q,s	ϕ	q
q	s	q,r	r
r	s	p	ϕ
(s)	ϕ	ϕ	ϕ

⇒ convert NFA to DFA

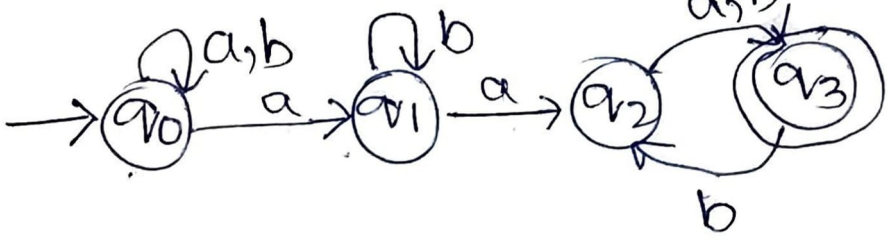
1)

	a	b
→ q_0	$q_1 q_2$	q_0
q_1	$q_0 q_1$	ϕ
(q_2)	q_1	$q_0 q_1$

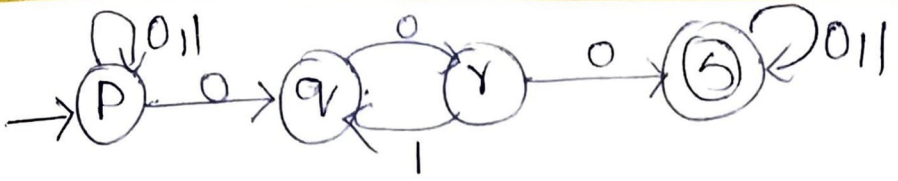
2)



3)



4)



Answer:

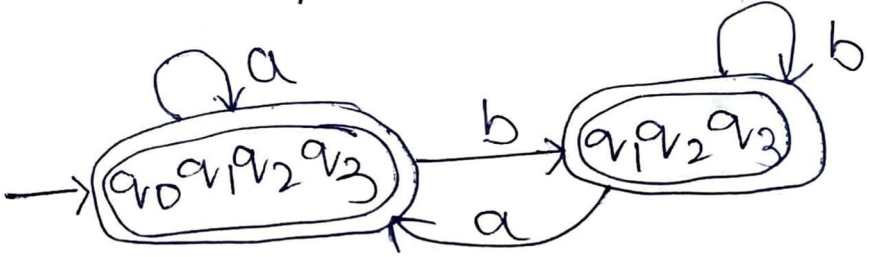
ϵ-NFA to DFA

①

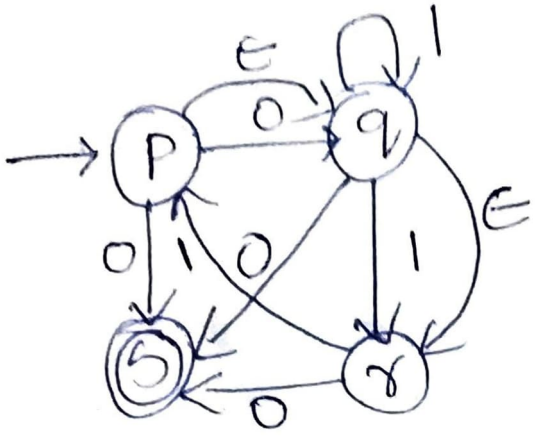
δ	a	b	ϵ-closure
→q_0	q_0	ϕ	$\{q_0, q_1, q_2, q_3\}$
q_1	ϕ	ϕ	$\{q_1, q_2, q_3\}$
q_2	q_0	ϕ	$\{q_2\}$
q_3	ϕ	q_3	$\{q_3, q_1, q_2\}$

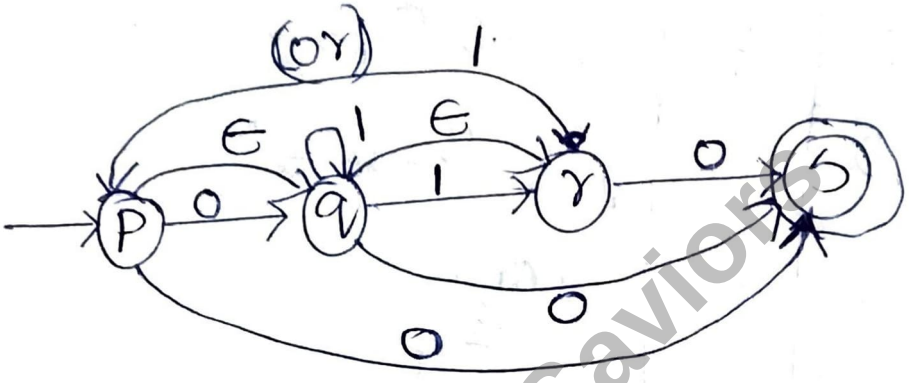
δ*	a	b
→$[q_0q_1q_2q_3]$	$[q_0q_1q_2q_3]$	$[q_3, q_1, q_2]$
$[q_1q_2q_3]$	$[q_0q_1q_2q_3]$	$[q_1q_2q_3]$

DFA:



(2)

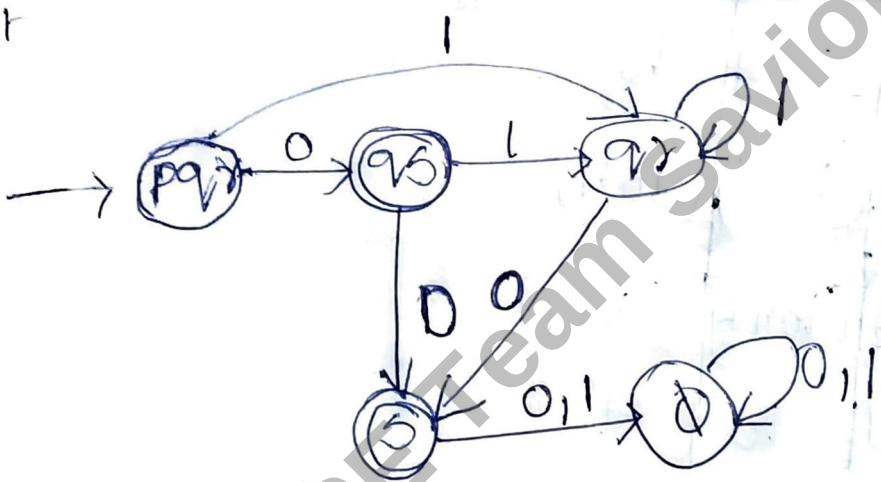




δ	0	1	ϵ-closure
p	q,5	ϕ	{p,q,r}
q	5	q,r	{q,r}
r	5	p	{r}
(5)	ϕ	ϕ	{5}

δ'	0	1
→[pqr]	[qs]	[qr]
[qs]	s	[qr]
[qr]	s	[qr]
s	ϕ	ϕ
ϕ	ϕ	ϕ

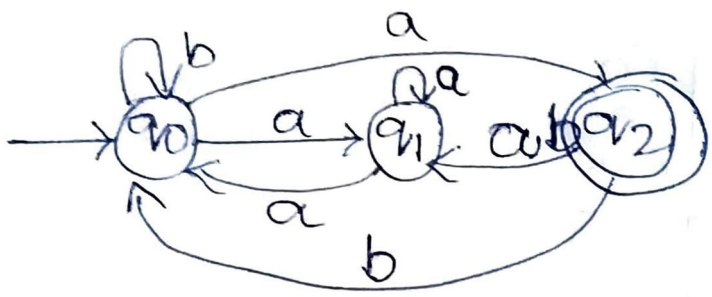
DFA:



Answers

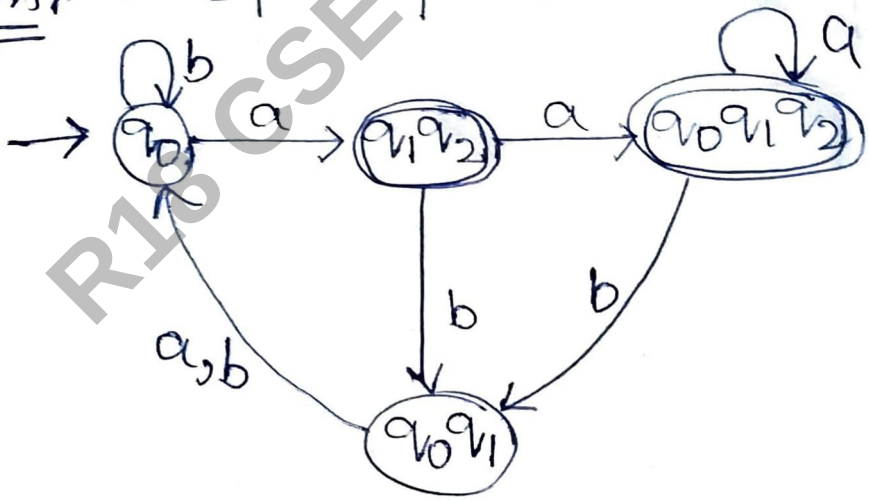
Convert NFA to DFA:-

1)



δ	a	b
→ q_0	$[q_1 q_2]$	q_0
$[q_1 q_2]$	$[q_0 q_1 q_2]$	$[q_0 q_1]$
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$	$[q_0 q_1]$
$[q_0 q_1]$	q_0	q_0

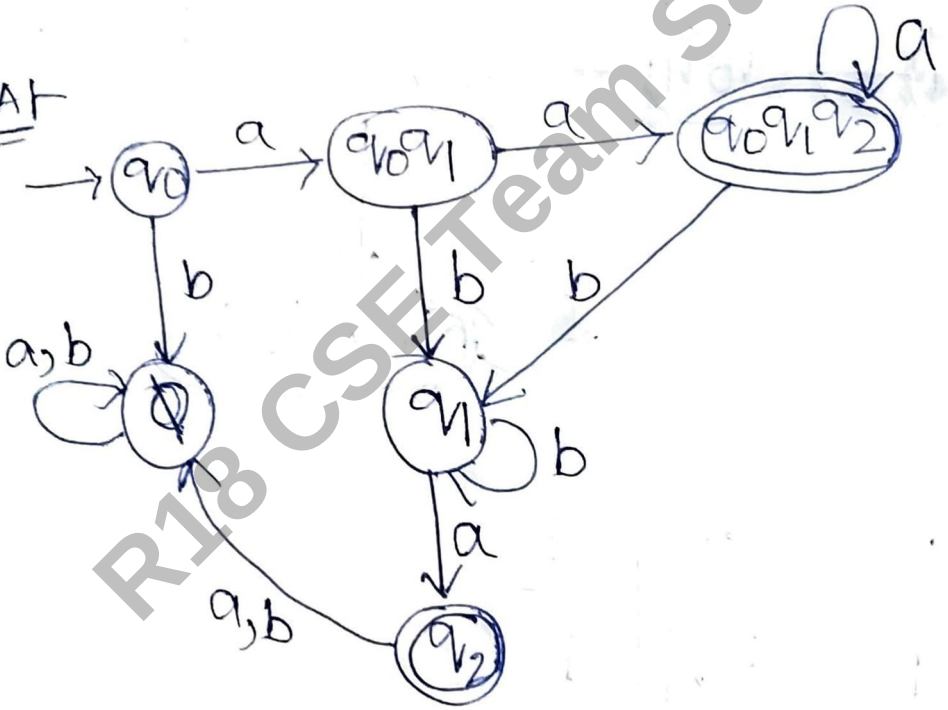
DFA:



2)

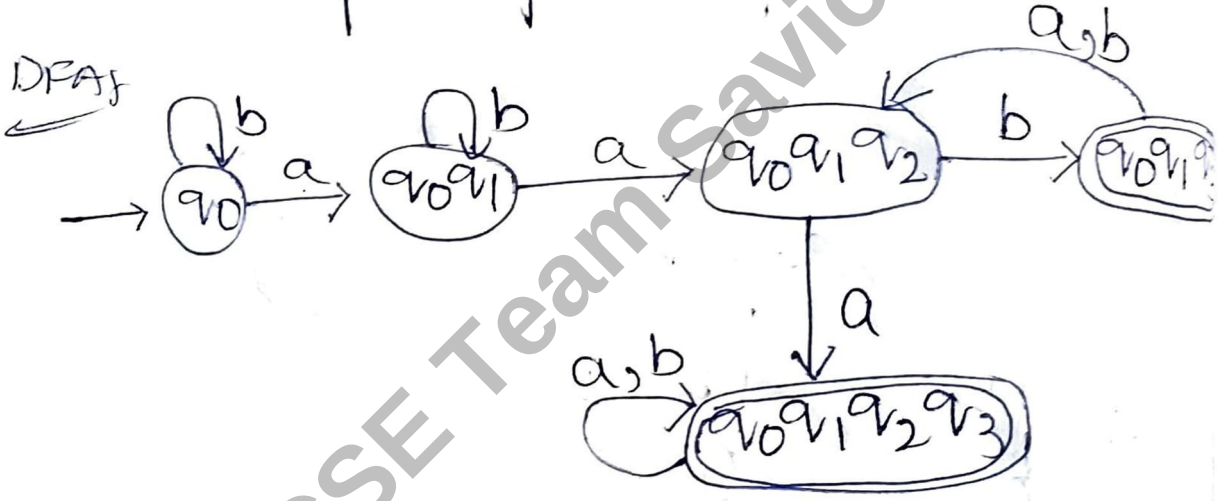
δ	a	b
$\rightarrow q_0$	$[q_0 q_1]$	ϕ
$[q_0 q_1]$	$[q_0 q_1 q_2]$	q_1
ϕ	ϕ	ϕ
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$	q_1
q_1	q_2	q_1
q_2	ϕ	ϕ

DFA:



3)

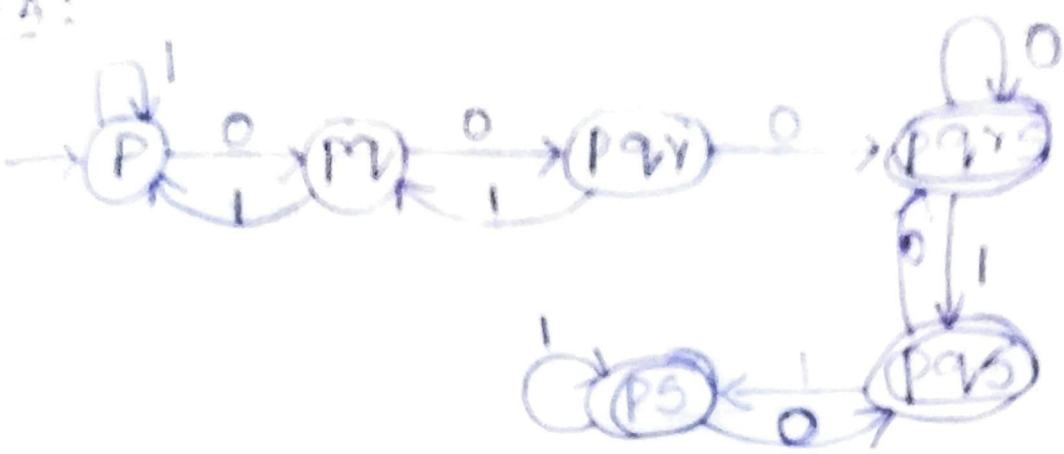
δ	a	b
$\rightarrow q_0$	$[q_0 q_1]$	q_0
$[q_0 q_1]$	$[q_0 q_1 q_2]$	$[q_0 q_1]$
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_3]$
$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_2 q_3]$
$[q_0 q_1 q_3]$	$[q_0 q_1 q_3]$	$[q_0 q_1 q_2]$



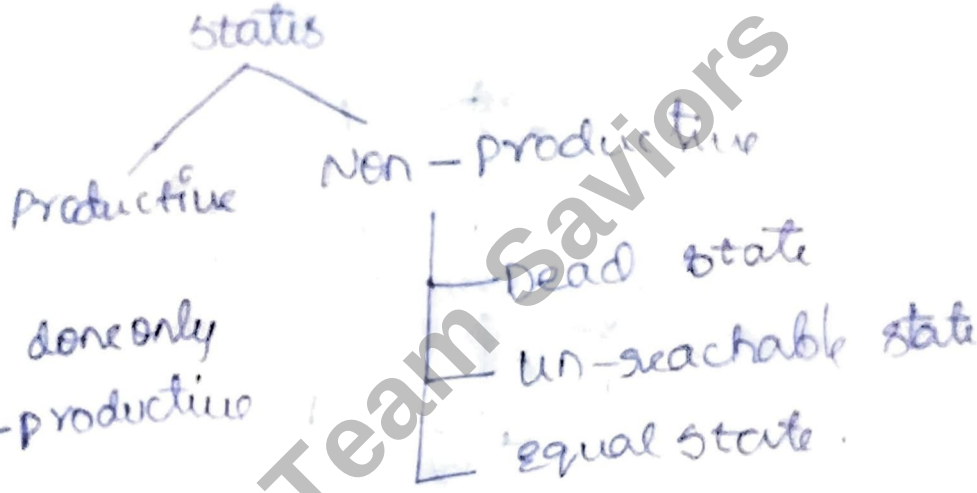
4)

δ	0	1
$\rightarrow p$	$[pq]$	p
$[pq]$	$[pqr]$	p
$[pqr]$	$[pqrs]$	$[pq]$
$[pqrs]$	$[pqrs]$	$[pqs]$
$[pqs]$	$[pqrs]$	$[ps]$
$[ps]$	$[pqs]$	$[ps]$

FA:



* Minimisation of Finite Automata:-



→ Reduction is done only on the non-productive states.

→ Minimisation is done for DFA.

→ Minimisation means reducing states in DFA.

Equivalence state:-

→ Two states q_1 & q_2 are equivalence denoted by $q_1 \cong q_2$, if both $\delta(q_1, w)$ and $\delta(q_2, w)$ are final states (or) both of them are non-final states for $w \in \Sigma^*$

$q_1 - w$

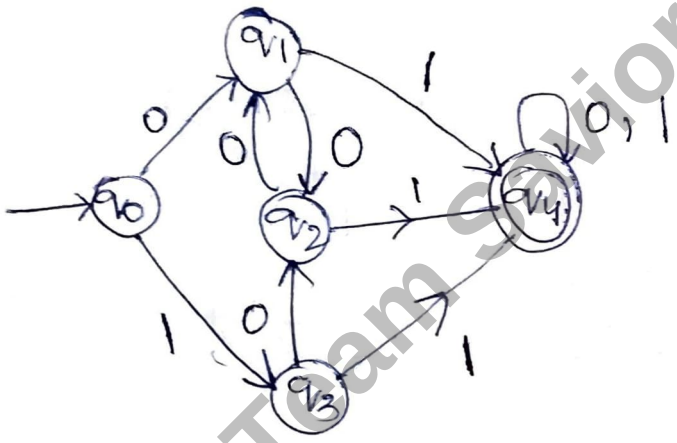
.

$q_2 - w$.

final states or non-final states

Partition method for minimum automata

⇒ calculate the minimum automata for given DFA.



Ans:

δ	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
(q_4)	q_4	q_4

Step-1 :- Dead state :- No dead state

Step-2 :- Unreachable :- NO

Step-3 :- equivalence state

0-equivalence Π^0 (Final & Non final)

$= \{q_4\}\ \{q_1 q_2 q_3\ q_0\}$

1-equivalence Π^1

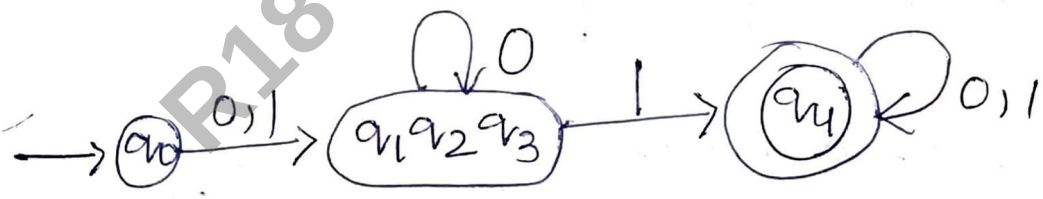
$= \{q_4\}\ \{q_0\}\ \{q_1 q_2 q_3\}$

DFA → $M = (Q, \Sigma, \delta, q_0, F)$

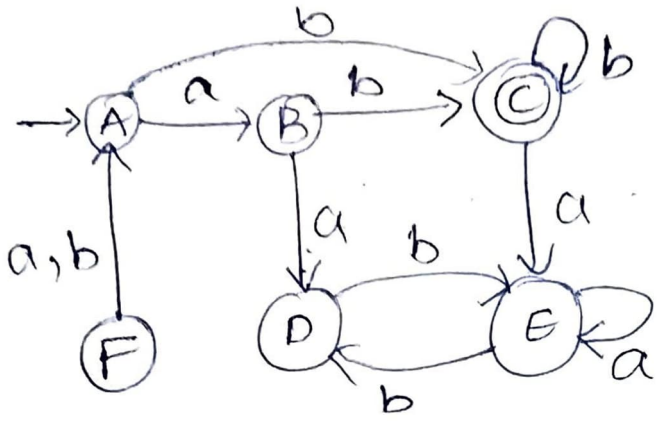
$F = q_4$

$q_0 = q_0$

δ	0	1
$\rightarrow q_0$	$[q_1 q_2 q_3]$	$[q_1 q_2 q_3]$
$[q_1 q_2 q_3]$	$[q_1 q_2 q_3]$	q_4
q_4	q_4	q_4



⇒ Minimize the DFA



Ans:

δ	a	b
A	B	C
B	D	C
C	E	C
D	E	C
E	E	D
F	A	A

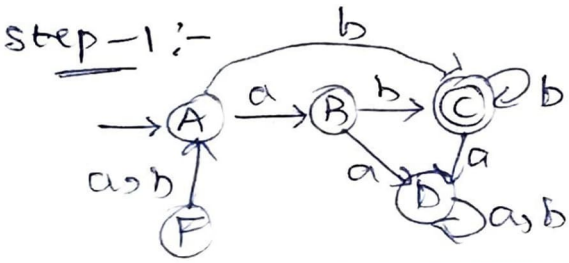
Step-1:- Dead state:

2 Dead states [D & E] {combine and write one}

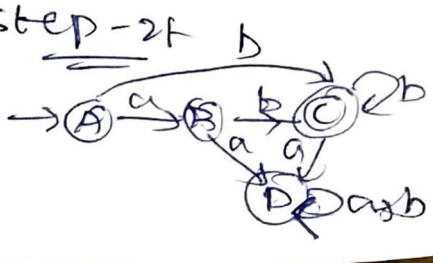
Step-2:- unreachable state:

F is unreachable so remove

step-1:-



step-2:-



Step-3: Equivalence states

δ	a	b
→A	B	C
B	D	C
C	D	C
D	D	D

0-equivalence π^0

$= \{C\}\ \{A\ B\ D\}$

1-equivalence π^1

$= \{C\}\ \{A\ B\}\ \{D\}$

2-equivalence π^2

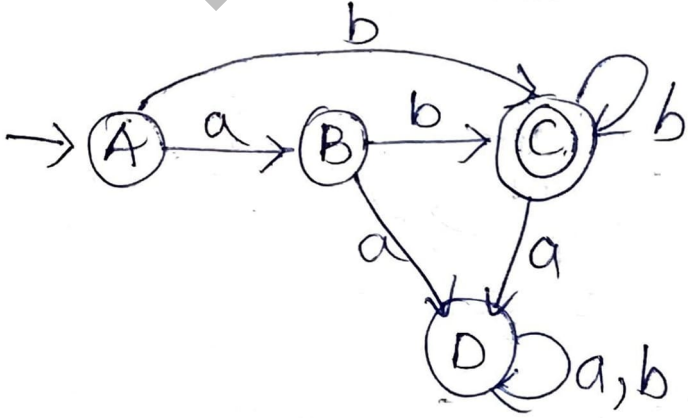
$= \{C\}\ \{A\}\ \{B\}\ \{D\}$

DFA $\rightarrow M = (Q, \Sigma, \delta, q_0, F)$

$F = C$

$q_0 = A$

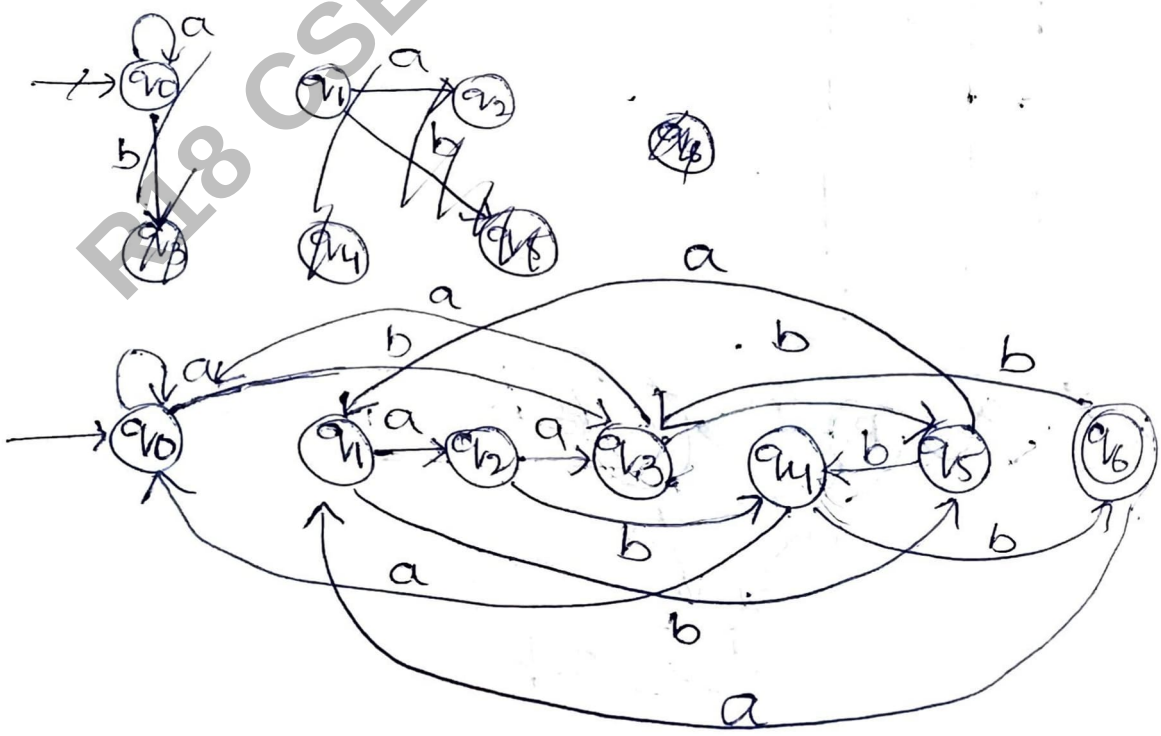
δ	a	b
→A	B	C
B	D	C
C	D	C
D	D	D



⇒ Construct minimum state automata equivalent to given automata M whose transition table

Q/Σ	a	b
→ q_0	q_0	q_3
q_1	q_2	q_5
q_2	q_3	q_4
q_3	q_0	q_5
q_4	q_0	q_6
q_5	q_1	q_4
(*) q_6	q_1	q_3

Ans:-



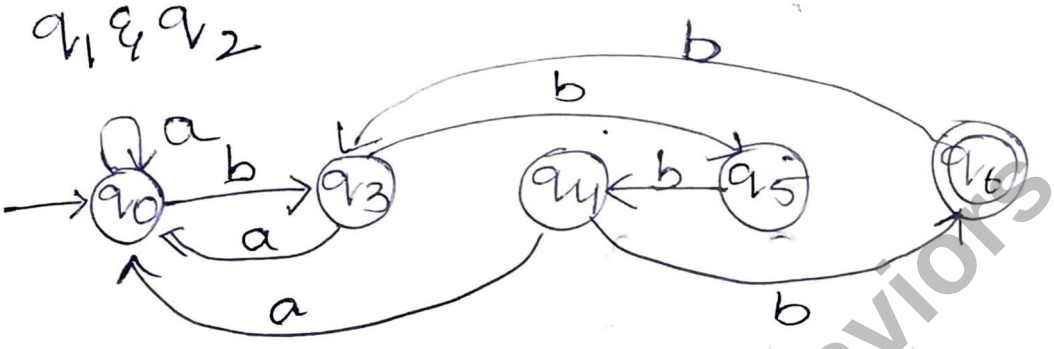
Step–1 :–

Dead statis :– NO dead statis

Step–2 :–

unreachable statis

q_1 & q_2



Step–3 :–

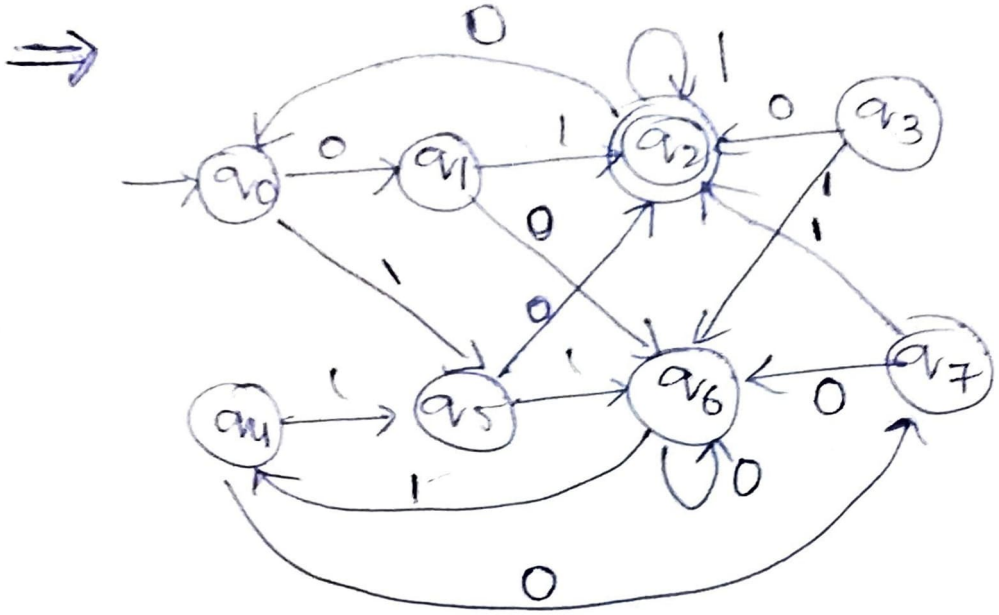
δ	a	b
→ q_0	q_0	q_3
q_3	q_0	q_5
q_4	ϕ	q_6
q_5	ϕ	q_4
(q_6)	ϕ	q_3

$\pi^0 = \{q_6\}\ \{q_0 q_3 q_4 q_5\}$

$\pi^1 = \{q_6\}\ \{q_0 q_3\}\ \{q_4 q_5\}$

$\pi^2 = \{q_6\}\ \{q_0 q_3\}\ \{q_4\}\ \{q_5\}$

δ	a	b
→ q_0	$[q_0 q_3]$	$[q_0 q_3]$
q_3	$[q_0 q_5]$	
$[q_0 q_3]$	$[q_0 q_3]$	

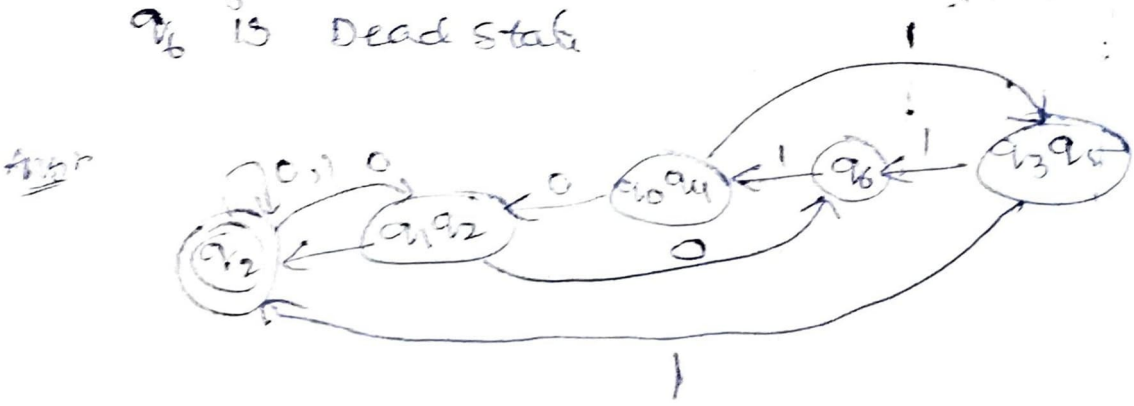


Ans:

δ	0	1
$\rightarrow q_0$	q_1	q_5
q_1	q_6	q_2
(q_2)	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

Step-(1):-

q_6 is Dead state



Equivalence of Finite State Machines :- (M, M')

q_0, P_0 are the .

step-1 :-

construct a comparison table with (n+1) columns where n is number of input symbols.

step-2 :-

comparision table starting with initial states of machine M and M'.

step-3 :-

Repeat the construction considering the pair in second subsequent columns which are not in first column. Row construction is repeated.

step-4 :- we need to check whether they are equivalent or not

case-1 :

If we reach x, y where

$x \in$ – final state

$y \in$ – non-final state ,

stop the construction of comparision table and conclude that two machines are not equal.

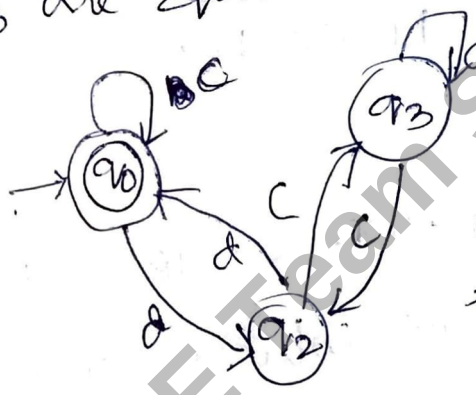
Case 2:

Terminate the table.

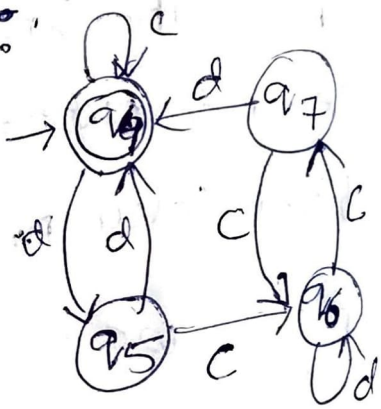
→ No new element appear in 2nd or subsequent column which are not in first column. conclude that these 2 are equivalent.

⇒ prove that the following finite state machines are equivalent.

M:



M':



Ans:

	c	d
$(q_0 q_4)$	$(q_0 q_4)$	$(q_2 q_5)$
$(q_2 q_5)$	$(q_3 q_6)$	$(q_0 q_4)$
$(q_3 q_6)$	$(q_2 q_7)$	$(q_3 q_6)$
$(q_2 q_7)$	$(q_3 q_6)$	$(q_0 q_4)$

⇒ Equivalent.

⇒ M =

	0	1
→A	B	D
B	A	C
C	D	B
(D)	C	A

M' =

	0	1
→P	R	R
Q	R	P
(R)	P	Q

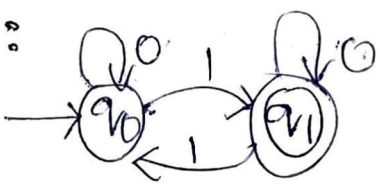
Ans:

	0	1
(DR)	(CP)	(AQ)
(CP)	(DR)	(BR)
(AQ)	(BR)	(DP)
(BR)	(AP)	(CQ)
(DP)	(CR)	(AR)
(AP)	(BR)	(DR)
(CQ)	(DR)	(BP)

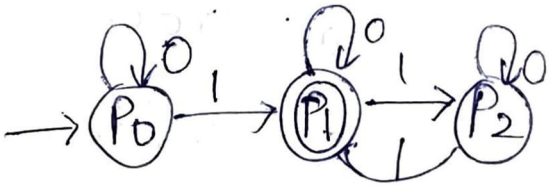
⇒ Not Equivalence

	0	1
(AP)	(BR)	(DR)
~~(BR)~~	~~(BP)~~	~~&&~~

⇒ M:

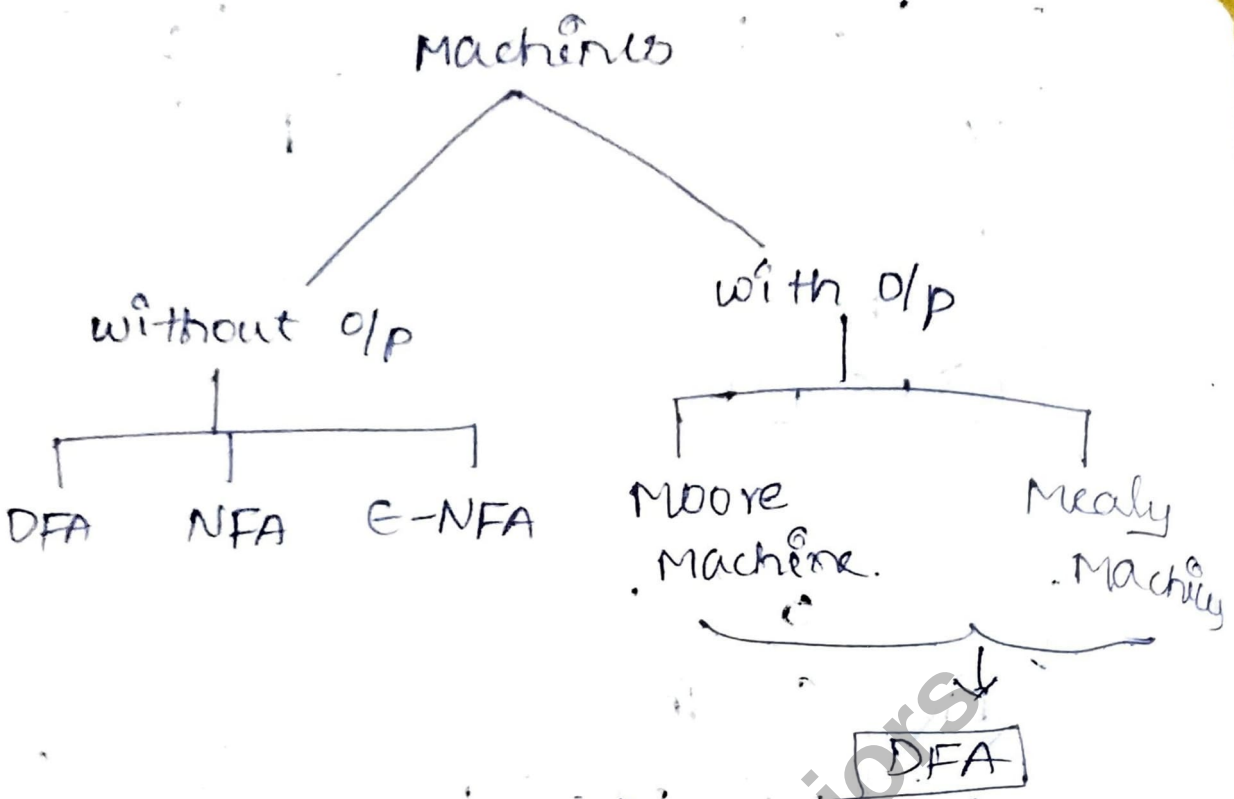


M':



	0	1
q_0P_0	q_0P_0	q_1P_1
q_1P_1	q_1P_1	q_0P_2
q_0P_2	q_0P_2	q_1P_1

⇒ Equivalence



Moore and Mealy machines:-
(with output)

→ Defines with 6-tuples

$M = (Q, \Sigma, \delta, \Delta, \lambda, q_0)$

where Q = A finite set of states

Σ = finite set of input symbols

δ = Transition function / mapping

Δ = finite set of output symbols

λ = output function

q_0 = Initial state $q_0 \in Q$

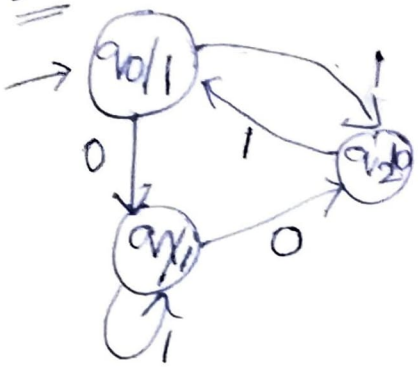
→ Moore and Mealy Machines are DFA.

Moore Machine:

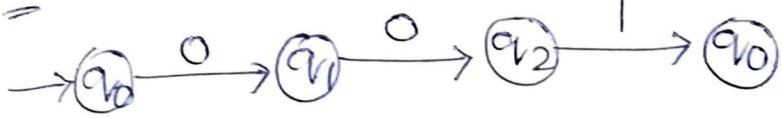
output function

1) $\lambda : Q \rightarrow \Delta$

2) Ex:



Ex:- 001



output: 1 1 0 1

$\therefore$ If $\Sigma = n$ then $\Delta = n+1$

3) → O/p associated with state

4) O/p depends on current state.

5) The O/p string one character more than input string (n+1)

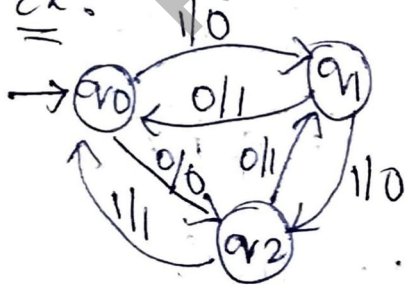
6) Moore needs more no of states than mealy.

Mealy Machine:-

output function

1) $\lambda : Q \times \Sigma \rightarrow \Delta$

2) Ex:-



Ex: 001

$q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_1 \xrightarrow{1} q_2$

0 1 0

If $\Sigma = n$ then $\Delta = n$

3) O/p depends on current state & input symbol.

4) Both i/p & o/p have equal no of characters

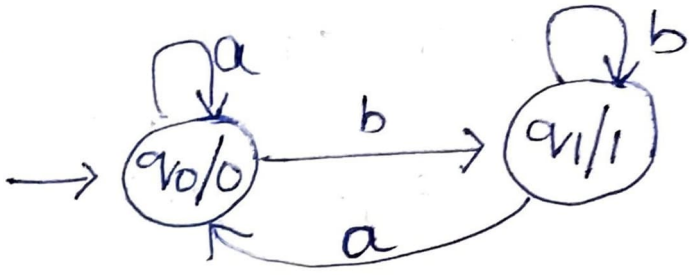
5) mealy needs relatively less no. of states.

⇒ problems on Moore Machines:

⇒ design Moore Machine to count no of b's in a given input string consisting with a's & b's.

$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$

Ans:-



Ex:- aabab

$q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_0 \xrightarrow{b} q_1$

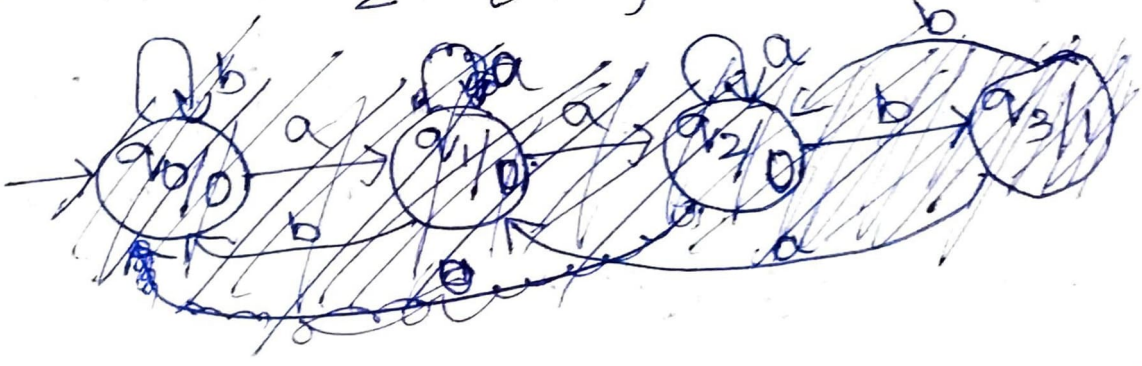
0 0 0 1 0 1

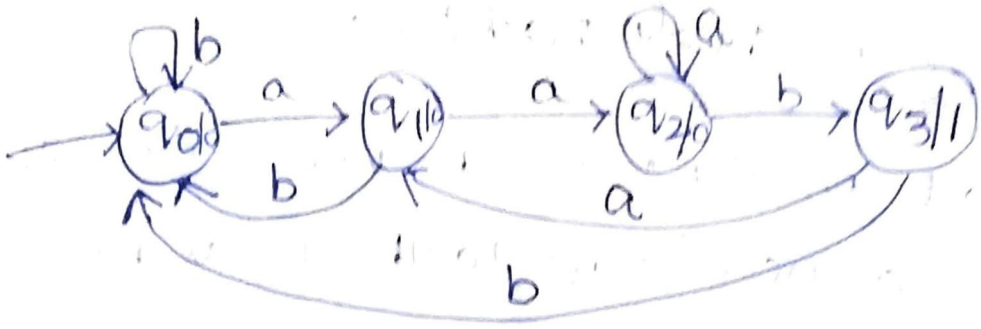
$= 2$

$\therefore$ no of b's $= 2$

⇒ Design Moore Machines to know how many the substring 'aab' occurs in a long inputstring consistings of a's & b's.

Ans:- $\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



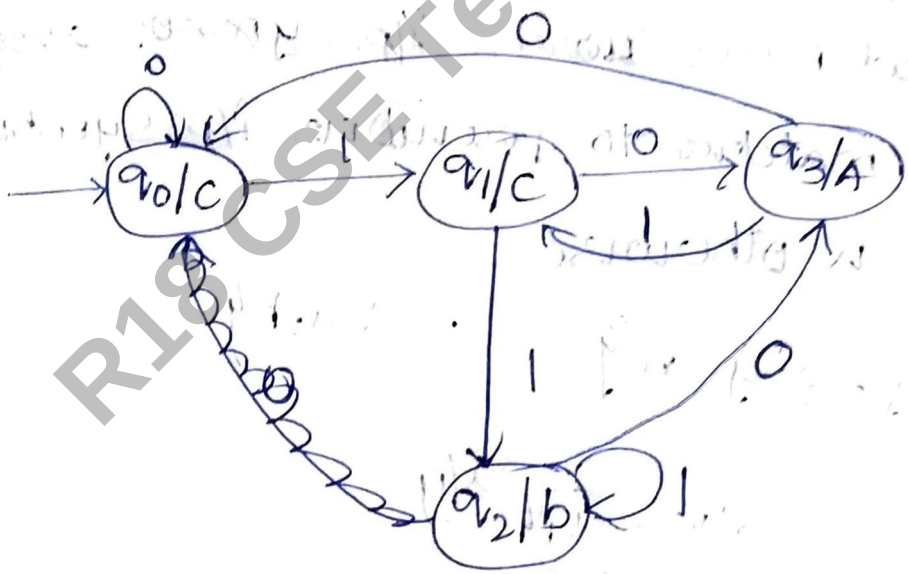


⇒ Construct Moore Machine $\Sigma = \{0, 1\}$ and produce i) 'A' as output if inputstring ends with '10'.

ii) 'B' as output if inputstring ends with '11'.

iii) otherwise — 'C'

Ans:-

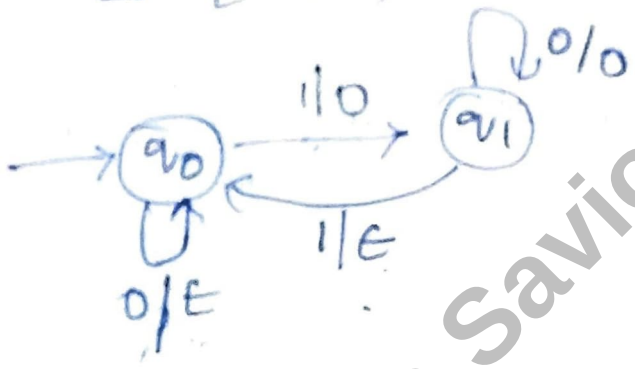


⇒ Problems on mealy machine :-

⇒ Construct mealy machine which can give o/p as ∈ or 0 according to the total no. of encountered even or odd.

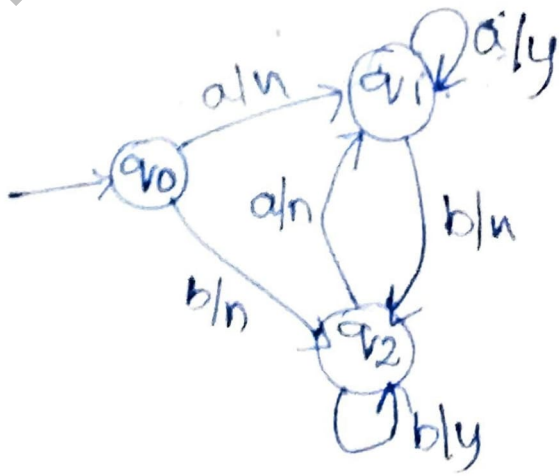
Ans: $\Sigma = \{0, 1\}$

$\Delta = \{\epsilon, 0\}$



⇒ Design mealy machine which yields o/p as y whenever current i/p symbol matches to previous i/p symbol and display n otherwise.

Ans: $\Delta = \{y, n\}$ $\Sigma = \{a, b\}$

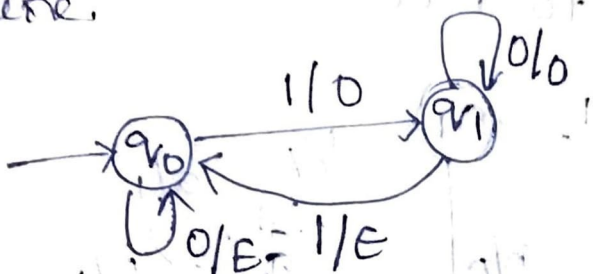


II :- Convert Mealy to Moore Machine :-

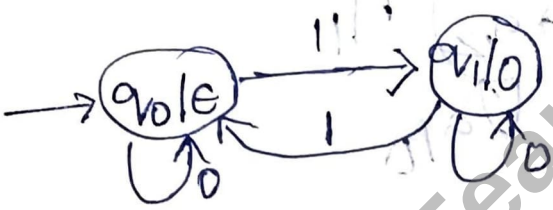
$$\delta'((q,b),a) = [\delta(q,a), \lambda(q,a)]$$

$$\lambda'([q,b]) = b$$

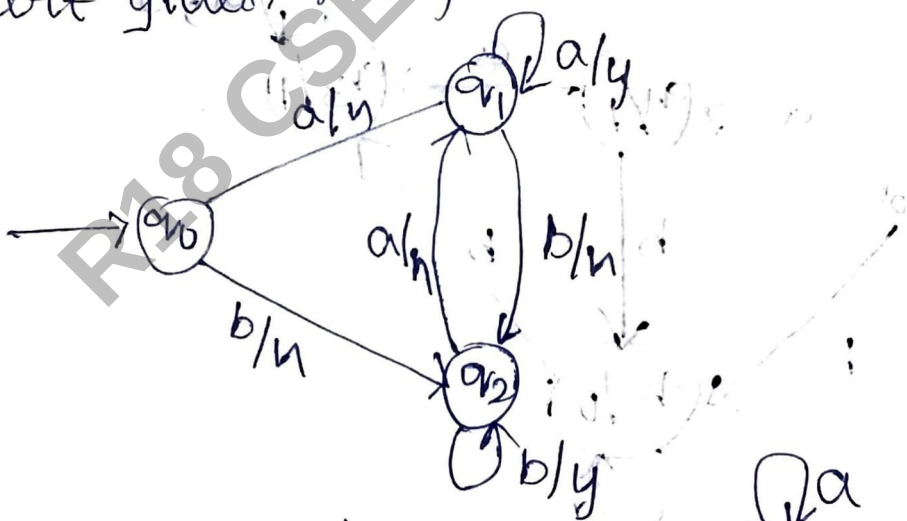
⇒ Convert given Mealy Machine to Moore Machine.



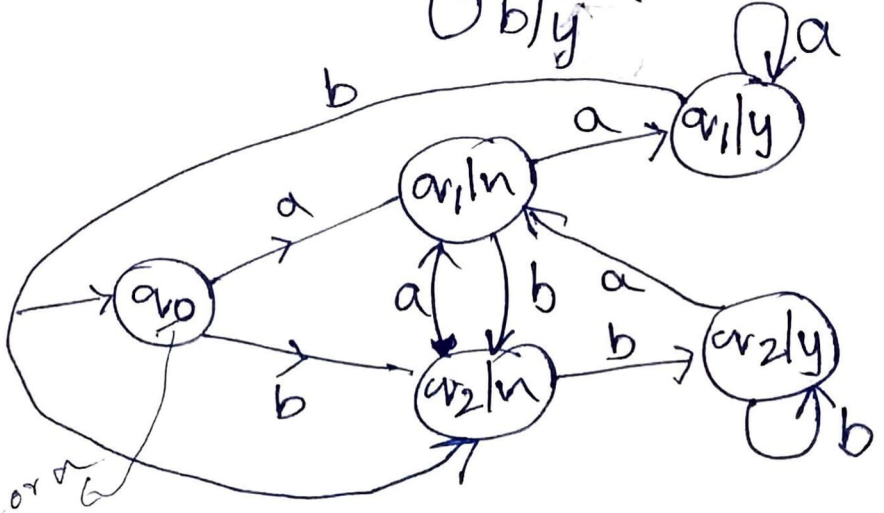
Ans :-



⇒ Convert given Mealy Machine to Moore Machine



Ans:



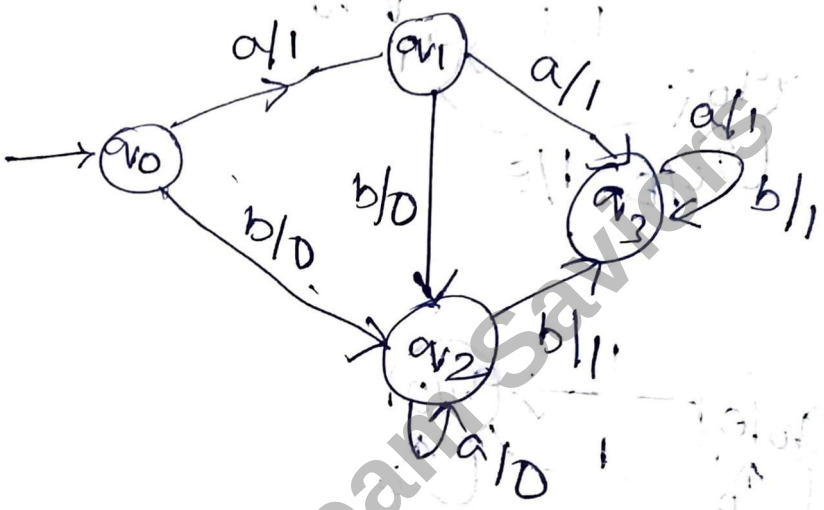
NOTE :-

→ while converting mealy to moore the no of states are $m \times n$ where

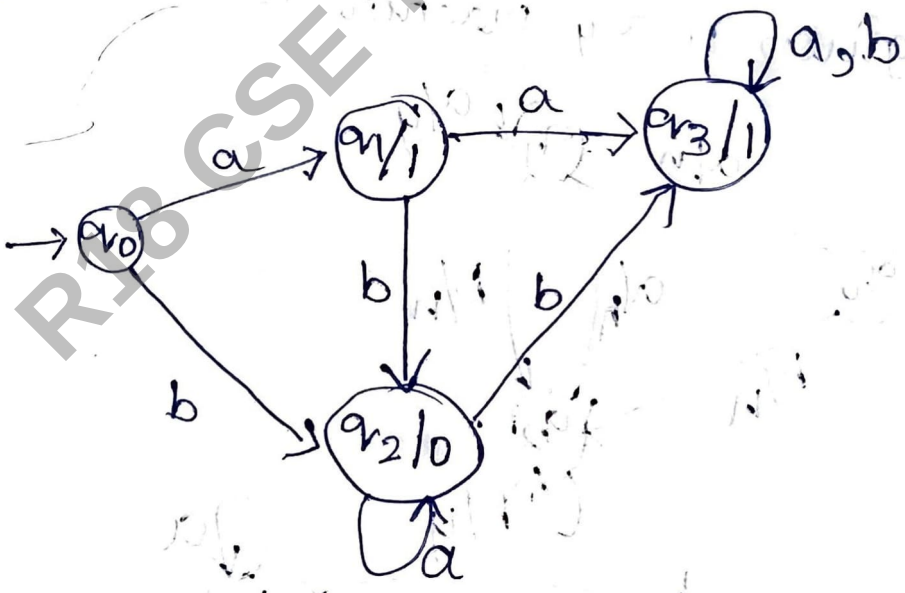
m = no of states of mealy

n = no of inputs of mealy

⇒ convert mealy to moore machine;



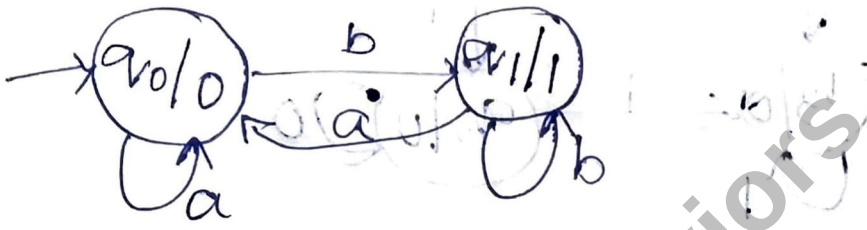
Ans:



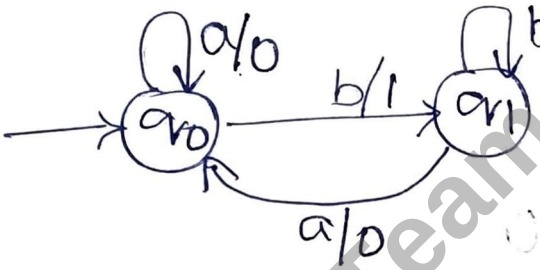
★ Convert Moore to Mealy Machine :-

$$\boxed{\lambda'(q, x) = \lambda(\delta(q, x))}$$

⇒ convert the given moore machine to equivalent mealy machine.



Ans :-

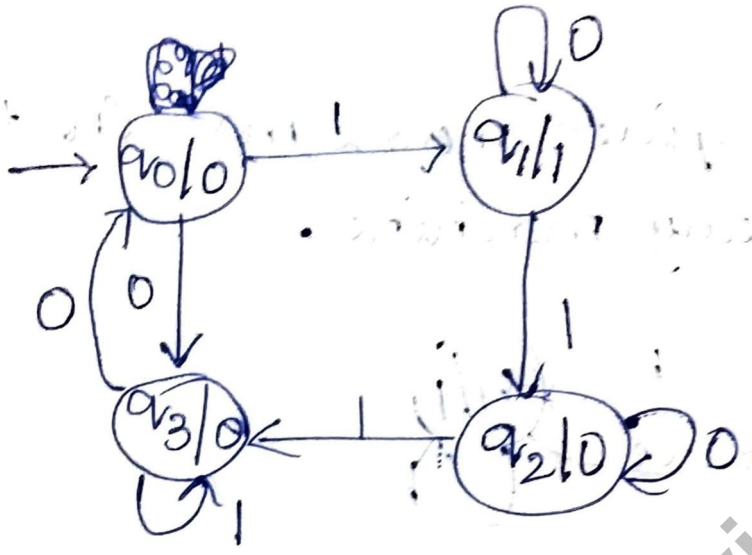


⇒ Convert given moore machine to mealy machine.

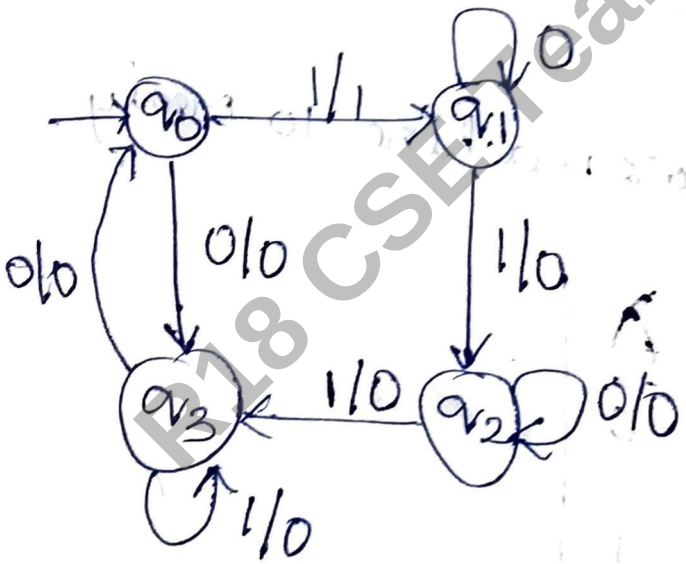
	0	1	λ
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Ans:

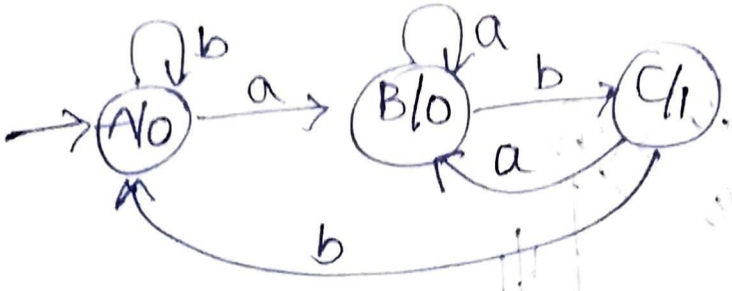
Moore – Machine:



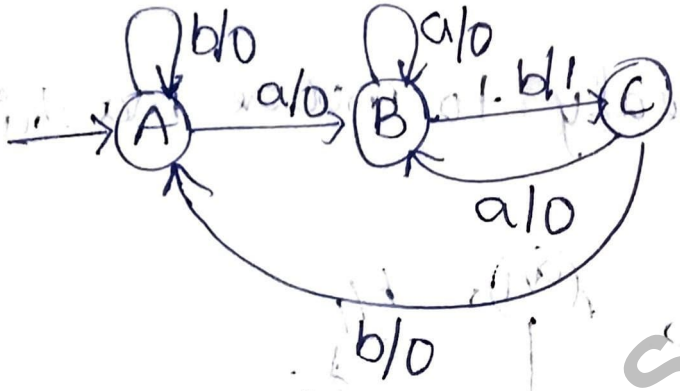
Mealy – Machine:



⇒ Convert given moore to mealy.



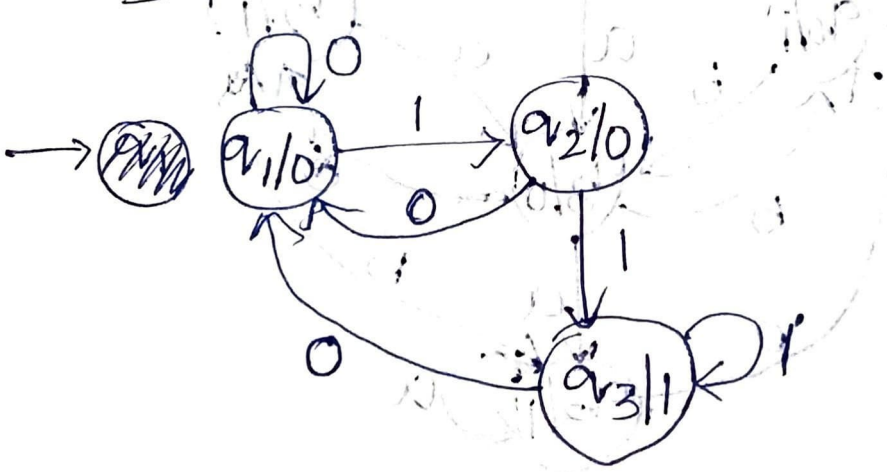
Ans:-



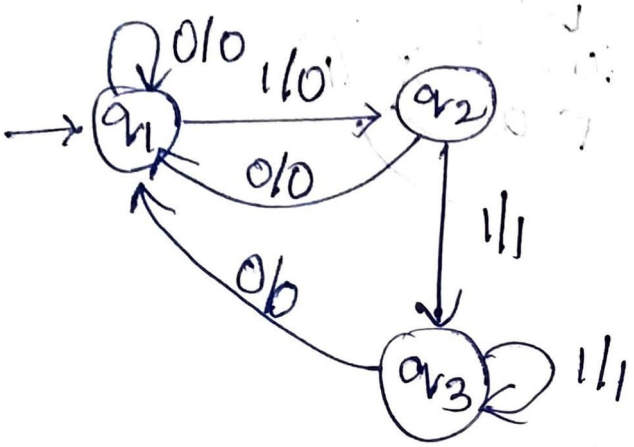
⇒ Convert moore to mealy

	0	1	
$\rightarrow q_1$	q_1	q_2	0
q_2	q_1	q_3	0
q_3	q_1	q_3	1

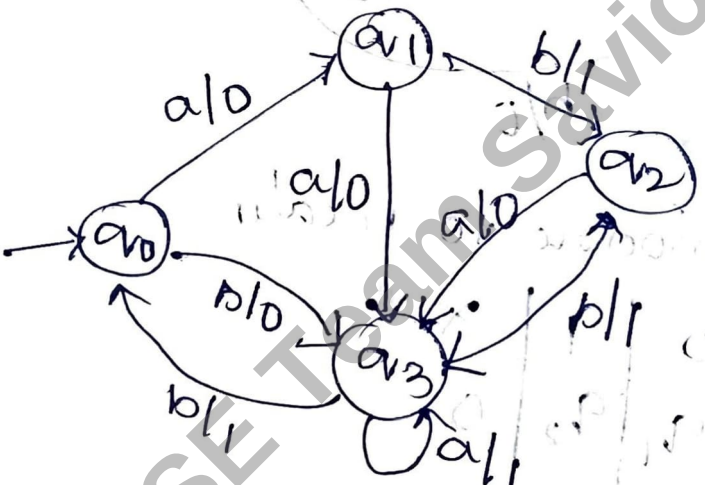
Ans:- moore-machine



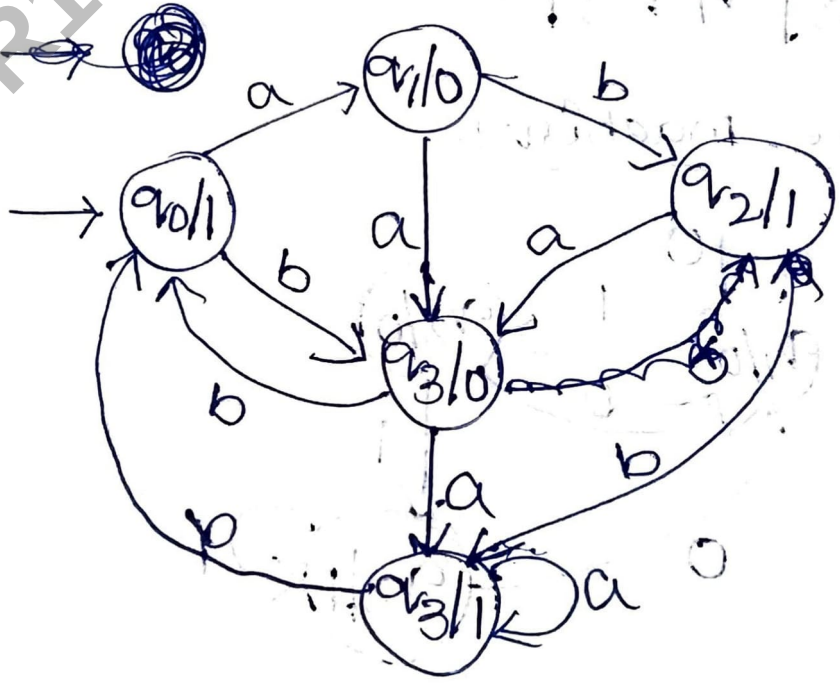
Mealy — machine:



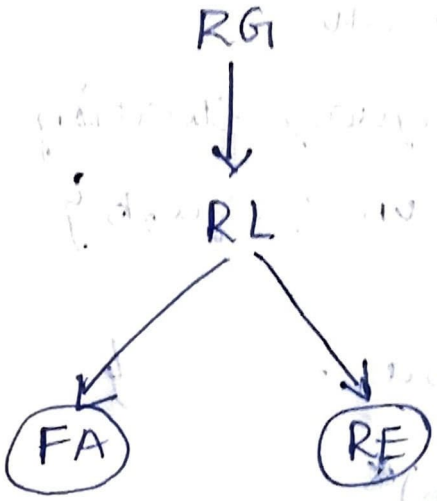
⇒ convert mealy to moore machine.



Ans:



Regular Expressions:-



→ A Regular Expression is a notation to represents certain sets of strings in an alphabets with algebric operators.

→ Here, operators are

1) * = klean closure
2) + = positive closure
3) U or + union
4) Concatination.

→ The alphabets are

Σ is 1) ϕ – Empty { }
2) ∈ – nullstring
3) a ∈ Σ

Problems on Language to RE :- (1)

→ r is regular Expression :

⇒ Write RE for a language starting with,

1. starting ab on $\Sigma = \{a, b\}$

↓

$L = \{ab, aba, \ldots\}$

Ans = $ab(a+b)^*$

2. Write RE starting with bba

Ans: $bba(a+b)^*$

3. Ends with abb

Ans: $(a+b)^*abb$

4. containing substring aab

Ans: $(a+b)^*aab(a+b)^*$

5. Start and end with a.

Ans:- $a + a(a+b)^*a$

6. start and end with same symbol

Ans:- $a + a(a+b)^*a + b + b(a+b)^*b$

7. start and end with different symbols.

Ans: $a(a+b)^* b + b(a+b)^* a$

8. $|w| = 3$ [Length = 3]

Ans:- $(a+b)(a+b)(a+b) = (a+b)^3$

9. $|w| \geq 3$

Ans:- $(a+b)^3 (a+b)^*$

10. $|w| \leq 3$

Ans:- $\epsilon + (a+b) + (a+b)^2 + (a+b)^3$

11. $L = \{w \in \Sigma^* / w$ has atleast one pair 0's

Ans:- $(0+1)^* 00 (0+1)^*$

12. RE that denotes all words with atleast 2 0's.

Ans: $(0+1)^* 0 (0+1)^* 0 (0+1)^*$

13. $L = \{w /$ Every odd position w is 1
even position w is 0

Ans:-

$(10)^* + (10)^* 1$

14. Write RE for a set of strings over {0,1} having even number of 0's & odd number of 1's.

Ans:- $1(11)^*0(00)^*$

15. Write RE over {0,1} in which the number of occurances of 0 is divisible by 3.

Ans:-

16. 10th symbol from right end is 1

Ans:-

17. String contains odd length

Ans:-

Manipulations of Regular Expression :-

Identity Rules :-

I_1 : $\phi + r = r$

I_2 : $\phi r = r\phi = \phi$

I_3 : $\epsilon r = r\epsilon = r$

I_4 : $\epsilon^* = \epsilon$

I_5 : $r + r = r$

I_6 : $r^* r^* = r^*$

I_7 : $r r^* = r^* r$

I_8 : $(r^*)^* = r^*$

I_9 : $\epsilon + r r^* = r^* = r r^* + \epsilon$

I_{10} : $(pq)^* p = p(qp)^*$

I_{11} : $(p+q)^* = (p^* q^*)^*$

$= (p^* + q^*)^*$

I_{12} : $(p+q)r = pr + qr$

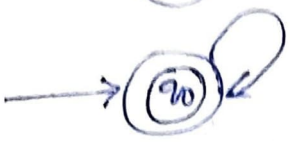
$r(p+q) = rp + rq$

Problems:-

Convert RE to FA

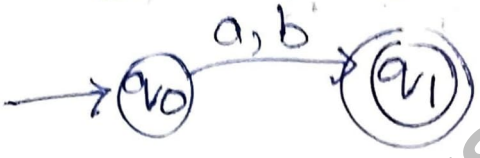
1) $r = \phi$

2) $r = \epsilon$



3) $r = a$

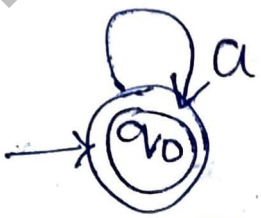
4) $r = a+b$



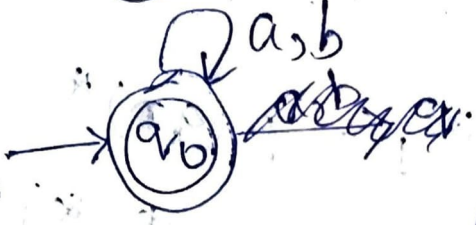
5) $r = ab$

(or)

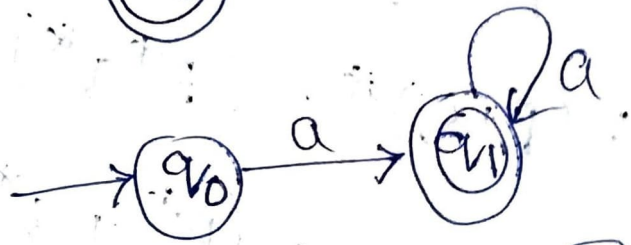
6) $r = a^*$



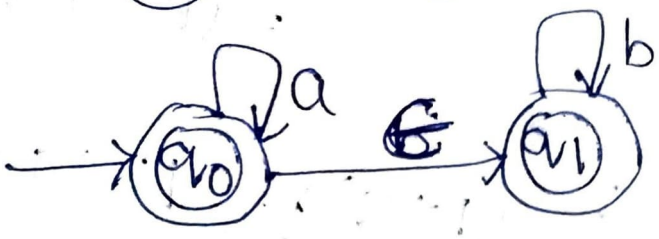
7) $r = (a+b)^*$



8) $r = a^+$

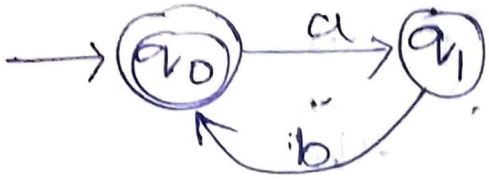


9) $r = a^* b^*$

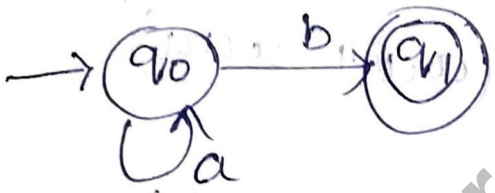


10) $r = (ab)^*$

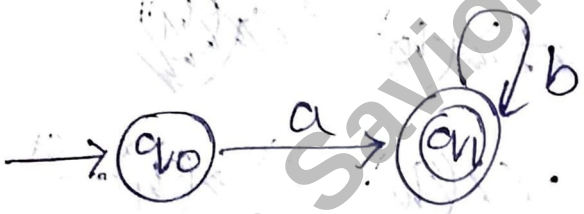
(or)



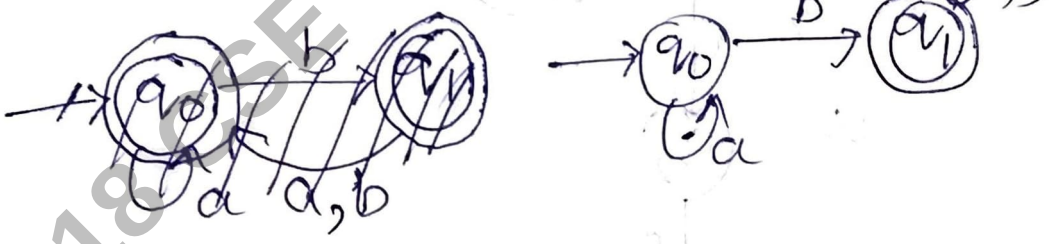
11) $r = a^* b$



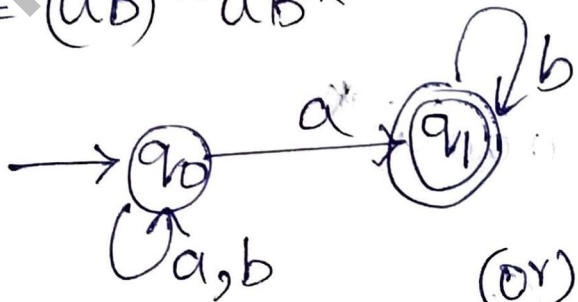
12) $r = a b^*$



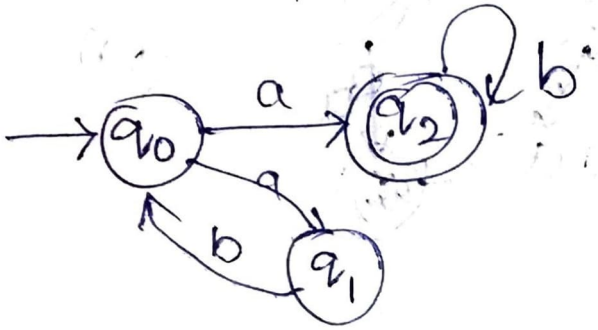
13) $r = a^* b (a+b)^*$



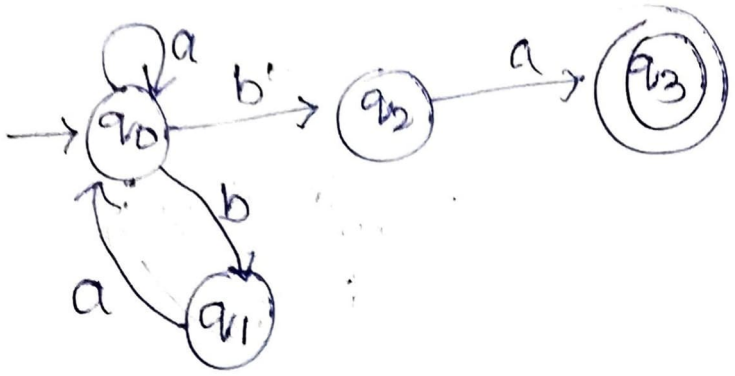
14) $r = (ab)^* ab^*$



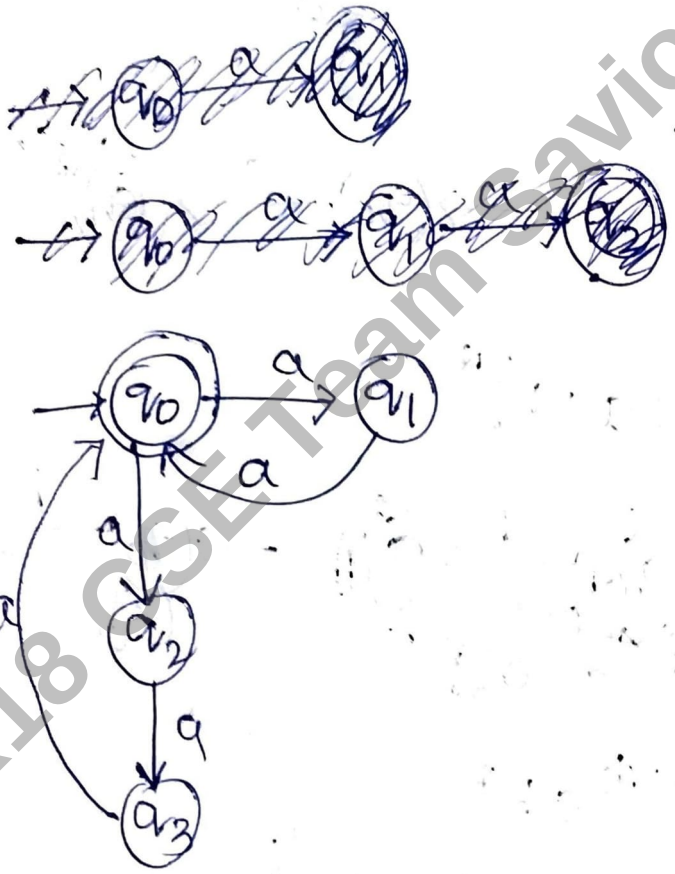
(or)



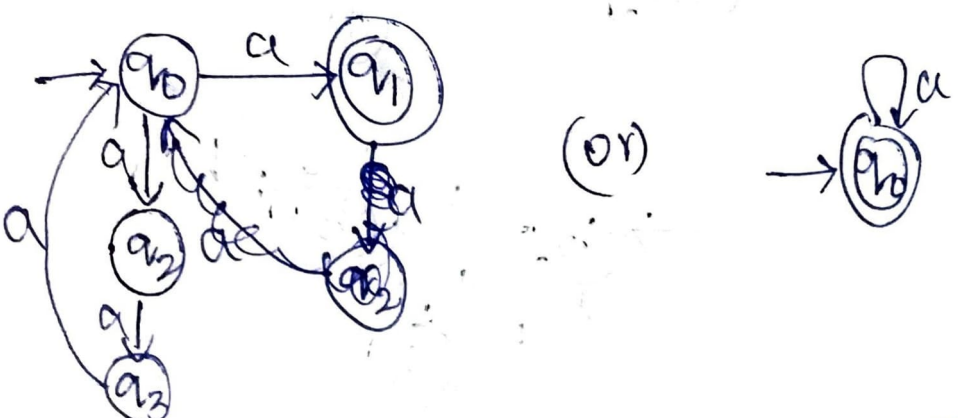
15) $r = (a+ba)^* ba$



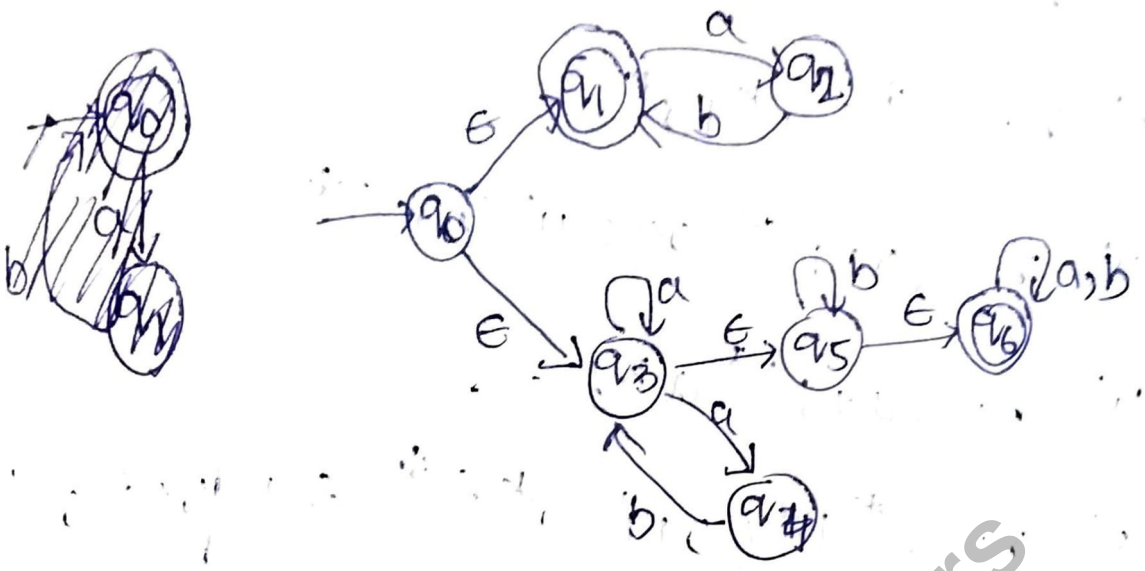
16) $r = (aa + aaa)^*$



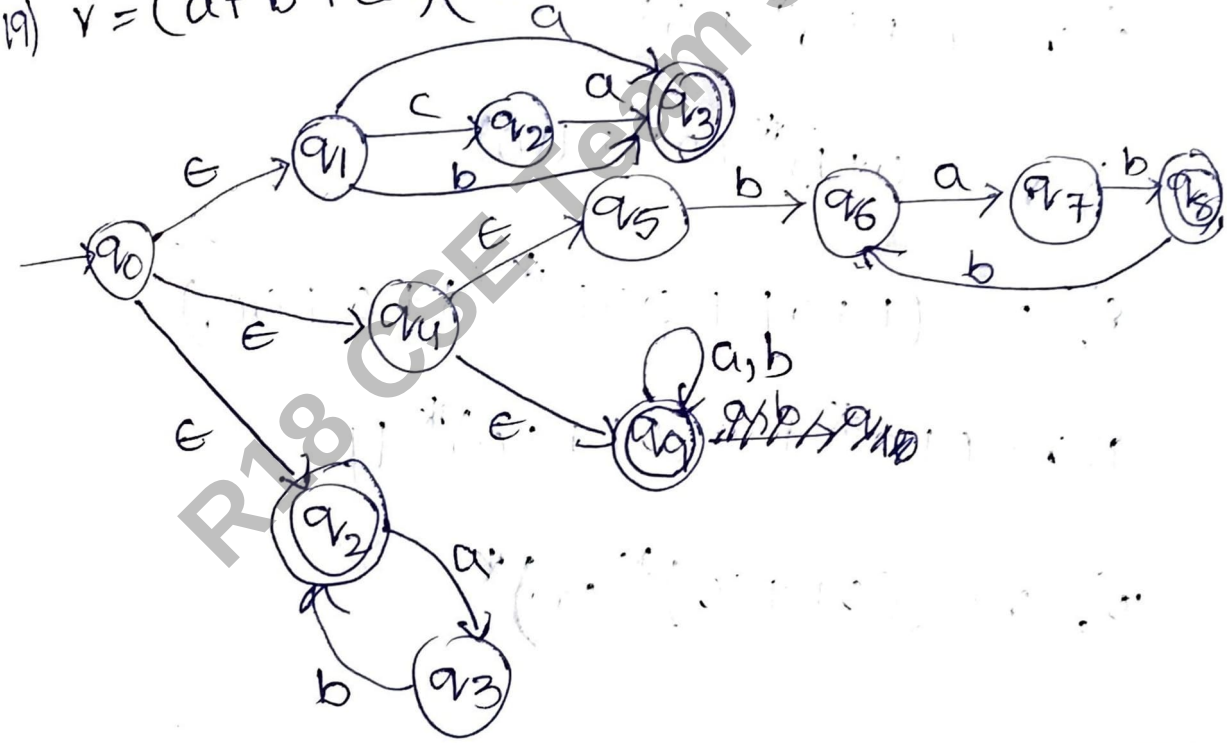
17) $r = (a + aaaa)^*$

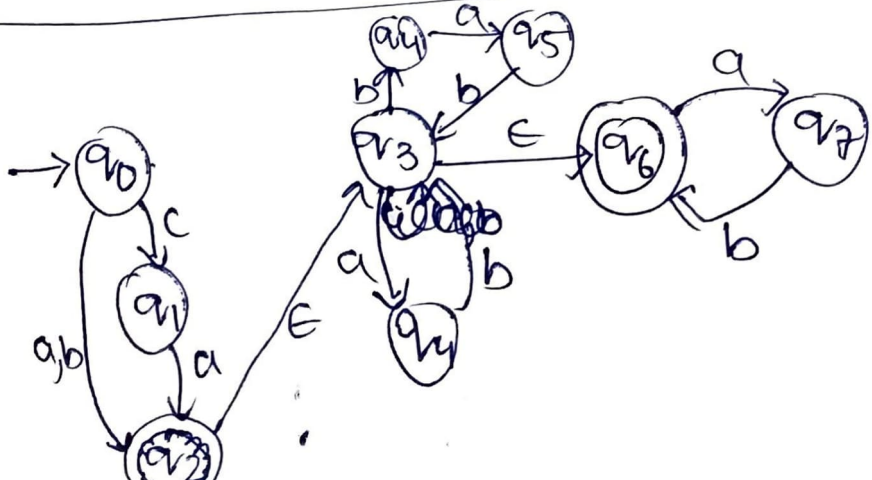


18) $r = (ab)^* + (a+ab)^* b^* (a+b)^*$



19) $r = (a+b+ca)((bab)+(a+b))^* (ab)^*$





Problems:

1) $r = (10) + (0 + 11)0^*1$

2) $r = 0^*10^*10^*$

3) $r = (0+1)^*(00+11)(0+1)^*$

4) $r = (a+b)^*abb$

5) $r = 0^*1((0+1)0^*1)^*(\epsilon + (0+1)(00)^* + 0(00)^*$

6) $r = 0 + 10^* + 01^*$

7) $r = (0+1)^*(01+110)$

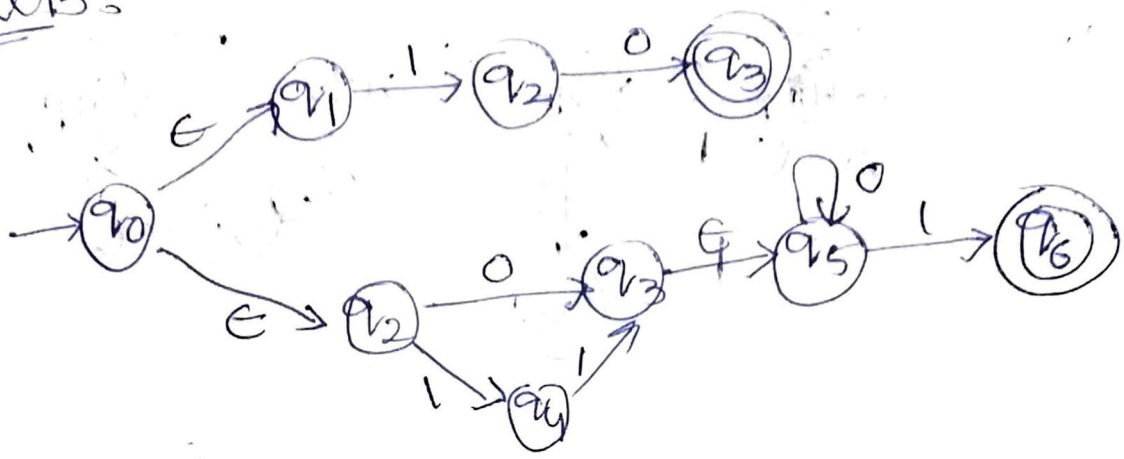
8) $r = (1+01+001)^*(1+0+00)$

9) $r = 010^* + 0(01+10)^*11$

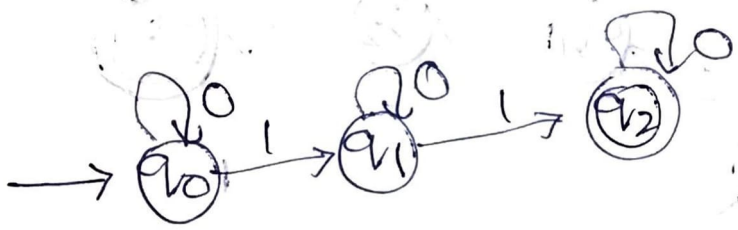
10) $r = ((01+10)^*00)^*$

Answers:-

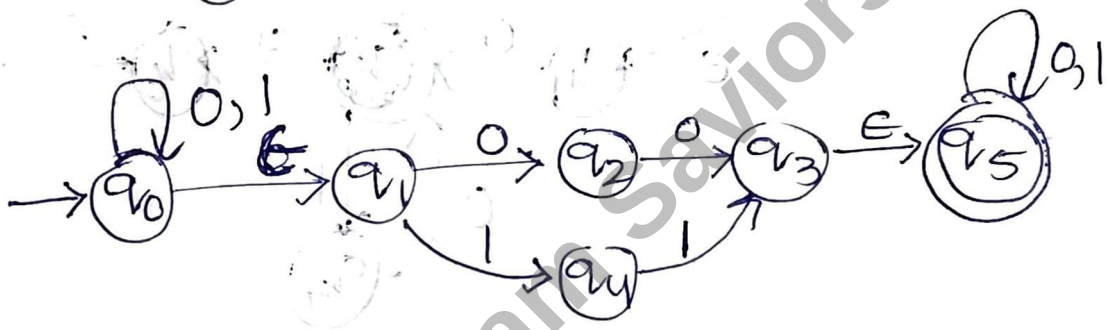
1)



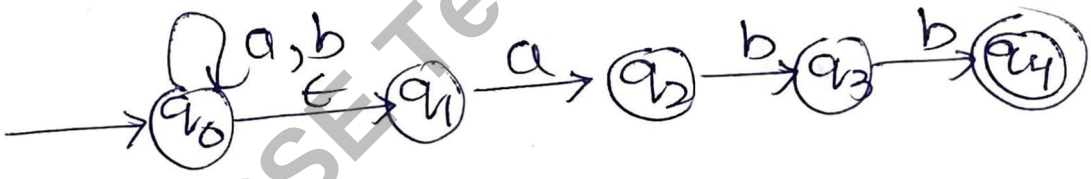
2)



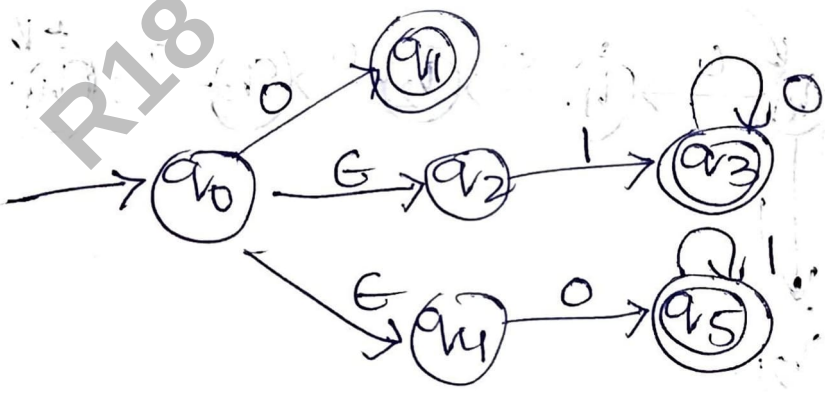
3)



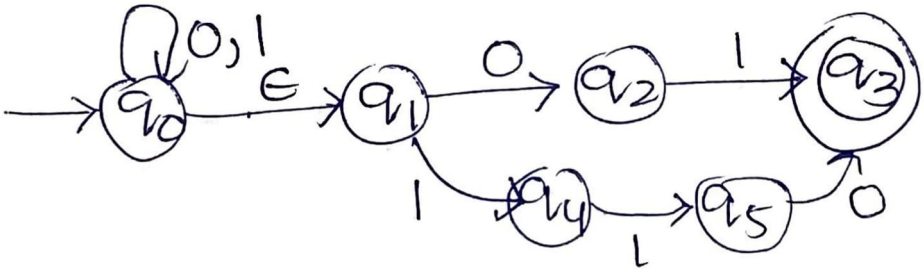
4)



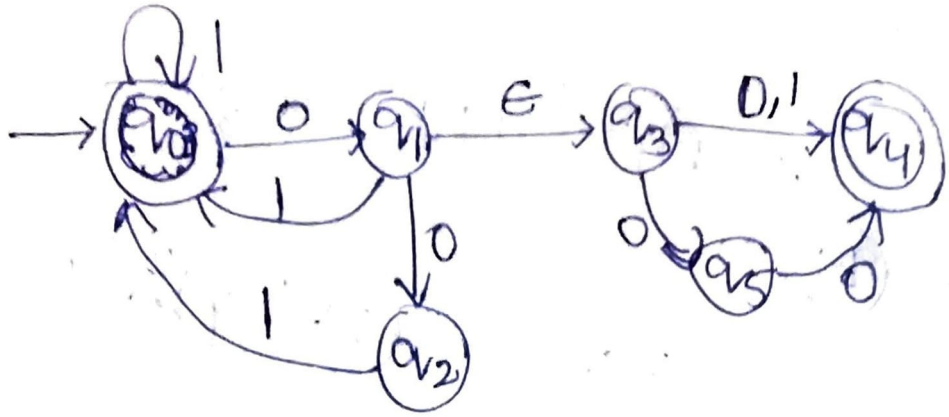
6)



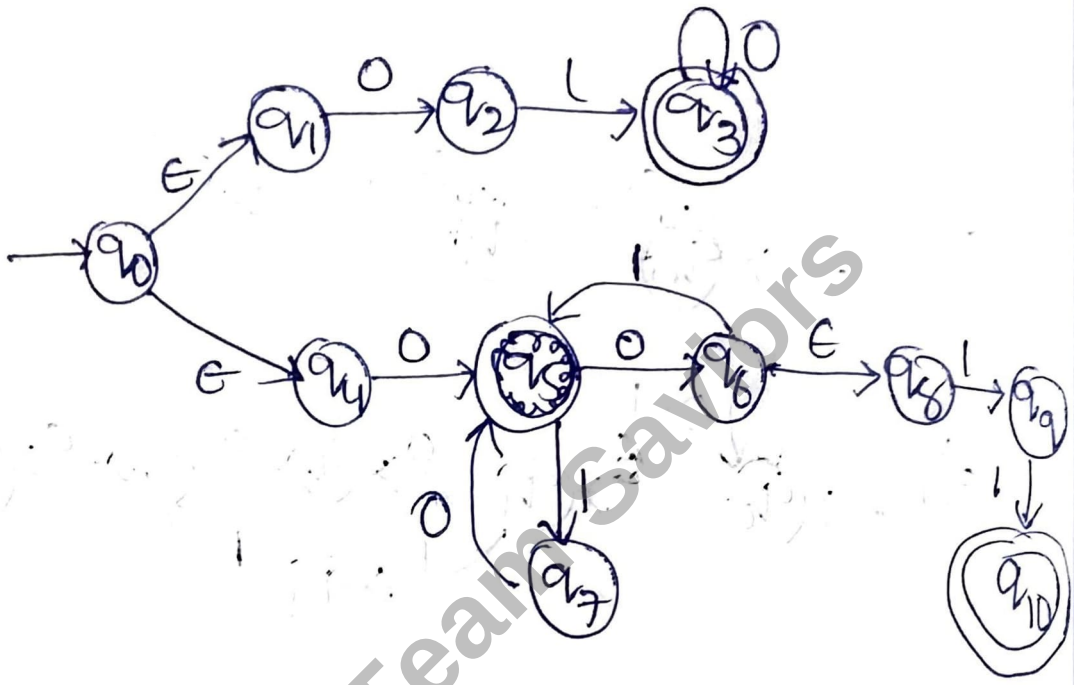
7)



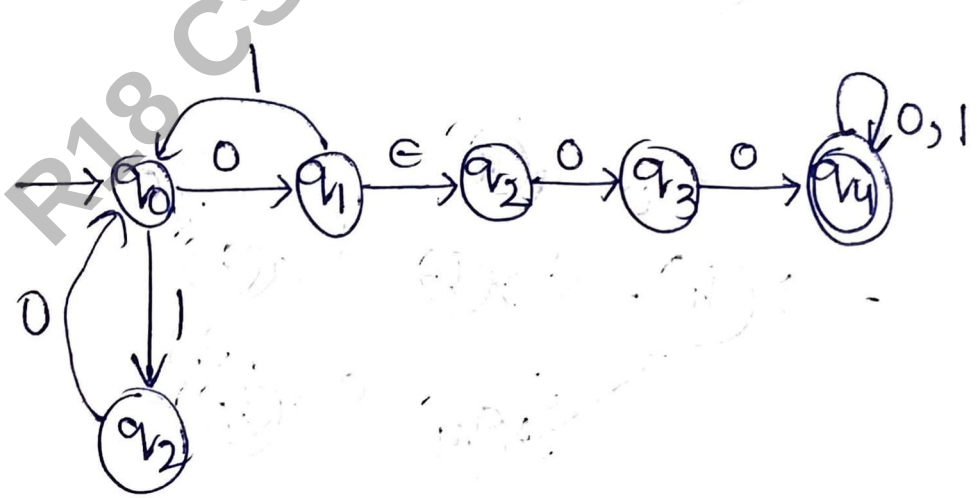
(8)



(9)

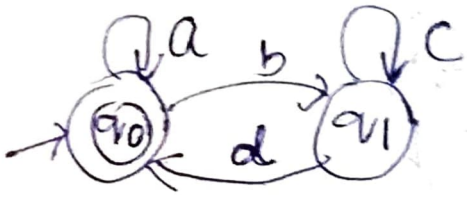


(10)



Problems on FA → RE :-

1)

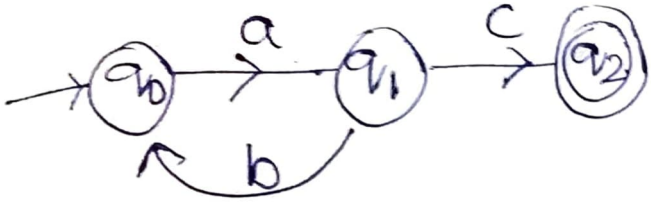


$\Rightarrow$ ~~$(a^* b c^* d)^*$~~

$a^*(bc^*d)^*$

$(a + bc^*d)^*$

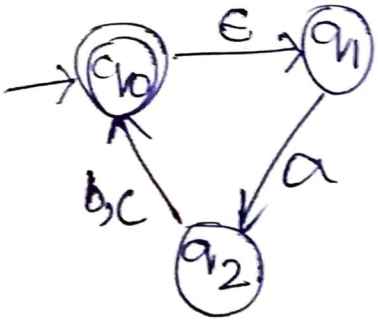
2)



$\Rightarrow$ ~~$(a(ba)^*c)^*$~~

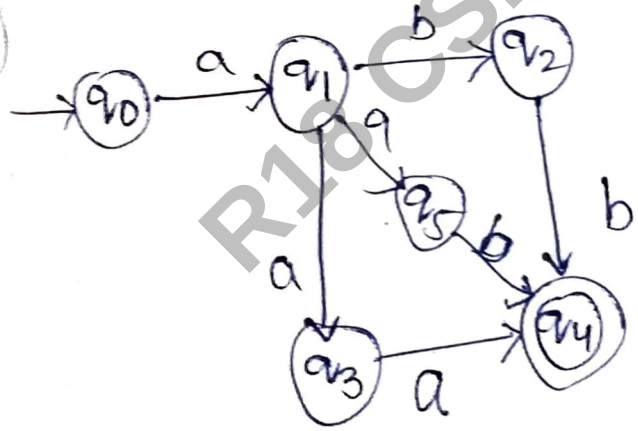
$(a(ba)^*c$

3)



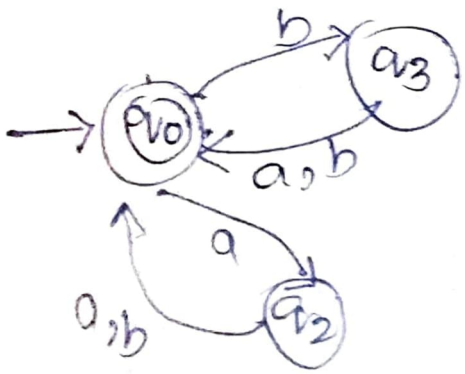
$\Rightarrow (\epsilon a(b+c))^*$

4)



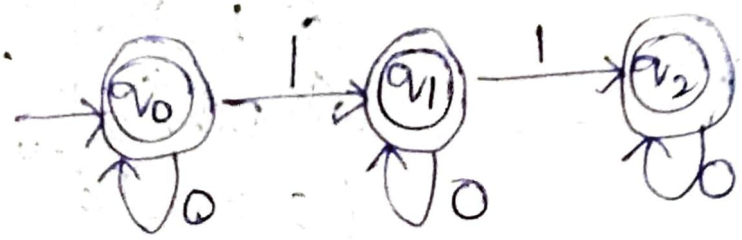
$\Rightarrow a(ab + aa + bb)$

5)



$\Rightarrow (b(a+b) + a(a+b))^*$

6)



$\Rightarrow 0^*10^*10^*$

UNIT – III

Regular Grammer

→ set of production:

→ 4 tuple notation

RG = (V, T, P, S)

where: V = finite non-empty variable
(vertex/non-terminals)
[upper case letters]

T = finite non-empty variable
(terminals)
[lower case letter]

P production $\alpha \rightarrow B a$ where

$\alpha \in V$

$B \in V, T$

S = starting symbol

Ex:- $S \to aSB$, $S \to aB$, $B \to a$ } → Instructions or production

Derivation:

a) Left derivation

$S \to \underline{a}S\underline{B}$

$S \to a\underline{a}B\underline{B}$

$S \to aa\underline{a}B$

$S \to aaaa$

b) Right derivation

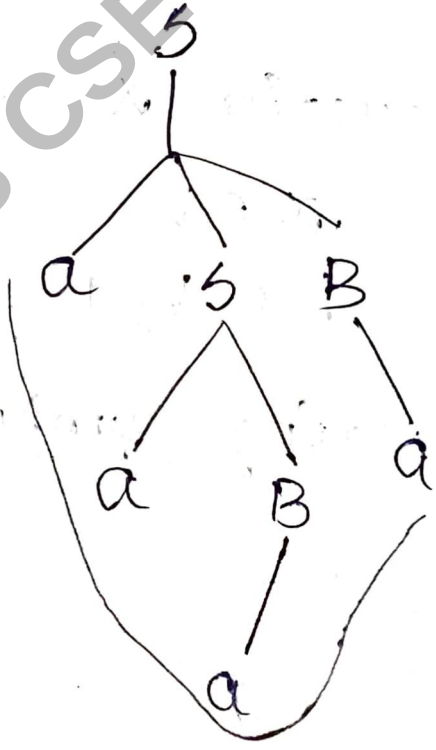
$S \to a\underline{S}\underline{B}$

$a \to a\underline{S}a$

$a \to aa\underline{B}a$

$a \to aaaa$

* Parse Tree :-



aaaa

Types of Regular Grammer :-

TYPE-3 — Regular

TYPE-2

TYPE-1

TYPE-0

⇒ Regular $\alpha \rightarrow a\beta$

$\alpha, \beta \in V$

$a \in T$

(1) Right Linear Grammer (RLG)

$\alpha \rightarrow a\beta$

(Non-terminal is Rightmost)

(2) Left Linear Grammer (LLG)

$\alpha \rightarrow \beta a$

(Non-terminal is leftmost)

⇒ Convert FA → RG (RLG)

Rule 1 : FA : (A) —a→ (B)

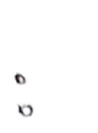
RG : $A \rightarrow aB$

Rule 2 : FA : (A) —a→ ((B))

RG : $A \rightarrow aB$

$A \rightarrow a$ (OR) $B \rightarrow \epsilon$

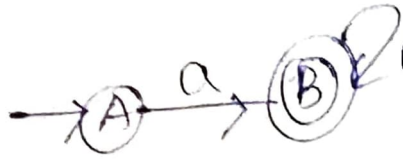
Rule 3: FA:



RG: $A \rightarrow aA$

$A \rightarrow a$ (or) $A \rightarrow \epsilon$

$\Rightarrow$



Ans: RG = $A \rightarrow aB$

$A \rightarrow a$

$B \rightarrow abB$

RG = ({A, B}, {a, b}, P, A)

where

P = $A \rightarrow aB$

$A \rightarrow a$

$B \rightarrow aB$

$B' \rightarrow bB$

$B \rightarrow \epsilon$

$\Rightarrow$

Ans: RG = ({A, B}, {a, b}, P, A)

where,

P = $A \rightarrow bA$

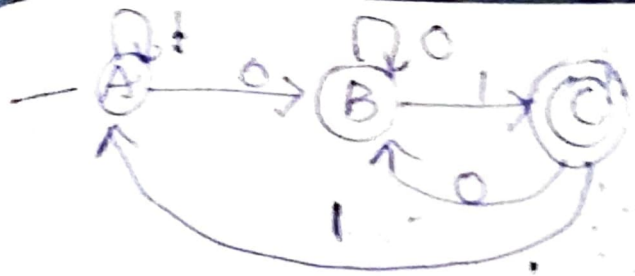
$A \rightarrow \epsilon$

$A \rightarrow aB$

$B \rightarrow bB$

$B \rightarrow aA$

$B \rightarrow a$



RG = ({A, B, C}, {1, 0}, P, A)

where,

P = $A \rightarrow 1A$

$A \rightarrow 0B$

$B \rightarrow 0B$

$B \rightarrow 1C$

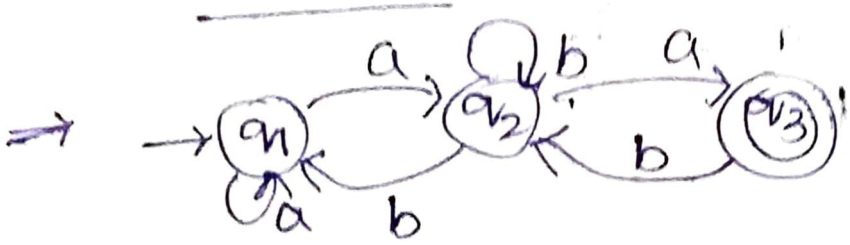
$B \rightarrow 1$

$C \rightarrow \epsilon$

$C \rightarrow 0B$

$C \rightarrow 1A$

⇒ Convert FA → RG1



Ans: $RG_1 = (\{q_1, q_2, q_3\}, \{a, b\}, P, q_1)$

where

$P = q_1 \rightarrow aq_1$

$q_1 \rightarrow aq_2$

$q_2 \rightarrow bq_2$

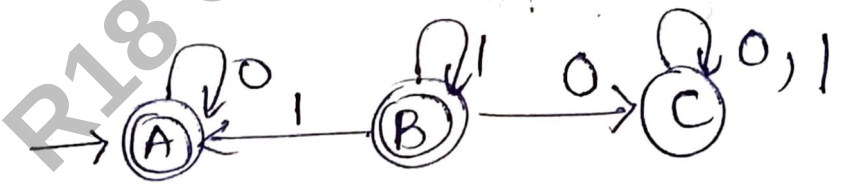
$q_2 \rightarrow aq_3$

$q_2 \rightarrow bq_1$

$q_3 \rightarrow \epsilon$

$q_3 \rightarrow bq_2$

⇒



Ans: $RG_1 = (\{A, B, C\}, \{0, 1\}, P, A)$

where, $P = A \rightarrow \epsilon$

$A \rightarrow 0A$

$B \rightarrow 1B$

$B \rightarrow \epsilon$

$B \rightarrow 1A$

$B \rightarrow 0C$

$C \rightarrow 0C$

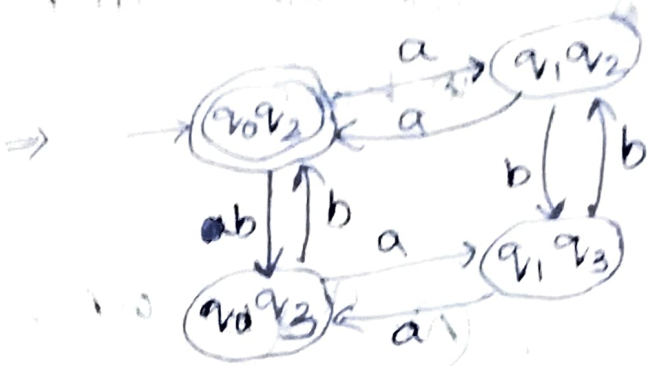
$C \rightarrow 1C$

problems on conversion of FA to RG

→ Even no of a's & Even no of b's.

Even no of a's

Even no of b's



$RG_1 = (\{q_0q_2, q_1q_2, q_0q_3, q_1q_3\}, \{a,b\}, P, q_0q_2)$

P ⇒

$q_0q_2 \rightarrow aq_1q_2 / bq_0q_3 / \epsilon$

$q_1q_2 \rightarrow aq_0q_2 / bq_1q_3$

$q_1q_3 \rightarrow bq_1q_2 / aq_0q_3$

$q_0q_3 \rightarrow aq_1q_3 / bq_0q_2$

⇒ Convert RG to FA

→ No of states in FA = No of Non-Terminals in RG + 1 (final state)

→ Each state in FA = Each Non-Terminal in RG

Method :-

① If P: A → aB ⇒ (A) —a→ (B) δ(A,a) = B

② If P: A → a ⇒ (A) —a→ ((B)) δ(A,a) = Final state

③ If P: A → ∈ ⇒ ((A)) δ(A,∈) = A and A is the final state

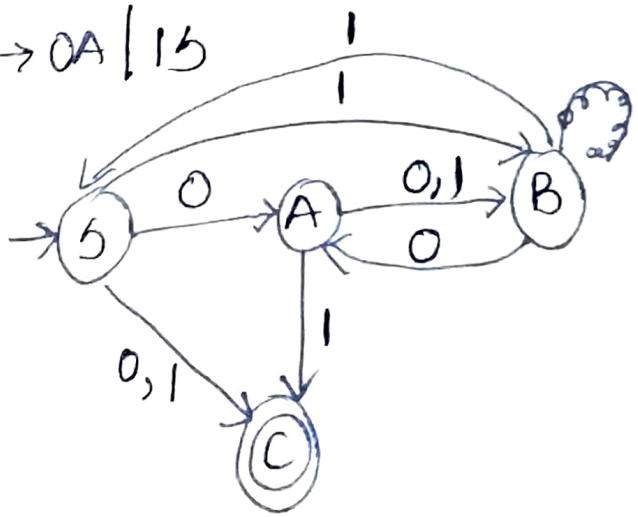
⇒

S → 0A | 1B | 0 | 1

A → 0B | 1B | 1

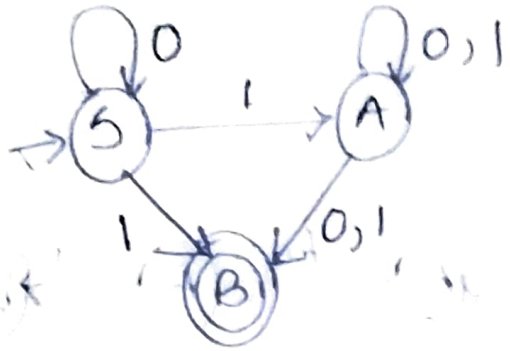
B → 0A | 1S

Ans:



$S \rightarrow 0S \mid 1A \mid 1$

$A \rightarrow 0A \mid 1A \mid 0 \mid 1$

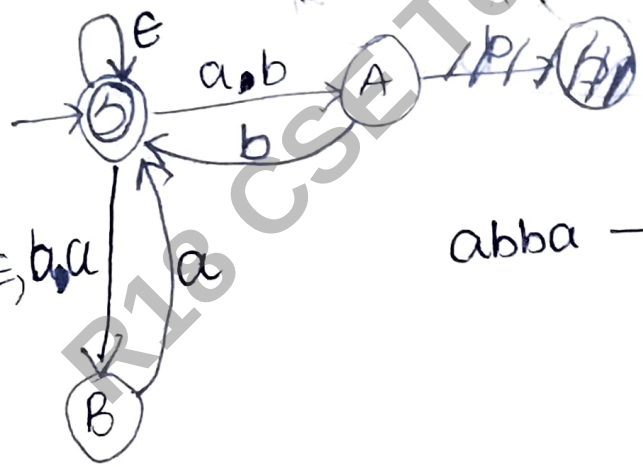


$S \rightarrow abA \mid B \mid baB \mid \epsilon$

$A \rightarrow bS \mid b$

$B \rightarrow aS$

check 'abba'



abba — Yes

⇒ $S \to aS \mid bS \mid aA$

$A \to bB$

$B \to aC$

$C \to \epsilon$

Ans:

⇒ $S \to aS \mid bA \mid b$

$A \to aA \mid bS \mid a$

Ans:

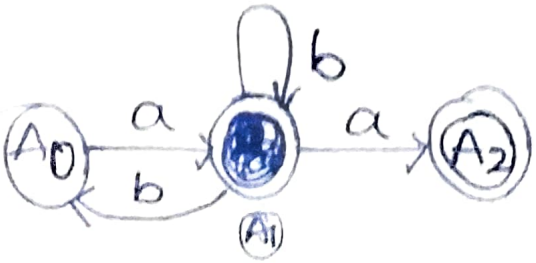
→ $A_0 \to aA_1$

$A_1 \to bA_1$

$A_1 \to a$

$A_1 \to bA_0$

Ans:



Context Free Grammer :- (CFG) [TYPE - 2]

→ All RG are CFG But not vice versa.

→ RG ⊂ CFG

→ Grammer is defined as

$$G_1 = (V, T, P, S)$$

where V = finite set of variables (Non-Terminals)

T = finite set of variables (Terminals)

S = starting state

P = production

↓

It is in the form of

$$A \longrightarrow \alpha$$

$$A \in V$$

$$\alpha \in (V \cup T)^*$$

where α = A string of symbols

→ Differences B/w RG & CFG.

CFG	RG
1) Type - 2 Grammar	1) Type 3 Grammars
2) All CFG are not RG	2) All RG are CFG
3) DFA cannot draw directly.	3) DFA can draw directly
4) Productions $A \to \alpha$ where $A \in V$, $\alpha \in (V \cup T)^*$	4) Productions $A \to \alpha B / \beta$ where $A, B \in V$, $\alpha, \beta \in T^*$

→ Parse tree is used to derive a language.

Derivation of CFG:

Left most derivations : Right most derivations

⇒ Derivation Tree :- (Parse Tree)

→ It is an ordered tree in which the nodes are labelled with left side of production and which the childrens of a node represents its corresponding right side terminals.

Properties :-

Let G = (V, T, P, S) be a CFG,

→ A Tree is a parse tree for a Grammer G. if

① The labelled of root is S

② Every vertex labelled with VUTU{ε}

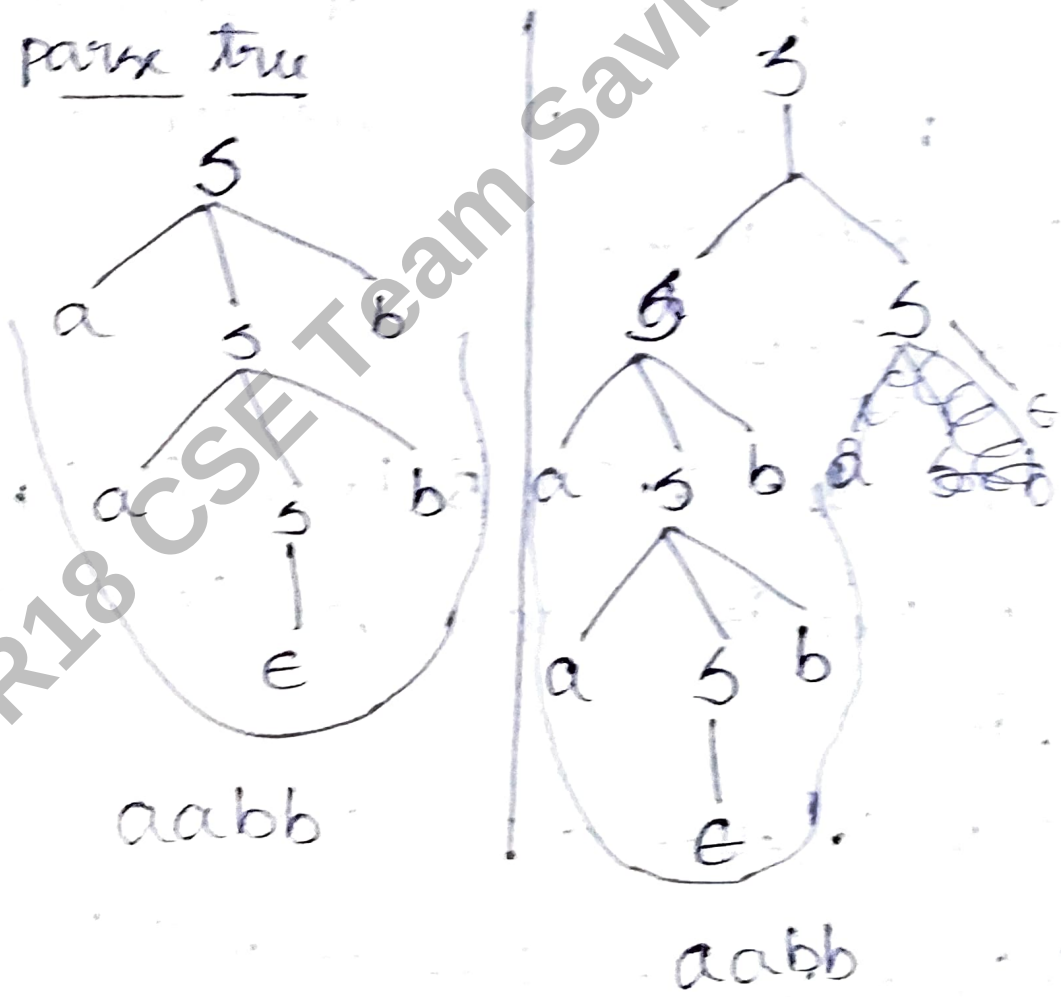
③ If vertex n has labelled with ε then n is a leaf node.

Ambiguity of the Grammer :-

→ CFG is said to be Ambiguous if there Exists some $w \in L(G)$ which has atleast two different parse tree is said to be Ambiguous.

⇒ check the given Grammer $S \to asb / ss$ is ambiguous for "aabb".

Ans :- parse tree



∴ This is Ambiguous Grammer

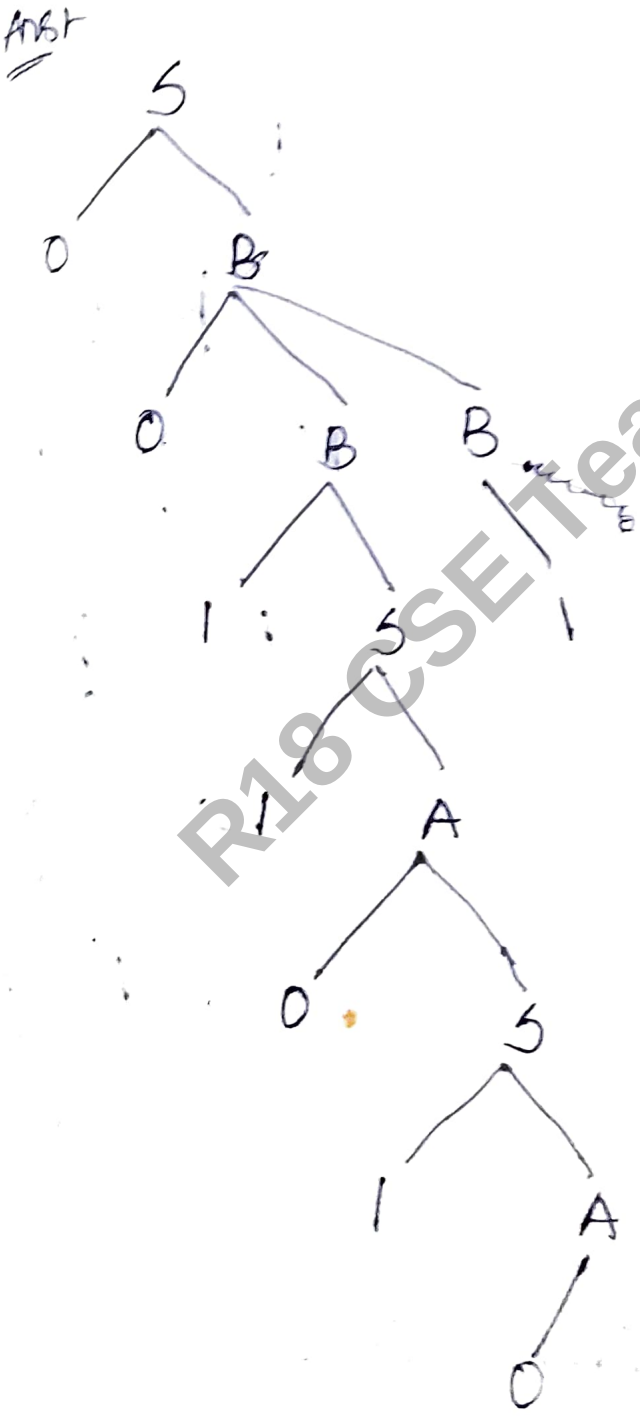
3. Show that the following Grammar is Ambiguious for '00110101'

Grammar: $S \rightarrow 0B/1A$

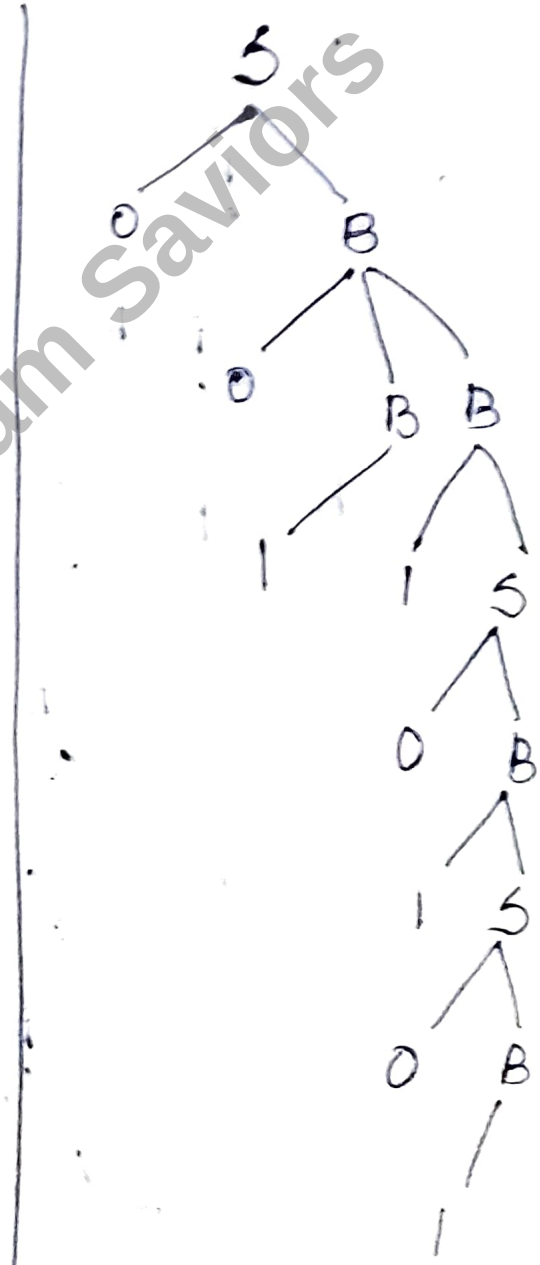
$A \rightarrow 0/0S/1AA$

$B \rightarrow 1/1S/0BB$

Ans:



00110101



00110101

G is Ambiguious

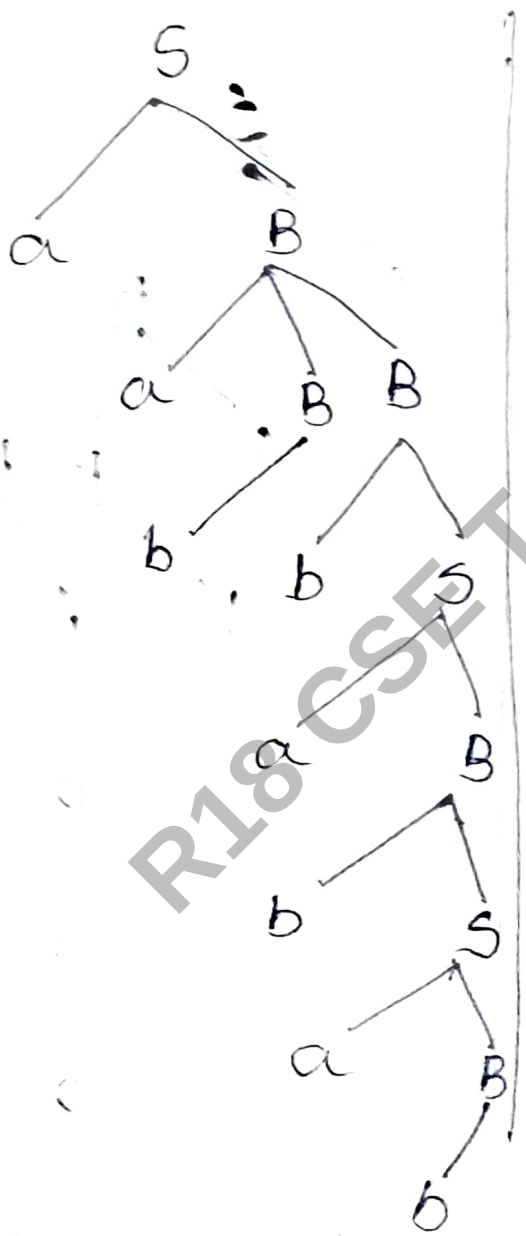
② Grammar : $S \rightarrow aB \mid bA$

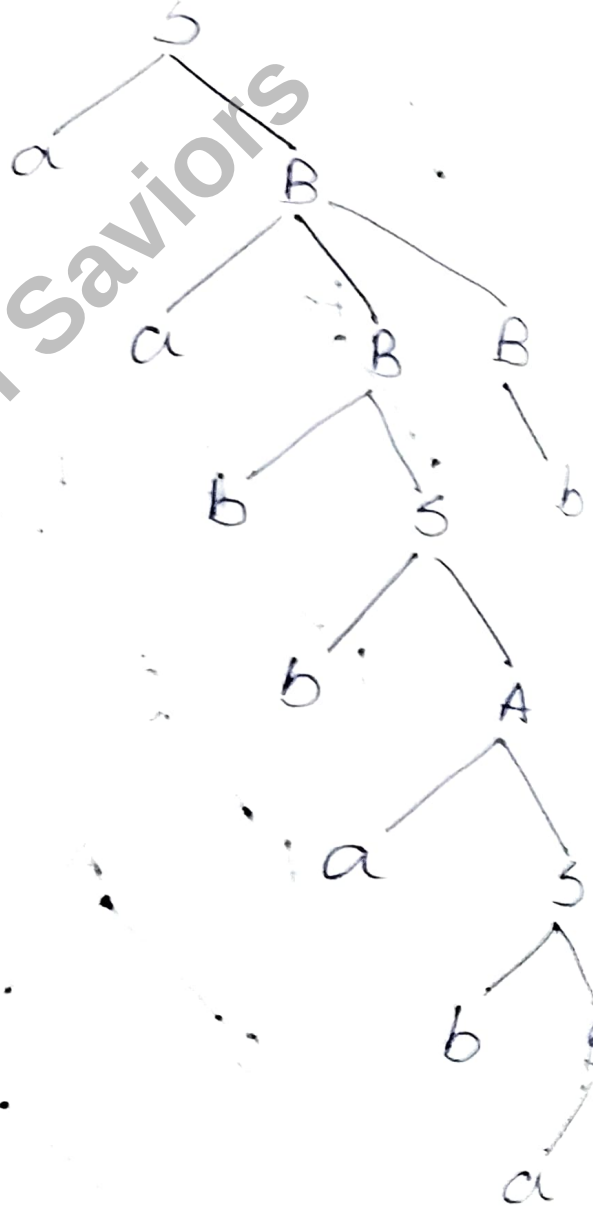
$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

Language : aabbabab

Ans:





G is Ambiguous

Generate CFG for given language:

① L = {any no of a's}

Ans: L = {a, aa, aaa, ...}

RE r = a^*

P: S → aS/a/ε (or) aS/ε

② L = {any no of a's & any no of b's}

Ans: L = {ε, ab, aabb, ...}

RE = a^*b^* $(a+b)^*$

P: S → aS/bS/ε

③ L = {exactly 2 a's over Σ = {a, b}}

Ans: L = {aa, aab, baa, aba, ...}

RE = $b^*ab^*ab^*$

P: S → AaAaA

A → bA/ε

⇒ Generate CFG for a string consisting with a, b and forms palindromes.

Ans: $L = \{ab, ba, \ldots\}$

$L = \{ww^R\}$

$RE = (a+b)^*((a+b)^*)^R$.

$S \to aSa \mid bSb \mid \epsilon$ — even

$S \to aSa \mid bSb \mid a \mid b$ — odd

⇒ string consisting of equal no of a's equal to equal no of b's.

Ans: $L = \{ab, aabb, \ldots\}$

$L = \{n_a(w) = n_b(w)\}$

$S \to SS$

$S \to aSb \mid bSa \mid \epsilon$

⇒ Construct CFG for a string of length $|w| = 2$.

Ans: $L = \{aa, ab, ba, bb\}$

$L = \{ww\}$ $\quad RE = (a+b)(a+b)$

~~$S \to aS/bS/a/b/\epsilon$~~

$S \to AA$

$A \to a/b$

⇒ $|w| \leq 2$

$RE = \epsilon + (a+b) + (a+b)^2$

$= (a+b+\epsilon)(a+b+\epsilon)$

$S \to AA$

~~A~~ $\to a/b/\epsilon$

⇒ $|w| \geq 2$

$RE = (a+b)(a+b)(a+b)^*$

$S \to AAB$

$A \to a/b$

$B \to aB/bB/\epsilon$

⇒ String start with a End with b

Ans: $RE = a(a+b)^* b$

$S \rightarrow aAb$

$A \rightarrow aA/bA/\epsilon$

⇒ Starting & ending with diffrn

Ans: $RE = a(a+b)^* b + b(a+b)^* a$

$S \rightarrow aAb/bAa$

$A \rightarrow aA/bA/\epsilon$

⇒ Start & End with same symbol

Ans: $RE = a(a+b)^* a + b(a+b)^* b + a + b + \epsilon$

$S \rightarrow aAa/bAb/a/b/\epsilon$

$A \rightarrow aA/bA/\epsilon$

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

$S \rightarrow aSb \mid ab$

Get all strings with even length

$\{ ab, aabb, \ldots \}$

$L = \left[\underbrace{(a+b)}_{A} \underbrace{(a+b)}_{A} \right]^*$

$\underbrace{\qquad\qquad}_{B}$

$S \rightarrow BS \mid \epsilon$

$B \rightarrow AA$

$A \rightarrow a \mid b$

$L = \{ a^n b^m \text{ where } n, m \geq 1 \}$

$S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$\Rightarrow L = \{ 0^i 1^j 2^k \mid j > i+k \}$

$L = \{ 001111111\ 222, \ldots \}$

$L = \{ \underbrace{0^i 1^i}_{A} \underbrace{1^k 2^k}_{B} \}$

C

$S \rightarrow ACB$

$A \rightarrow 0A1 \mid \epsilon$

$C \rightarrow 1C \mid 1$

$B \rightarrow 1B2 \mid \epsilon$

① $L = \{ \underbrace{a^n b^n}_{A} \underbrace{c^m}_{B} \mid n, m \geq 1 \}$

② $L = \{ a^n b^n c^m a^m \mid n, m \geq 1 \}$

③ $L = \{ a^n b^n c^n \mid n \geq 1 \}$

④ $L = \{ a^n b^m c^{n+m} \mid n, m \geq 1 \}$

Ans

① $S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow ACB \mid c$

(2) $S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow cBd \mid cd$

(4) $S \rightarrow aSc \mid aAc$

$A \rightarrow bAc \mid bc$

(3) We cannot Generate CFG

→ Simplification of CFG:-

- Eliminating ϵ-productions
- Eliminating unit productions
- Eliminating useless productions

I) Eliminating ϵ-productions

① Find ϵ-productions

i) $A \rightarrow \epsilon$

ii) $A \rightarrow B$, $B \rightarrow \epsilon$ } $A \rightarrow \epsilon$

② Replace with/without

③ Eliminate ϵ-productions.

Eliminate ε productions for the given CFG

$S \to aSb \mid aAb$

$A \to \epsilon$

Ans:

i) Null production $A \to \epsilon$

$S \to aSb \mid aAb \mid ab$

$S \to AB$

$A \to aAA \mid \epsilon$

$B \to bBB \mid \epsilon$

Ans: i) ε - productions

$A \to \epsilon$ $B \to \epsilon$ $\{A, B, S\}$

$S \to \epsilon$

ii) $S \to AB \mid B \mid A \mid \epsilon$

$A \to aAA \mid aA \mid a$

$B \to bBB \mid bB \mid b$

iii)

⇒ Eliminate ϵ - productions

$S \rightarrow AbaC$

$A \rightarrow BC$

$B \rightarrow b/\epsilon$

$C \rightarrow D/\epsilon$

$D \rightarrow d$

Ans: ϵ - productions

$B \rightarrow \epsilon \quad C \rightarrow \epsilon$

$A \rightarrow \epsilon$ $\{A, B, C\}$

$S \rightarrow Abac/bac/Aba/ba$

$A \rightarrow BC/B/C/\epsilon$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow d$

$S \to ABC$

$A \to BC \mid a$

$B \to bAC \mid \epsilon$

$C \to cAB \mid \epsilon$

ϵ-productions

$S \to \epsilon$

$A \to \epsilon \quad B \to \epsilon \quad C \to \epsilon \quad \{S, A, B, C\}$

$S \to ABC \mid AB \mid AC \mid A \mid B \mid C \mid \epsilon$

$A \to BC \mid B \mid C \mid \epsilon \mid a$

$B \to bAC \mid bA \mid bC \mid b$

$C \to cAB \mid cA \mid cB \mid c$

$\Rightarrow S \to ABAC$

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$

$C \to c$

ϵ-productions

$A \to \epsilon \quad B \to \epsilon \quad \{A, B\}$

$S \to ABAC \mid BAC \mid ABC \mid AAC \mid BC \mid AC \mid C$

$A \to aA \mid a$

$B \to bB \mid b \qquad C \to c$

II) Eliminate of unit production :-

Non terminal → one Non-Terminal

i) $A \rightarrow B$

ii) $A \rightarrow B$, $B \rightarrow a$ } $A \rightarrow a$

⇒ eliminate unit productions

$S \rightarrow Aa \mid B$

$B \rightarrow A \mid bb$

$A \rightarrow a \mid bc \mid B$

Ans:

$S \rightarrow B$

$B \rightarrow A$ } unit productions.

$A \rightarrow B$

$S \rightarrow Aa \mid bb \mid a \mid bc$

$B \rightarrow a \mid bc \mid bb$

$A \rightarrow a \mid bc \mid bb$

S → AB

A → a

B → c/b

C → D

D → E

E → a

C → D
D → E
} unit productions

S → AB

A → a

B → b/c

C → a

D → a

E → a

III) Eliminating useless symbols:

1) Symbols which we cannot reach from starting symbol.

2) There is no way to go to terminal.

=> Eliminate useless symbols from G

$S \rightarrow AB/a$

$A \rightarrow BC/b$

$B \rightarrow aB/c$

$C \rightarrow aC/B$

Ans:

Terminals $T = \{a, b\}$

Non-Terminals $V = \{S, A, B, C\}$

i) Cond-1:-

B, C does not form a string

$B \rightarrow aB/c$ } delete this

$C \rightarrow aC/B$ }

ii) Productions after Cond-1

$S \rightarrow a$

$A \rightarrow b$

ii) cond -2

$S \to a$

→ Eliminate useless symbols

$S \to AB / AC$

$A \to aAb / bAa / a$

$B \to bbA / aaB / AB$

$C \to abCA / aDb$

$D \to bD / aC$

Ans $T = \{a, b\}$

$V = \{S, A, B, C, D\}$

i) cond -1 :-

C, D are useless does not form a string

$C \to abCA / aDb$

$D \to bD / aC$

Productions after cond -1

$S \to AB$

$A \to aAb / bAa / a$

$B \to bbA / aaB / AB$

ii) cond 2 :-

$S \rightarrow AB$

$A \rightarrow aAb/bAa/a$

$B \rightarrow bbA$

$\Rightarrow S \rightarrow ABc / BaB$

$A \rightarrow aA / BaC / aaa$

$B \rightarrow bBb / a$

$C \rightarrow CA / Ac$

Ans i) cond 1 :-

$S \rightarrow ABC / BaB$

$A \rightarrow aA / aaa$

$B \rightarrow bBb / a$

} productions

ii) cond II :

$S \rightarrow ABC / BaB$

$S \rightarrow BaB$

$B \rightarrow bBb / a$

Chomsky normal form (CNF):

Format (i): (Non-Terminal) → Exact TWO Non-Terminals

Ex :- S → AB

Format (ii) : (Non-Terminal) → Only one Terminals

A → a

Reduction of CNF FROM CFG :-

Step-1 : Eliminate unit, Null

Step-2 : Eliminate terminal on RHS

Step-3 : Restrict variable in RHS

i) TWO Non-Terminals

ii) Only one terminal.

⇒ Reduce CNF for given productions:

$S \to 1A/0B$

$A \to 1AA/0S/0$

$B \to 0BB/1$

Ans: $T = \{0, 1\}$

$V = \{S, A, B\}$

Step-1:- No unit and Null

Step 2:- $S \to 1A$ — $S \to C_1A$

$S \to 0B$ $C_1 \to 1$

$S \to C_0B$

$A \to 1AA$ $C_0 \to 0$

$A \to 0S$

$A \to 0$

$A \to C_1C_a$

$C_a \to AA$

$B \to 0BB$

$B \to 1$

$A \to C_0S$

$A \to 0$

$B \to C_0C_b$

$C_b \to BB$

$B \to 1$

$S \to C_1A / C_0B$

$A \to C_1Ca / C_0S / 0$

$B \to C_0C_b / 1$

$C_1 \to 1$

$C_0 \to 0$

$Ca \to AA$

$C_b \to BB$

⇒ Reduce CNF for given CFG

$S \to bA / aB$

$A \to bAA / as / a$

$B \to aBB / bS / b$

Ans: $T = \{a, b\}$ $V = \{S, A, B\}$

Step-1:- No unit & Null

Step-2:- $S \to bA$ — $S \to C_1A$, $C_1 \to b$

$S \to aB$ — $S \to C_0B$, $C_0 \to a$

$A \to bAA$ — $A \to C_1Ca$, $Ca \to AA$

$A \to as$ — $A \to C_0S$, $A \to a$

$B \to C_0C_b$, $B \to BB$, $C_b \to C_1S$, $C_1 \to b$

$A \to a$

$B \to aBB$

$B \to bS$

Ans:

$S \to C_1A \mid C_0B$

$A \to C_1C_a \mid C_0S \mid a$

$B \to C_0C_b \mid C_1S \mid b$

$C_1 \to b$

$C_0 \to a$

$C_a \to AA$

$C_b \to bb$

$\Rightarrow S \to aA \mid bB$

$A \to aA \mid a$

$B \to bB \mid b$

Ans: Step-1: No unit & Null

Step-2: $S \to aAbB$ — $S \to C_aC_b$, $C_a \to aA$, $C_b \to bB$

$A \to aA$ — $A \to C_0A$, $C_0 \to a$

$B \to bB$ — $B \to C_1B$, $C_1 \to b$

✓ $A \to a$

✓ $B \to b$

Step-3:

$\rightarrow C_0A$

C_a

$C_b \rightarrow C_1B$

Ans

$S \rightarrow C_aC_b$

$A \rightarrow C_0A$

$B \rightarrow C_1B$

$C_0 \rightarrow a$

$C_1 \rightarrow b$

$C_a \rightarrow C_0A$

$C_b \rightarrow C_1B$

$A \rightarrow a$

$B \rightarrow b$

⇒ find CNF for language.

$$L = \{a^{2n} / n \geq 1\}$$

$$L = \{aa, aaaa, \ldots\}$$

Ans:

Convert to CFG:

$$S \to aaS$$

$$S \to aa$$

Convert CFG into CNF:-

Step-1: No null, unit

Step-2: $S \to CaCaS$

$$Ca \to a$$

$$S \to CaCa$$

Step-3:

Ans: $S \to C_1S$

$$C_1 \to CaCa$$

$$Ca \to a$$

$$S \to CaCa$$

GREIBACH Normal Form (GNF) :-

NF

CNF

① $A \to BC$

$A \to a$

where

$A, B, C \in V$

$a \in T$

② Restriction on RHS variable

GNF

① $S \to \epsilon$

$A \to a\alpha$

$A \to a$

where

$S, A \in V$

$a \in T$

$\alpha \in (V)^*$

② NO Restriction on RHS variable

Converse GNF :-

Step-1 :- Eliminate null, unit productions
Eliminate useless symbols

Step-2 :- convert CFG into CNF.

Step-3 :- convert CNF to GNF

a) change the names of the variables

b) check a from $A_i \to A_j$ $i < j$ always

But not $i \geq j$

c) check left Recursion is there or not

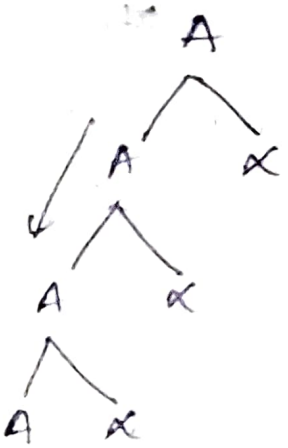
if Yes, eliminate Left recursion.

b) Repeat above step until productions are in GNF.

Recursion

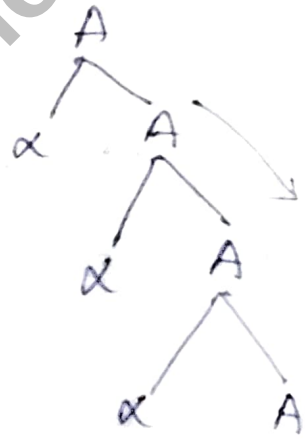
Left Recursion

$A \rightarrow A\alpha | \beta_1 | \beta_2$



Right Recursion

$A \rightarrow \alpha A | \beta_1 | \beta_2$



→ To eliminate Left Recursion

Introduce — Non-Terminal

⇓

X

Productions:

$A \rightarrow \beta_1 | \beta_2 | \beta_1 X | \beta_2 X$

$X \rightarrow \alpha | \alpha X$

LEMMA

→ Convert the following grammar into GNF.

Ans:-

$$S \rightarrow AA|0$$

$$A \rightarrow SS|1$$

step-1:- No unit; null; useless

step-2:- Convert CFG into CNF

→ already in CNF

step-3:- Rename the variable

$$S \rightarrow A_1$$

$$A \rightarrow A_2$$

Productions:-

$A_i \rightarrow A_j$

$i < j$

$A_1 \rightarrow A_2A_2|0$

$A_2 \rightarrow A_1A_1|1$ → not in Rule

Replace A_1 with A_1 production

$$A_2 \rightarrow A_2A_2A_1|0A_1|1$$

↑

Left recursion

eliminate left Recursion

$A_2 \rightarrow A_2 \underbrace{A_2A_1}_{\alpha} / \underbrace{0A_1}_{\beta_1} / \underbrace{1}_{\beta_2}$

Introduce new variable X

$A_2 \rightarrow 0A_1 / 1 / 0A_1X / 1X$ ✓

$X \rightarrow A_2A_1 / A_2A_1X$

$A_1 \rightarrow A_2A_2 / 0$

Replace A_2 in X, A_1 with A_2 productions

$A_2 \rightarrow 0A_1 / 1 / 0A_1X / 1X$

$X \rightarrow 0A_1A_1 / 1A_1 / 0A_1XA_1 / 1XA_1 /$
$0A_1A_1X / 1A_1X / 0A_1XA_1X / 1XA_1X$

$A_1 \rightarrow 0A_1A_2 / 1A_2 / 0A_1XA_2 / 1XA_2 / 0$

$S \rightarrow XB \mid AA$

$A \rightarrow a \mid SA$

$B \rightarrow b$

$X \rightarrow a$

Ans:

Step-1: No unit, null, useless

Step-2: Already in CNF

Step-3: Rename the variable

$S \rightarrow A_1$

$X \rightarrow A_2$

$B \rightarrow A_3$

$A \rightarrow A_4$

$i \geq j$ ✗

$A_i \rightarrow A_j$

$i < j$

Productions:

$A_1 \rightarrow A_2A_3 \mid A_4A_4$ ✓

$A_4 \rightarrow a \mid A_1A_4$ ✗

$A_3 \rightarrow b$ ✓

$A_2 \rightarrow a$ ✓

⇒ find GNF for $S \to AB$

$A \to BS|a$

$B \to SA|b$

Ans:

Step-1 :- No unit, null & useless

Step-2 :- Already in CNF

Step-3 :-

Rename the variables

$S \to A_1$

$A \to A_2$

$B \to A_3$

Productions:

$A_1 \to A_2A_3$ ✓ , $A_i \to A_j$

$A_2 \to A_3A_1 | a$ ✗ $i<j$

$A_3 \to A_1A_2 | b$ ✗

Sub A_1 productions in A_3

$A_3 \to A_2A_3A_2 | b$ ✗

Sub A_2 productions in A_3

$\rightarrow A_2 A_1 A_3 A_2 \mid a A_3 A_2 \mid b$ — Left Recursion

Remove Left Recursion by using a Variable

$A_2 \rightarrow a A_3 A_2 \mid b \mid a A_3 A_2 Z \mid b Z$

$Z \rightarrow A_1 A_3 A_2 \mid A_1 A_3 A_2 Z$

Grammar:

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 \mid a$

Sub A_3 in A_2

$A_2 \rightarrow a A_3 A_2 A_1 \mid b A_1 \mid a A_3 A_2 Z A_1 \mid b Z A_1 \mid a$

✓GNF

Sub A_2 in A_1

$A_1 \rightarrow a A_3 A_2 A_1 A_3 \mid b A_1 A_3 \mid a A_3 A_2 Z A_1 A_3 \mid b Z A_1 A_3 \mid a A_3$

✓GNF

Sub A_1 in Z

$Z \rightarrow a A_3 A_2 A_1 A_3 A_3 A_2 \mid b A_1 A_3 A_3 A_2 \mid$

$a A_3 A_2 Z A_1 A_3 A_3 A_2 \mid b Z A_1 A_3 A_3 A_2 \mid a A_3 A_3 A_2$

S → ab

S → aS

S → aaS

step-1 :- No unit, null & useless

step-2 :- convert CNF

S → ab — S → CaCb
Ca → a
Cb → b

S → aS → S → CaS

S → aaS → S → CaCaS
S → CaS
Ca → Ca

Productions:

S → CaCb | CaS | a

Sub A_2 in A_4

$A_4 \rightarrow aA_2$ ✓ GNF

Sub A_2 and A_4 in A_1

$A_1 \rightarrow aA_3 / aA_1 / aA_2A_1$ ✓ GNF

$\Rightarrow$ $S \rightarrow aAbB$

$A \rightarrow aA / a$ ✓ GNF

$B \rightarrow bB / b$ ✓ GNF

PUSH DOWN Automata (PDA)

CFG

↓

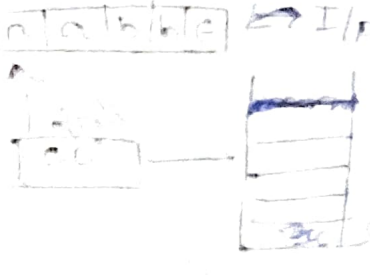
CFL

↓

PDA ⟶ PDA = FA + stack

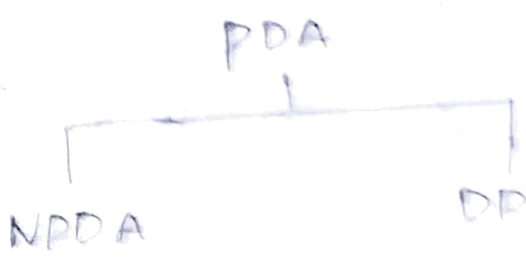
NPDA DPDA

→ PDA is essentially a finite Automata with control of both input tape and stack.



- PDA is also called as stack automata
- It works on LIFO principle
- Operations of PDA
  1. PUSH into stack
  2. POP into stack
  3. no change in the stack

- Types of PDA:



- Notation for PDA — 7 tuple notation

$$PDA = (Q, \Sigma, \Gamma, \delta, Z_0, q_0, F)$$

Q = Non-empty finite set of states

Σ = Non-empty finite set of i/p symbols

Γ = stack symbol

δ = Transition function

Z_0 = Top of the stack

q_0 = Initial state $q_0 \in Q$

F = Final state $F \subseteq Q$

DPDA :- $\delta = Q \times (\Sigma \cup \epsilon) \times T \longrightarrow Q \times T^*$

NPDA :- $\delta : Q \times (\Sigma \cup \epsilon) \times T \longrightarrow {}_2^Q \times T^*$

Instantaneous Description (ID) :- (3 tuple)

Representation = $\boxed{(q, w, \gamma)}$

$q \rightarrow$ current state $q \in Q$

$w \rightarrow$ I/P symbol $w \in \Sigma$

$\gamma \rightarrow$ Top of stack $\gamma \in T$

$\rightarrow$

Acceptance of String :-

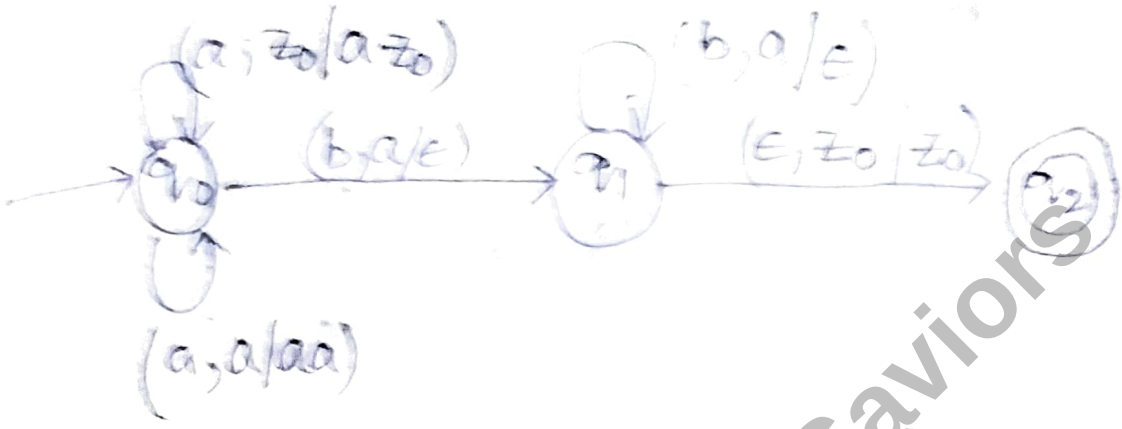
→ Two ways to Accept string

- Empty stack
- Reaching Final State

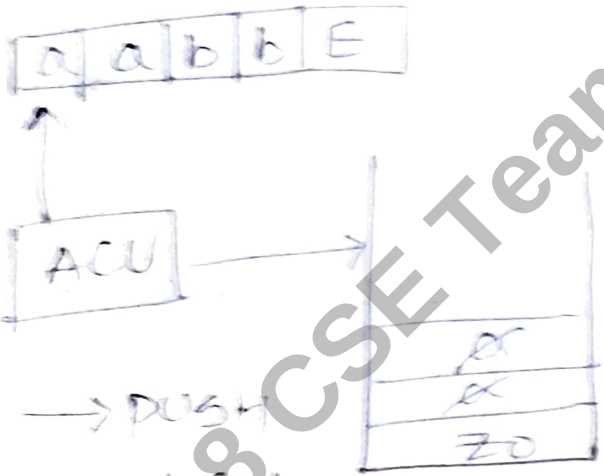
Construct PDA for a Language

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

ex: $L = \{aabb, \ldots\}$



⇒



a Z0 → PUSH

a a → PUSH

a a → POP

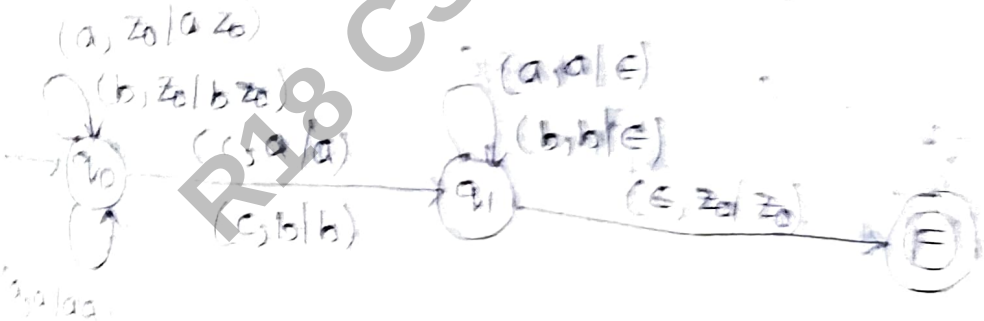
b a → POP

ε Z0 → ε

Construct PDA for $L = \{wcw^R \mid w \in (a+b)^*\}$

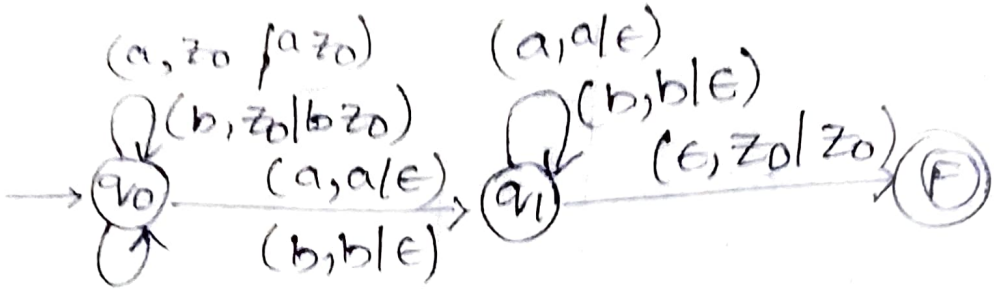
$a, b, c \in \{a, b, c\}$

i/p	top	
a	Z_0	push a
b	Z_0	push b
a	a	push a
b	b	push b
c	b	pop b
b	a	pop b
c	a/b	nothing
a	a	pop
b	b	pop



⇒ Construct PDA for $L = \{ww^R / w \in \{a, b\}^*\}$

Ans:



(a, a/aa)

(b, b/bb)

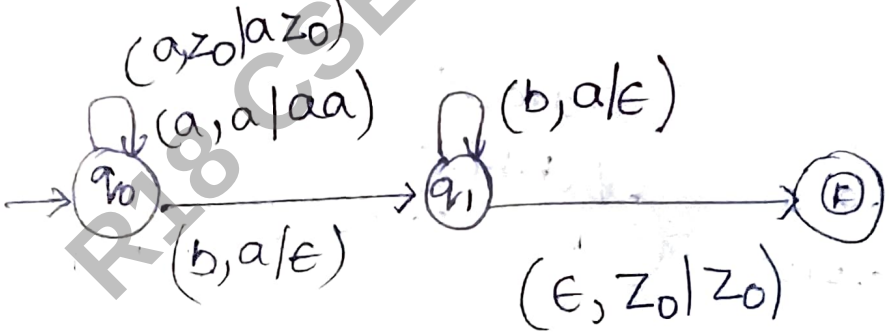
(a, b/ab)

(b, a/ba)

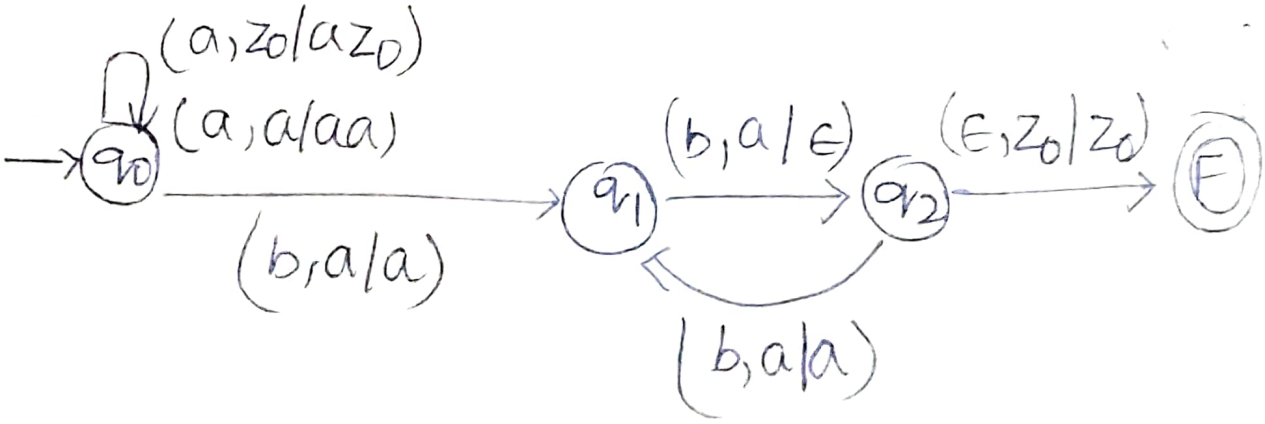
⇒ $L = \{a^n b^{2n} / n \geq 1\}$

Ans:- $L = \{aabbbbbb, \ldots\}$

Method-1 :-

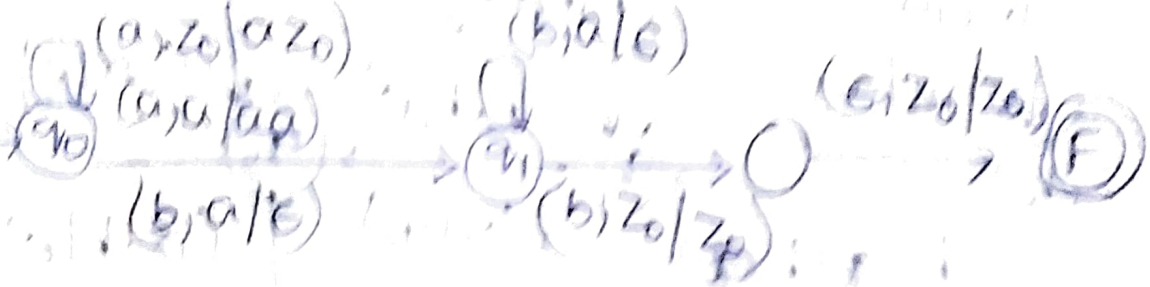


Method-2 :-



$L = \{a^n b^{2n} / n \geq 1\}$

$L = \{a^n b^{2n} b, \}$



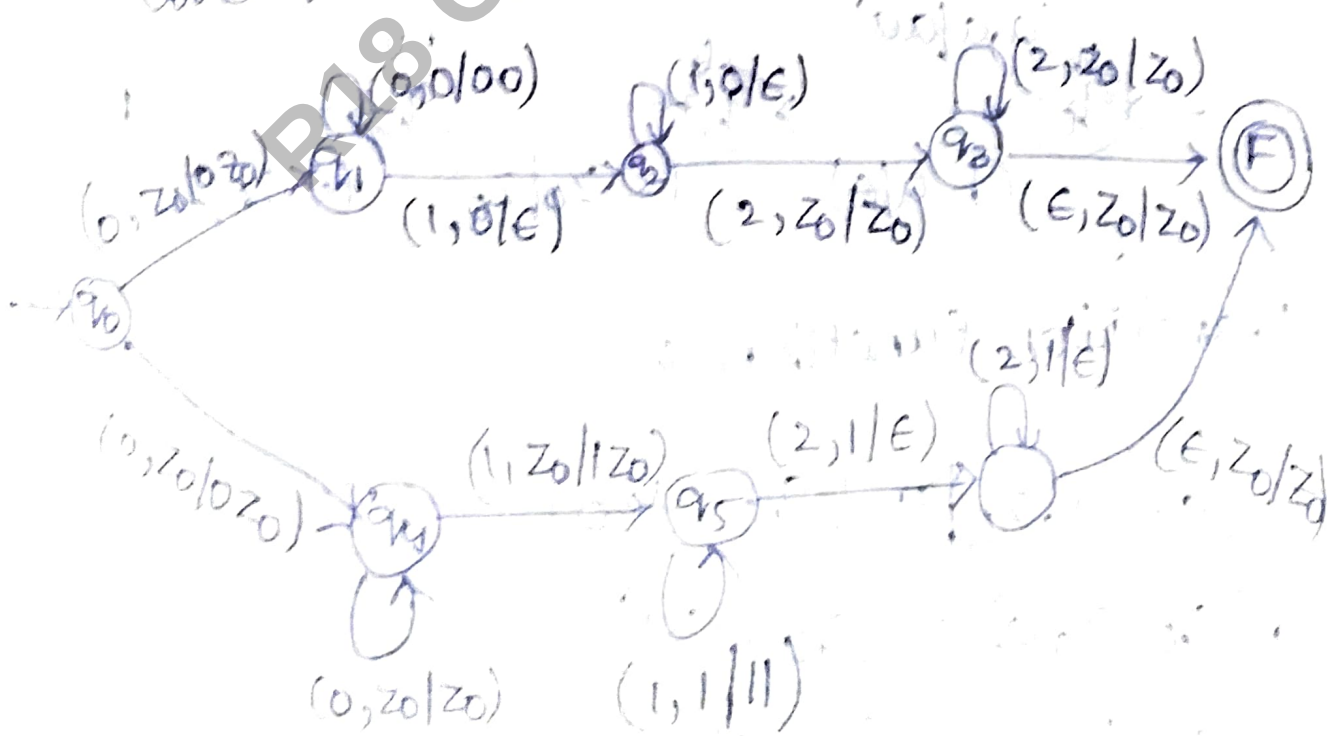
$\Rightarrow L = \{a^n b^n c^n / n \geq 1\}$

It is not in CFG

Construct PDA

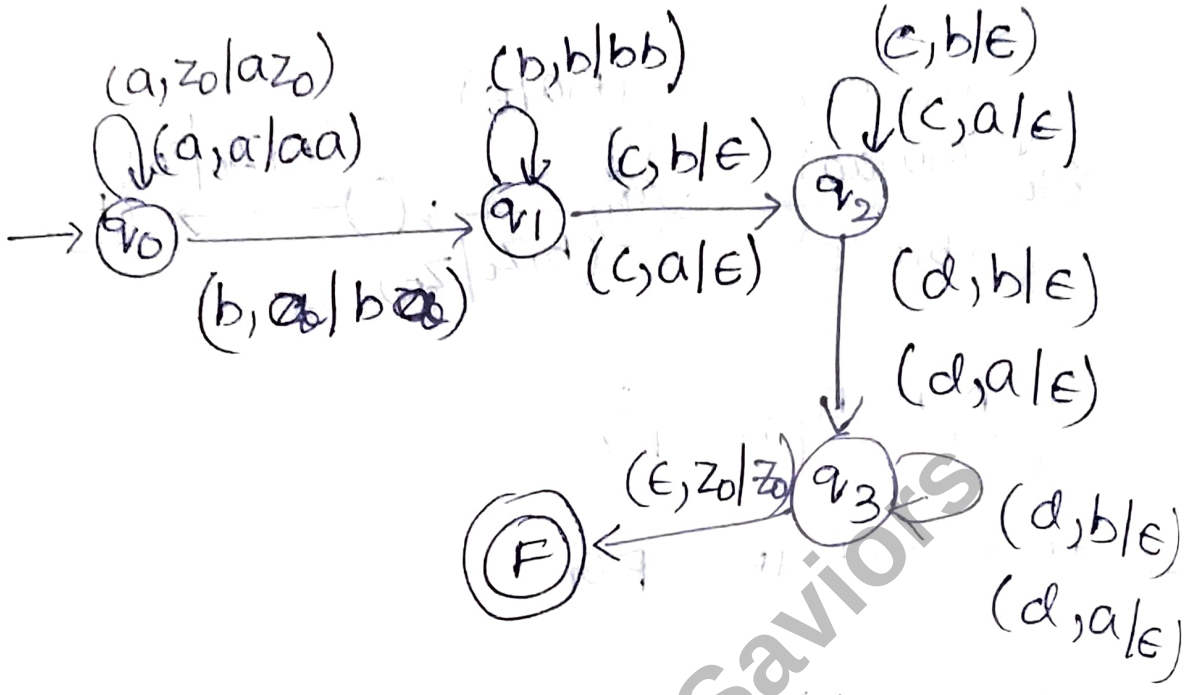
$L = \{0^i 1^j 2^k / i = j \text{ or } j = k \quad i, j, k \geq 1\}$

cond 1 : $i = j$: cond 2 : $j = k$



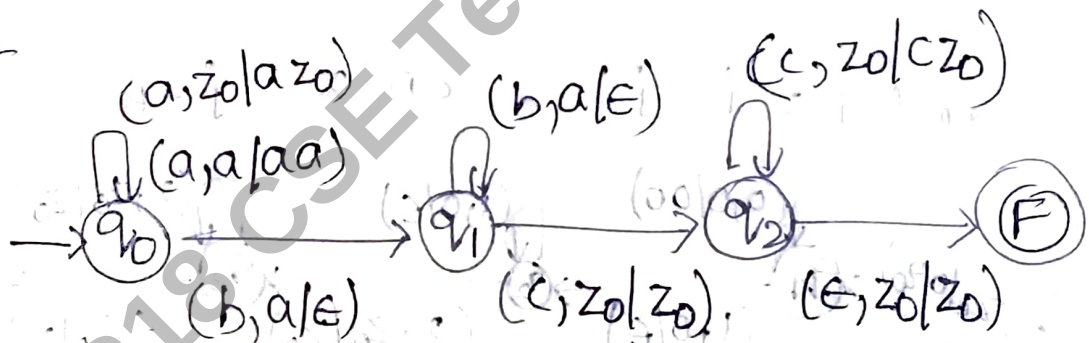
$\Rightarrow L = \{a^n b^m c^p d^q / n+m = p+q / n,m,p,q \geq 1\}$

Ans:-



$\Rightarrow L = \{a^n b^n c^m, n,m \geq 1\}$

Ans:-



Transition function δ :-

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, c, z_0) = (q_2, z_0)$

$\delta(q_2, c, z_0) = (q_2, z_0)$

$\delta(q_2, \epsilon, z_0) = (q_3, z_0)$

$q_3 \rightarrow$ final state

Convert CFG TO PDA:-

CFG = $G = (V\ T\ P\ S)$

↳ PDA = $M = (Q, \Sigma, T, z_0, q_0, F, \delta)$

step-1 :- Convert CFG into Normal forms

step-2 :- PDA will have only one state i.e.,

$\{q\}$

step-3 :- $Q = \{q\}$

Σ = Terminals of CFG (T)

$T = (z_0 \cup V \cup T)$

$q_0 = q$

$z_0 = S$

$F = \phi$ (Empty state)

Step-4 :- Transition Function 'δ'

i) Non Terminals

$$A \to \alpha \Big/ \begin{matrix} A \in V \\ \alpha \in V \cup T \end{matrix}$$

$(q, \epsilon, A) = (q, \alpha)$ Transition PDA

ii). Terminals

$$A \to a \Big/ \begin{matrix} A \in V \\ a \in T \end{matrix}$$

$(q, a, a) = (q, \epsilon)$ Transition PDA

Problems :-

(or) PDA equivalent to FFA

⇒ Convert CFG to PDA for

$$S \to .OBB$$
$$B \to OS/1S/0$$

Ans :-

Step-1 :- Already in NF

Step-2 :- $\{q\}$

Step-3 :- PDA $= (\{q\}, \{0, 1\}, \{S, B, 0, 1\}, S, q, \phi, \delta)$

Step-4:-

Transition Function 'δ'

i) Non-terminals :- $S \rightarrow OBB$

$$\delta(q, \epsilon, S) = (q, OBB)$$

$$\delta(q, \epsilon, B) = (q, OS) \quad B \rightarrow OS$$

$$\delta(q, \epsilon, B) = (q, 1S) \quad B \rightarrow 1S$$

$$\delta(q, \epsilon, B) = (q, 0)$$

ii) Terminals :-

$$\delta(q, 0/0) = (q, \epsilon)$$

$$\delta(q, 1, 1) = (q, \epsilon)$$

⇒ Construct PDA Equivalents to a Grammar

CFG is

$$E \rightarrow E+E/E*E/a$$

Ans:- $T = \{a, +, *\}$ $V = \{E\}$

Step-1 :- Already in NF

Step-2 :- $\{q\}$

Step-3 :- PDA $= (\{q\}, \{a, +, *\}, \{\epsilon, a, +, *\}, E, q, Q, \delta)$

Step 4 :-

Transition state δ:

i) Non-Terminal :-

$\delta(q, \epsilon, E) = (q, E+E)$ / $E \to E+E$

$\delta(q, \epsilon, E) = (q, E*E)$ / $E \to E*E$

$\delta(q, \epsilon, E) = (q, a)$ / $E \to a$

ii) Terminals :-

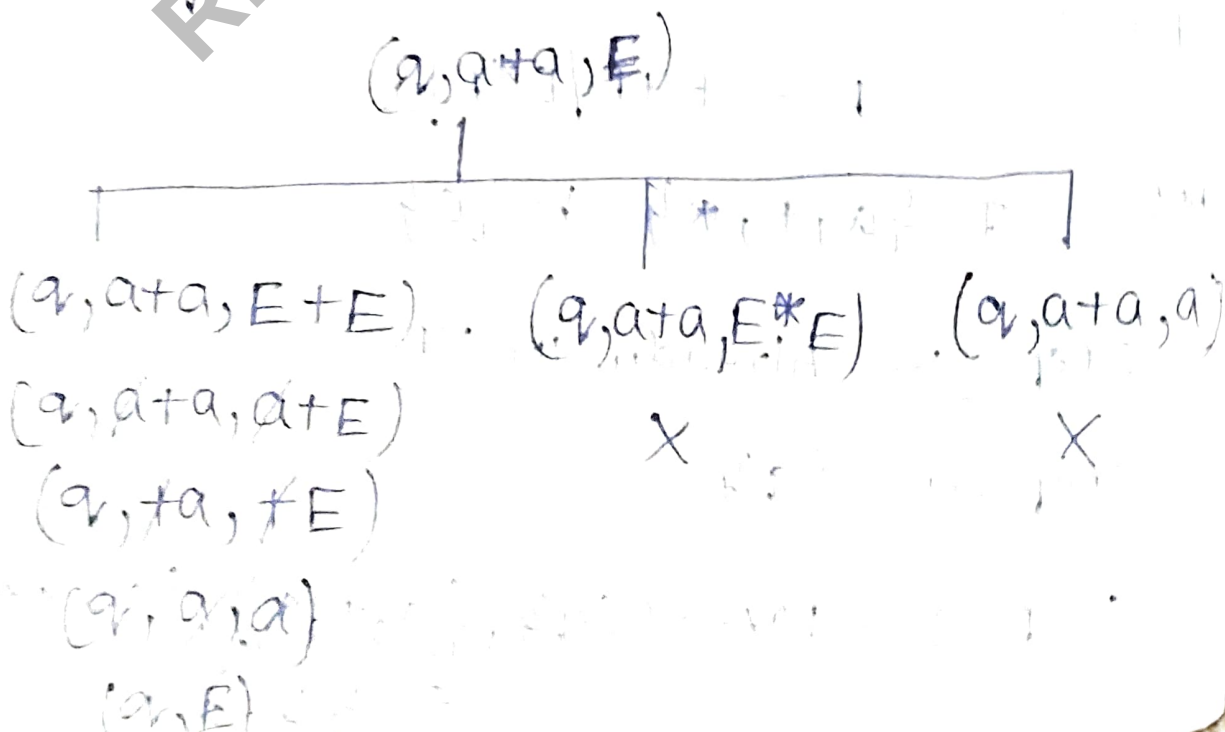
$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, +, +) = (q, \epsilon)$

$\delta(q, *, *) = (q, \epsilon)$

Check :-

string = "a+a" $E \to E+E/E*E/a$



3. construct PDA equivalent to CFG

$I \rightarrow a|b|Ia|Ib|Io|I1$

$E \rightarrow I|E*E|E+E|(E)$

Sol:

$\Sigma = \{a, b, 0, 1, *, +, (,)\}$

$V = \{I, E\}$

step-1 :- Already in NF

step-2 :- $\{q\}$

step-3 :- $PDA = (\{q\}, \{a, b, 0, 1, *, +, (,)\}, \{I, E, a, b, 0, 1, *, +, (,)\}, I, q, \emptyset, \delta)$

step-4 :- Transition Function (δ)

i) Non-terminals :-

$\delta(q, \epsilon, I) = (q, a)$

$\delta(q, \epsilon, I) = (q, b)$

$\delta(q, \epsilon, I) = (q, Ia)$

$\delta(q, \epsilon, I) = (q, Ib)$

$\delta(q, \epsilon, I) = (q, Io)$

$\delta(q, \epsilon, I) = (q, I1)$

$\delta(q, \epsilon, E) = (q, I)$

$\delta(q, \epsilon, E) = (q, E*E)$

$\delta(q, \epsilon, E) = (q, E+E)$

$\delta(q, \epsilon, E) = (q, (E))$

ii) Terminals :-

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

$\delta(q, 0, 0) = (q, \epsilon)$

$\delta(q, 1, 1) = (q, \epsilon)$

$\delta(q, *, *) = (q, \epsilon)$

$\delta(q, +, +) = (q, \epsilon)$

$\delta(q, (, () = (q, \epsilon)$

$\delta(q,),)) = (q, \epsilon)$

⇒ construct PDA equivalent to CFG

i) $S \rightarrow aAA$

$A \rightarrow aS / bS / a$

Ans :-

i) Non-terminals :-

$\delta(q, \epsilon, S) = (q, aAA)$

$\delta(q, \epsilon, A) = \{(q, aS), (q, bS), (q, a)\}$

ii) terminals :-

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

ii) $S \rightarrow AA$

$A \rightarrow SA \mid b$

Ans:-

Step-4:-

i) Non-terminals:-

$\delta(q_1, \epsilon, S) = (q_1, AA)$

$\delta(q_1, \epsilon, A) = \{(q_1, SA), (q_1, b)\}$

ii) terminals:-

$\delta(q_1, a, b) = (q_1, \epsilon)$

⇒ Convert PDA to CFG:-

Let PDA $= (Q, \Sigma, T, Z_0, q_0, F, \delta)$

$CFG = (V, T, P, S)$

Step-1:- $T = \Sigma$ (input symbols of PDA)

$S =$ Starting state of PDA

Step-2:- $V =$ Non-terminals. [Q T Q]

$T = Z_0, a$

$\{S, [q_0 Z_0 q_0]\ [q_0 Z_0 q_1]\ [q_1 Z_0 q_0]\ [q_1 Z_0 q_1]$
$[q_0 a q_0]\ [q_0 a q_1]\ [q_1 a q_0]\ [q_1 a q_1]\}$

$S \rightarrow [q_0 T q_0] / [q_0 T q_1]$

Step-3:- define Productions P

i) Pop transition:-

$\delta(q_0, a, z_0) = (q_0, \epsilon)$

Production:

$[q_0 z_0 q_0] \longrightarrow a$

ii) No change transition:-

$\delta(q_0, a, z_0) = (q_1, z_0)$

Production:-

$[q_0 z_0 q_0] \longrightarrow a[q_1 z_0 q_0]$

$[q_0 z_0 q_1] \longrightarrow a[q_1 z_0 q_1]$

iii) Push transitions:-

$\delta(q_0, a, z_0) = (q_0, a z_0)$

Productions:-

$[q_0 z_0 q_0] \longrightarrow a[q_0 a q_0][q_0 z_0 q_0]$

$[q_0 z_0 q_1] \longrightarrow a[q_0 a q_0][q_0 z_0 q_1]$

$[q_0 z_0 q_1] \longrightarrow a[q_0 a q_1][q_1 z_0 q_1]$

$[q_0 z_0 q_0] \longrightarrow a[q_0 a q_1][q_1 z_0 q_0]$

push two symbols:-

$\delta(q_0, a, z_0) = (q_0, aaz_0)$

$[q_0\ z_0\ -] \longrightarrow a[q_0\ a\ -][-\ a\ -][-\ z_0\ -]$

8 productions

⇒ construct CFG for equivalent PDA for a language $L = \{a^n b^n / n \geq 1\}$

ns:- $\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z) = (q_2, \epsilon)$

Convert CFG:-

Step-1:- $T = \{a, b\}$ $\quad q_0 q_1 \in Q$

$S = q_0$

Step-2:-

i) $V = \{S, [q_0 z_0 q_0] [q_0 z_0 q_1] [q_1 z_0 q_0] [q_1 z_0 q_1]$

$[q_0 a q_0] [q_0 a q_1] [q_1 a q_0] [q_1 a q_1]\}$

ii) starting production

$S \longrightarrow [q_0 z_0 q_0] / [q_0 z_0 q_1]$

Step-3:- Define productions

i) pop transition:-

$\delta(q_0, b, a) = (q_1, \epsilon)$

$[q_0\ a\ q_1] \longrightarrow b$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$[q_1\ a\ q_1] \longrightarrow b$

$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

$[q_1\ z_0\ q_1] \longrightarrow \epsilon$

ii) push transition:-

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$[q_0\ z_0\ q_0] \longrightarrow a[q_0\ a\ q_0][q_0\ z_0\ q_0]$

$[q_0\ z_0\ q_0] \longrightarrow a[q_0\ a\ q_1][q_1\ z_0\ q_0]$

$[q_0\ z_0\ q_1] \longrightarrow a[q_0\ a\ q_0][q_0\ z_0\ q_1]$

$[q_0\ z_0\ q_1] \longrightarrow a[q_0\ a\ q_1][q_1\ z_0\ q_1]$

$\therefore (q_0, a, a) = (q_0, aa)$

$[q_0 \; a \; q_0] \rightarrow a \; [q_0 \; a \; q_0][q_0 \; a \; q_0]$

$[q_0 \; a \; q_1] \rightarrow a \; [q_0 \; a \; q_1][q_1 \; a \; q_1]$

$[q_0 \; a \; q_1] \rightarrow a \; [q_0 \; a \; q_0][q_0 \; a \; q_1]$

$[q_0 \; a \; q_1] \rightarrow a \; [q_0 \; a \; q_1][q_1 \; a \; q_1]$

Pumping Lemma

Discuss on — ① Regular Languages
② Context free grammars.

I Pumping Lemma for Regular Languages

- Why ? (i) $L = \{ a^n b^n / n \geq 0 \}$ not a regular.

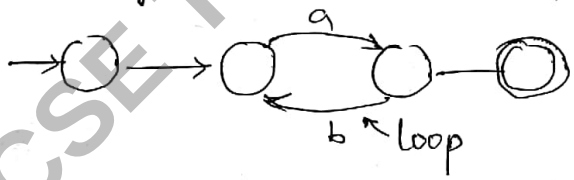
- Applications :- 1, Used to show Language is regular or not.

2, Procedure to show not regular
(i) if L is a regular, it satisfies Lemma
(ii) if L is not satisfies ⇒ it is a not Regular Lemma.

- Regular Languages :-

L → (i) if Language is finite ⇒ R.E ⇒ R.L.
(ii) if infinite : $L = \{ ab, abab, ababab, \ldots \}$.

We can solve by Finite Automata if. Loop can generate FA.
no Loop cannot " FA.

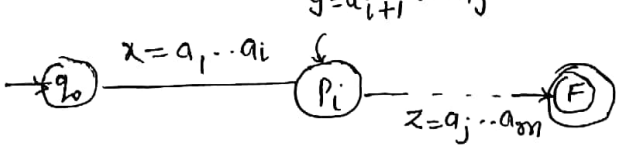


Lemma : Let L be a regular Language, Then there exist a constant n (which depends on L) such that for every string $w \in L$ such that $|w| \geq n$, we can break 'w' into three strings $w = xyz$ such that :

(i) $y \neq \epsilon$ or ≥ 1
(ii) $|xy| \leq n$
(iii) For all $k \geq 0$, the string $xy^k z \in L$

∴ n = no. of states of FA = Pumping Length.

∴ $x = a_1 a_2 \cdots a_i$
$y = a_{i+1}, a_{i+2} \cdots a_j$
$z = a_{j+1}, a_{j+2} \cdots a_m$



Problems:

(1) Prove that $L = \{0^n 1^n / n \geq 1\}$

(i) It is not a finite Language $\Rightarrow$ not regular Language.

(ii) As per pumping Lemma.

(i) Assume L = is Regular, n = no. of states

(ii) Let $w = 0^n 1^n$ thus $|w| = 2n \geq n$

(iii) Let $w = xyz$ where $|xy| \leq n$

(iv) Break string w into three strings such that

$|xy| \leq n$, (v) $|y| \geq 1$ — (vi)

$\therefore x = 0^a$, $y = 0^b$, $z = 0^c 1^n$ where $a+b \leq n$, $b \geq 1$, $c \geq 0$

$\therefore a+b+c = n$.

But condition (iii) $k = 0$:

$xy^0z = xz = 0^a 0^c 1^n$ $\therefore a+c \neq n$.

Hence $w \notin L \Rightarrow$ It is not a regular.

Method 2 $L = \{0^n 1^n / n \geq 1\}$

$L = \{01, 0011, 000111 \ldots\}$.

Let $w = \underbrace{001}_{x}\underbrace{1}_{y}z$. $n = 3$

Case 1. $|xy| \leq n$ $\therefore$ ~~$1+1 = 2 \leq 3$~~ $2 \leq 3$ ✓

Case 2 $|y| \geq 1$ $1 \geq 1$ ✓

Case 3 $\neq xy^kz$. $\therefore k \geq 0$.

$\therefore$ Let $k = 2$ $\therefore xy^2z = 00(1)^2 1$

$= \underline{00\,1\,0111} \notin L$ Hence not regular

(2) Prove that $L = \{a^{n^2} / n \geq 1\}$ is not regular

$L = \{a^1, a^4, a^9, a^{16}, a^{25} \ldots\}$ $n = 2$.

$z = aaaa$ $|z| \geq 2$

Let $z = \underbrace{aa}_{x}\underbrace{a}_{y}\underbrace{a}_{z}$ $\therefore |xy| \leq 2$ $\therefore$ ~~$1+1 \leq 2$~~ $1+1 \leq 2$ ✓

$|y| \geq 1$ $1 \geq 1$ ✓

$z = xy^kz \in L$ for $k \geq 0$

$\therefore k = 2$ $aaaaaa = a^6 \notin L$ not RL

Problem 3: Show that $L = \{a^p / p \text{ is a prime number}\}$ is not regular.

$L = \{a, aa, aaa, aaaaa \ldots\}$ Prime numbers 1, 2, 3, 5, 7, 11 ...

Assume pumping constant $n = 2$

Let $w = aaa$ $|w| \geq n$ $\therefore 3 \geq 2$ ✓

split $w = \underset{x}{a}\ \underset{y^k}{a}\ \underset{z}{a}$ $\therefore |xy| \leq n$ $\therefore 2 \leq 2$ ✓

$|y| \geq 1$ $1 \geq 1$ ✓

$w = xy^kz \in L$ for all $k \geq 0$

$\therefore w = aaa \in L \therefore k = 1$

$w = a(a)^2a$ $k = 2$

$aaaa \notin L$, not a RL

Problem 4: show that $L = \{ww^R / w \in (a+b)^*\}$ is not regular

$L = \{abba, aabbaa, baab \ldots\}$

Assume pumping constant $n = 3$.

Let $w = abba$ $|w| \geq n$ $4 \geq 3$ ✓

split $w = \underset{x}{a}\ \underset{y^k}{b}\ \underset{z}{ba}$ $\therefore |xy| \leq n$ $\therefore 2 \leq 3$

$|y| \geq 1$ $1 \geq 1$

$w = abba$ $k = 1$

$w = abbba \notin L$ $k = 2$, Not in R.L.

Problem 5: show that $L = \{ww / w \text{ in } (a+b)^*\}$ is not regular

$L = \{abab, baba, \ldots\}$.

$z = \underset{x}{ba}\ \underset{y^k}{ba}\ \underset{z}{}$

$z = b(ab)^2a$ $k = 2$

$bababa \notin L$, Not R.L

Problem 6. $L = \{a^n b^{2n} / n > 0\}$ is not regular.

$L = \{abb, aabbbb, aaabbbbbb \ldots\}$

$z = \underset{x}{a}\ \underset{y}{bb}\ \underset{z}{}$ $\therefore z = xy^kz = a(bb)^2b \therefore k = 2$

$= abbbb \notin L$

Problem 7. show that $L = \{ 0^{i^2} / i \geq 1 \}$ is not regular

$L = \{ 0, 0^4, 0^9, 0^{16}, 0^{25} \ldots \}$

$\because z = \underset{x}{0} \underset{y}{000} \underset{z}{0}$ $\quad \because z = xy^kz = 0(00)^20 \quad \because k = 2$

$= 000000 \notin L$, not R.L

Problem 8: show that following is not regular

$L = \{ 0^i 1^j / \gcd(i,j) = 1 \}$

$L = \{ 01, 011, 0111, 00111 \ldots \}$

$\because z = 00111$

$z = \underset{x}{00} \underset{y^k}{111} \underset{z}{} \quad \because k = 2 \quad = 0(0)^2 111$

$= 000111 \notin L$, not RL

gcd(i,j) = 1	
i	j
1	1
1	2
1	3
2	3
3	4
:	:

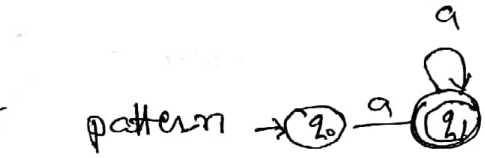
Problem 9 $L = \{ a^n b a^n / n \geq 0 \}$

$L = \{ b, aba, aabaa \ldots \}$

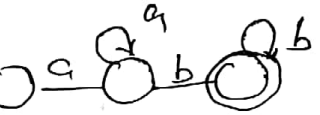
$z = \underset{x}{a} \underset{y}{b} \underset{z}{a} \quad \because k = 2 \quad a(b)^2a = abba \notin L$, Not RL.

Competative Exams:

Problem set:

1) $L = \{ a^n / n \geq 1 \}$ Regular pattern → 

2) $L = \{ a^{2n} / n \geq 1 \}$ "

3) $L = \{ a^n b^m / n, m \geq 1 \}$ Regular → 

4) $L = \{ a^n b^n / n \leq 10^{10^{10}} \}$ — Finite ⇒ Regular

5) $L = \{ a^i b^{i^2} / i, j \geq 1 \}$ - Not Regular - no pattern

6.

Conclusions: 1) Finite ?

2) Infinite ? → yes → genarate FA

3) Consisting "pattern" or not ?

if yes ⟹ R.L.

II Pumping Lemma for CFL (Context Free Language)

— Pumping Lemma used to prove that certain Languages are not Context Free Languages.

Let L(G) is CFL, then the following conditions must be satisfied.

i) every $z \in L(G)$ with $|z| \geq n$ / n is natural number and $z = uvwxy$ for same string.

ii) $|vx| \geq 1$

iii) $|vwx| \leq n$

iv) $uv^k wx^k y \in L$ for all $k \geq 0$

Problems: 1) Prove that $L = \{a^i b^i c^i / i \geq 1\}$ is not CFL.

Ans: i) Assume L = CFL and n is a natural number.

$\therefore L = \{abc, aabbcc, aaabbbccc, aaaabbbbcccc \ldots\}$ Let $n = 10$.

ii) Let $z = aaaabbbbcccc$ $\therefore |z| \geq n$

split z such that $z = uvwxy$.

case i) v and x contain same symbol or not same symbol

$z = \underbrace{aa}_{u}\underbrace{aa}_{v}\underbrace{bbbb}_{w}\underbrace{cc}_{x}\underbrace{cc}_{y}$

$\therefore |vx| \geq 1$ $\therefore 2+1 \geq 1$ ✓

$|vwx| \leq n$ $\therefore 2+4+1 = 7 \leq 10$ ✓

$uv^k wx^k y \in L$ for all $k \geq 0$

$\therefore k=1$ aa aa bbbb cc cc $\in L$

$k=2$ aa aaaa bbbb cccc $\notin L \Rightarrow$

Not in CFL.

Problem 2 Show that $L = \{a^{n^2} / n \geq 1\}$ is not CFL

$\therefore L = \{a^1\ a^4\ a^9\ a^{16}\ a^{25} \ldots\}$.

Let $L = \{\ldots aaaaaaaaa \ldots\}$

$\therefore n = 3$

$\therefore$ Let $z = aaa\ aaa\ aaa$ $\quad \therefore |z| \geq 3$.

Split into $z = uvwxy$

$\underline{aa}\ \underline{a}\ \underline{a}\ \underline{a}\ \underline{aaa}$ $\quad \therefore |vx| = 2 \geq 1$ ✓

$u\ v\ w\ x\ y$ $\quad |vwx| = 1+1+1 \leq 3$ ✓

$z = uv^k wx^k y \in L$ for $k \geq 0$.

$\therefore k = 2$ $\quad aaa\ (a)^2 a\ (a)^2 aaa$

$aaa\ aaa\ aaa\ aaaa = a^{11} \notin L$ Not CFL

Problem 3

Show that following Language is not CFL

$L = \{a^{n!} / n \geq 0\}$.

$L = \{a, aa, aaaaaa, a^{24} \ldots\}$.

Let $n = 5$.

Let $z = aaaaaa$ $\quad |z| \geq n \therefore 6 \geq 5$ ✓

split into $z = uv^k wx^k y \in L$ for all $k \geq 0$

$z = \underline{a}\ \underline{a}\ \underline{a}\ \underline{a}\ \underline{a}\ \underline{a}$ $\quad |vx| \geq 1 \therefore 2 \geq 1$ ✓

$u\ v\ w\ x\ y$ $\quad |vwx| \leq n \therefore 3 \leq 5$ ✓

Let $k=1$ $\quad z = aaaaaa \in L$ ✓

$k=2$ $\quad z = a\ (a)^2\ a\ (a)^2\ aa$.

$aaaaaaaa = a^8 \notin L$ Hence not in CFL

Problem 4: Show that $L = \{ 0^m 1^m 2^n / m \le n \le 2n \}$ is not CFL

$L = \{ 01\,22, 0011222, 00112222, \cdots \}$ $m=1,2,2$ $n=2,3,4$

Let pumping constant $n = 4$

Let $z = 0011222$ $|z| \ge n$ $\therefore 7 \ge 4$ ✓

split $z = \underline{0}\,\underline{0}\,\underline{11}\,\underline{222}$ ($u\ v\ w\ x\ y$) $\therefore |vx| \ge 1$ $2 \ge 1$ ✓

$|vwx| \le n$ $4 \le 4$ ✓

$\therefore k=1$ $z = 0011222 \in L$ ✓

$k=2$ $z = 0(0)^2 11 (2)^2 22$

$= 00011222 \notin L$, Hence not in CFL

Problem 5: Show that $L = \{ a^i b^j / i \le j^2 \}$ is not CFL

$L = \{ ab, abb, aabb, aaabb, aaaabb \cdots \}$ $j=1$ $i=1$

$\{ j=2$ $i=1,2,3,4$

Let $n = 4$,

$z = aaabb$

$z = \underline{a}\,\underline{a}\,\underline{a}\,\underline{b}\,\underline{b}$ ($u\ v\ w\ x\ y$) $|vx| \ge 1$ ✓

$|vwx| \le n$ $3 \le 4$ ✓

$k=1$ $z = aaabb \in L$ ✓

$k=2$ $z = a(a)^2 a(b)^2 b$

$= aaaabbb \notin L$, not in CFL

Problem 6 show that following language is not context-free

$L = \{ a^p / p$ is prime number $\}$ $P = 1,2,3,5,7, \cdots$

$L = \{ a, a^2, a^3, a^5, a^7, a^{11}, \cdots \}$

Assume pumping constant $n = 5$

choose string $z = aaaaaaa$ $|z| \ge n$ $7 \ge 5$ ✓

split into $z = uv^k wx^k y$. $\in L$ for $k \ge 0$

$z = \underline{aa}\,\underline{a}\,\underline{aa}\,\underline{aa}$ ($u\ v\ wx\ y$)

$\therefore k=1$ $\therefore z = uv^1 wx^1 y = aa(a)^1 a(a)^1 aa \in L$

$k=2$ $z = uv^2 wx^2 y = aa(a)^2 a(a)^2 aa \notin L$

$k=0$ $z = uv^0 wx^0 y = aa(a)^0 a(a)^0 aa \in L$

Not CFL

— Closure properties of CFL

1) CFL are closed for CFG. under Union, Concatenation, Kleen closure
2. CFL closed under Intersection.
3. CFL closed under complementation
4. CFL " " Substitution
5. CFL closed under homomorphism
6. CFL closed under Inverse homomorphism
7. If ~~CFL~~ L is CFL and R Regular set, Then $L \cap R$ is a CFL.

— Closure properties of RL

(Let L_1, L_2 Regular Languages) The following are regular

1) R.L is closed under
   (i) union, concatenation, kleen closure
   (ii) Intersection $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ ($L_1 \cup L_2$, $L_1 L_2$, L_1^*)
   (iii) complementation $\overline{L_1}$
   (iv) Substitution
   (v) Homomorphism.
   (vi) Inverse Homomorphism
   (vii) Difference. $L_1 - L_2 = L_1 \cap \overline{L_2}$

Turing Machine

- A mathematical model is called Turing machine

- Problem is:

  Let Real world problem.

  ~~We~~ can we solve above problem with computer

  ⇓

  Need mathematical model.

  (Turing machine).

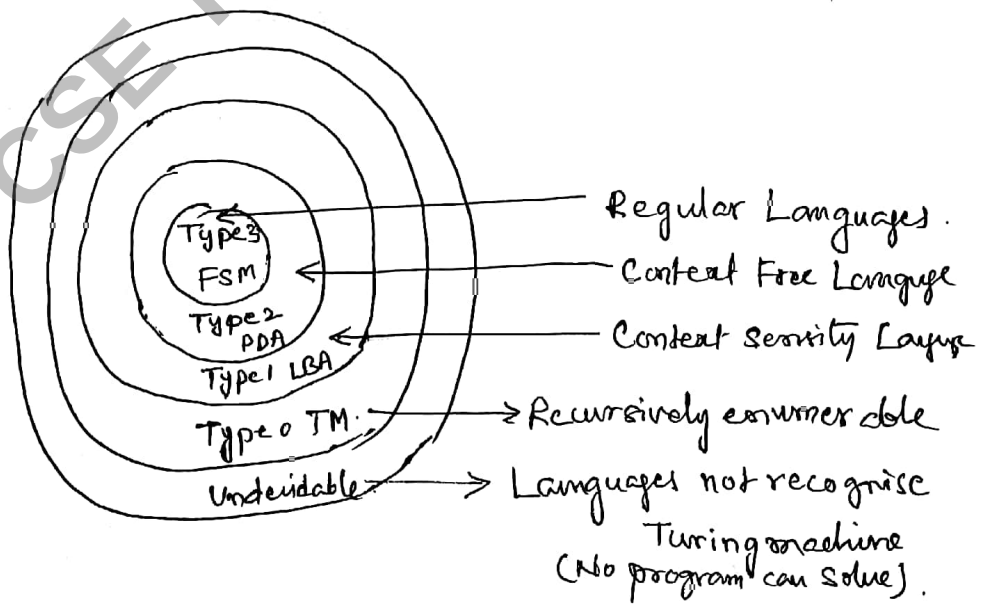
  - Know about limitations of computer
  - Convert physical machine to mathematical model.

Real problems 1) Find good sentences from this text

2) Find power of computer

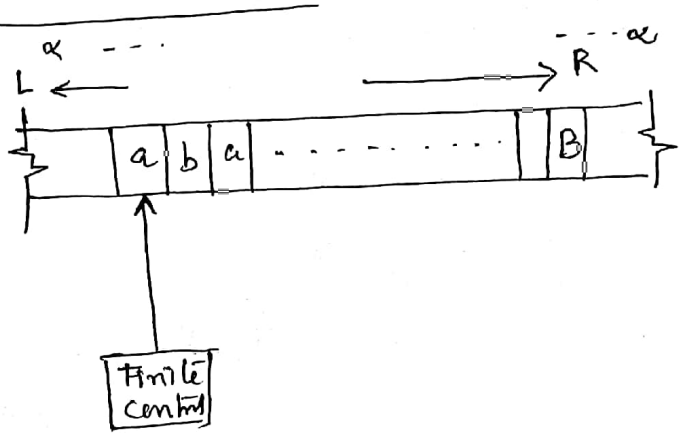
3) Find ambiguity of CFL.

- Turing machine is the most powerful mathematical model which define abstract computer.



- Developed in 1936 by Alan Turing.

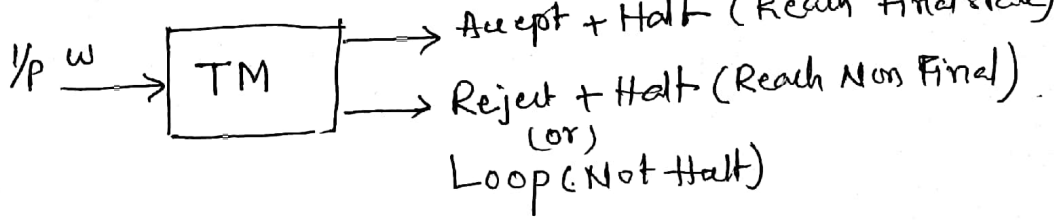
Turing Machine Model



- Consist of Read/write '∝' length tape towards Left and Right.
- It recognise Recursive Enumerable Languages (unrestricted languages). type 0 languages
- $\alpha \rightarrow \beta$: $\alpha, \beta \in \Sigma^*$
- Operations :- - Read - move Left
  - write - move Right
  - change state
  - Halt

Applications:
1. Language recognizer
2. Language generator
3. Language evaluator
4. Language Decider
5. * To prove Halting problem is undecidable
6. Mathematical model for modern computer.

Acceptence:



Type '0' grammar

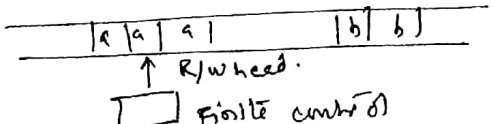
UNIT IV

TURING MACHINE :- It recognises Unrestricted grammar (i.e $\alpha \to \beta$, $aAB \to a$)

→ TM is capable of performing any calculation which can performed by any computing machine.

→ TM is an automation whose temporary storage is tape.

Tape devided into cells which store one symbol.



→ In one move, the machine examines the present symbol under R/w head on the tape and present state of automation to determine

(i) a new symbol to be written on the tape in cell under R/w head.

(ii) Moves its head left or right one cell.

(iii) change state.

T.M. Notation: defined as a 7-tuple notation

$M = (Q, \Sigma, \tau, \delta, q_0, B, F)$

Q = Finite set of states

τ = finite set of tape symbols

B = Blank symbol ($B \in \tau$)

q_0 = Starting state

Σ = subset of τ, not including B is set of input symbol.

$F \subseteq Q$ is set of final state

δ is mapping function from $Q \times \tau \to Q \times \tau \times \{L, R\}$.

Acceptability : Let $TM = (Q, \Sigma, \tau, \delta, q_0, B, F)$

A string $w \in \Sigma^*$ is said be accepted by if $q_0 w \vdash \alpha_1 p \alpha_2$ for some $p \in F$ $\alpha_1, \alpha_2 \in \tau^*$..

Instantaneous Description: (ID):-

We denote ID of the TM by $\alpha_1 q \alpha_2$ where $\alpha_1, \alpha_2 \in \tau^*$

$q_4 a_3 a_2$ ← L α_1 ; q_2 ↑ q_2 ; $a_5 a_6$ → α_2 R

Differences between Turing Machine and Push Down Automata

PDA	TM
1. PDA is non deterministic finite Automaton coupled with a stack, that can be used to store string.	1. TM is abstract computing machine with the power of both real computers and of other mathematical definitions.
2. Stack used LIFO principle	2. TM consists of finite-state control and an infinite tape divided into cell
3. PDA choose its next move based on its current state, next I/p symbol, and symbol at the top of stack	3. TM makes move based on its current state and the tape symbol at the cell scanned by the tape head.
4. PDA accepts by empty stack, final state	4. TM accepts its I/p if it ever enters an accepting state.
5. ID consist (q, w, γ)	5. ID of TM describes current configuration of a TM by finite length string.
6. PDA accepts CFL (Type 2) $A \to \alpha \quad \therefore \alpha \in (V \cup T)^*$	6. TM accepts Recursively Enumerable (RE) languages. (Type 0) Unrestricted grammar $\alpha \to \beta \quad \alpha, \beta \in V^*$
7. A PDA lies strictly in b/w regular languages and CFL's	7. RE Languages.
(8) Defined $M = (Q, \Sigma, \tau, \delta, q_0, z_0, F)$	(8) $M = (Q, \Sigma, \tau, \delta, q_0, B, F)$
(9) Mapping: $Q \times (\Sigma \cup \{\in\}) \times \tau \to Q \times \tau^*$	(9) $Q \times \tau \to Q \times \tau \times \{L, R\}$
(10) Acceptance: (i) Empty stack (ii) Final	(10) Reach Final state
(11) Instantaneous Description (ID) (q, w, γ) state ↑ I/p tape ↖ Stack content	(11) ID: $(\alpha_1 q \alpha_2) / (q \alpha_1 \alpha_2)$ $\therefore \alpha_1, \alpha_2 \in \tau^*$ $q \in Q$

DESIGN OF T.M.:-

Basic guide lines: 1, scanning a symbol by R/W head is to 'know' what to do in the future. The machine must remember the past symbols scanned.

2, change the states only when there is a change in the movement of R/W head.

problems: i) 1 Design TM accepted $L = \{0^n 1^n / n \geq 1\}$ ⇒ equal 1's and equal 0's

2, " " " $L = \{$ all strings consists with odd no. of 1's $\}$

3, " " " $L = \{1^n 2^n 3^n / n \geq 1\}$

4, " " " $L = \{0^n 1^n 0^n / n \geq 1\}$

ii) 5, " " " $L = \{ww^R / w \in (a+b)^*\}$ ⇒ even polyndrome

6, " " " $L = \{wcw^R / w \in (a+b)^*\}$ odd polyndrome.

7, " " " $L = 00^*$

8, " " " $L = 0(0+1)^*$

9, " " " to implement total recursive function multiplication

10, " " that computes $p+q$ for given two positive integers.

11, " " that copies a string of 0's performs the following computation.
$q_0 w \overset{*}{\Rightarrow} q_f ww$, $w \in \{0\}^*$

iii) 12, " " to compute proper substraction $m \dot{-} n$ define to be $m-n$ for $m \geq n$ and zero to $m < n$

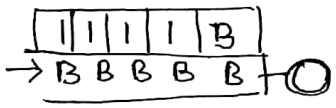
13, " " to compute 1's compliment of given binary number.

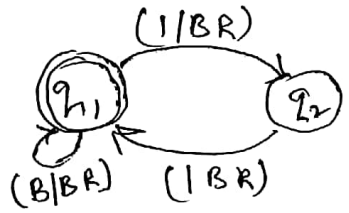
Problems on Turing machine:

(1) Design TM to accept all strings consisting of even no. of 1's.

String = {11, 1111, 111111 ....}

Logic:





transition table: δ

I/P = Σ^*

Present state	1	B	⊔
q_1	(q_2, BR)	(q_1, B, R) ← Final state	
q_2	(q_1, BR)	—	

$\delta = \delta(q_1, 1) = (q_2, BR)$
$\delta(q_2, 1) = (q_1, BR)$
$\delta(q_1, B) = (q_1, BR)$ ⇒

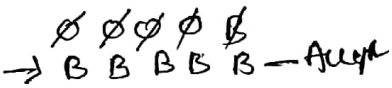
Check: Example 11

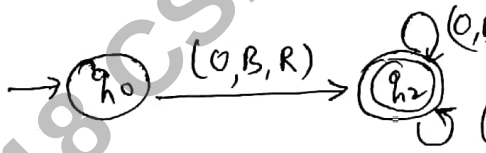
$q_1 11 \vdash b q_2 1 \vdash BB q_1 B$ accepted.

(2) Design TM, that accepts language denoted by RE = 00^*

$L = \{0, 00, 0000, 00000 \ldots\}$

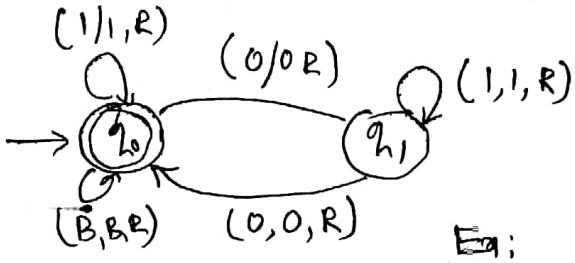
Logic



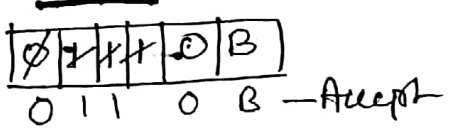


(3) Design TM any no. of 1's and even no. of 0's. over $\Sigma = \{0, 1\}$.

$\therefore L = \{100, 01110, 1011110 \ldots\}$.



Logic



Eg: $q_0 100B \vdash 1q_0 00B \vdash 10q_1 0B$

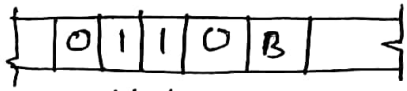
$\vdash 100q_1 B \vdash 100q_1 B$. ← Final state Accepted

(4) Design a T.M which recognise the Language

$L = 01^*0$

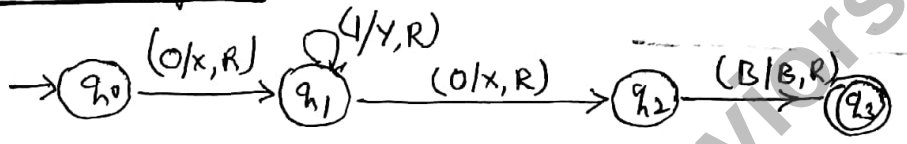
$L = \{010, 0110, 01110 \ldots\}$

Logic:



→ X Y Y X ↑ Accept.

Transition Diagram:



Transition table

States	0	1	x	Y	B
→ q_0	(q_1, X, R)	–	–	–	–
q_1	(q_2, X, R)	(q_1, Y, R)	–	–	–
q_2					(q_3, B, R)
q_3	–	–	–	–	–

Transition Function (δ)

$\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_1, 0) = (q_2, X, R)$

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

$\delta(q_2, B) = (q_3, B, R)$ Accepted.

Check:- $q_0 0110B \vdash Xq_1 110B \vdash XYq_1 10B \vdash XYYq_2 0B$

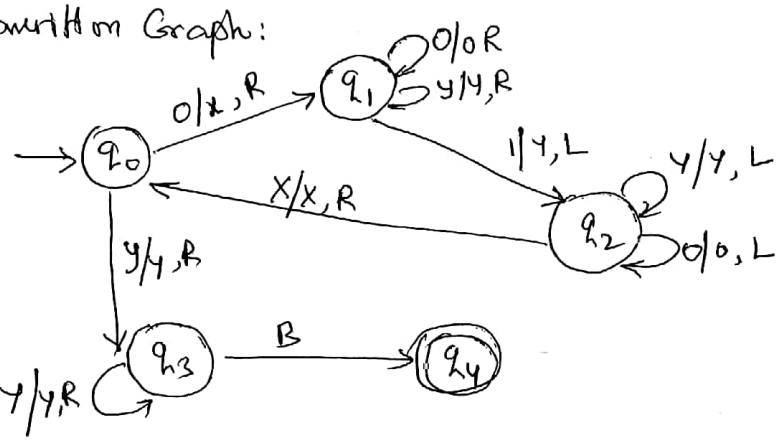
$\vdash XYYXq_3 B \vdash XYYXBq_3$ ← Accepted + Halt.

Problem 1 Design Turing machine accept the Language

$$L = \{0^n 1^n / n \geq 1\}$$

Let $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{0,1\}$, $\tau = \{0,1,X,Y,B\}$ and $F = \{q_4\}$.

Transition Graph:



→R ←L

0 0 0 X 1 1 B
X 0 0 Y X 1 B
X 0 Y Y 1 B
X Y Y 1 B
Y Y Y B

Accepted

The function of δ.

State	0	1	symbol X	Y	B
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, R)	—
q_3	—	—	—	(q_3, Y, R)	(q_4, B, R)
q_4	—	—	—	—	—

Eg: $q_0 0011$

$\vdash X q_1 011$
$\vdash X 0 q_1 11$
$\vdash X q_2 0 Y 1$
$\vdash q_2 X 0 Y 1$
$\vdash X q_0 0 Y 1$
$\vdash X X q_1 Y 1$
$\vdash X X Y q_1 1$
$\vdash X X q_2 Y Y$
$\vdash X q_2 X Y Y$
$\vdash X X q_2 Y Y$
$\vdash X X Y q_3 Y$
$\vdash X X Y Y q_3$
$\vdash X X Y Y B q_4$

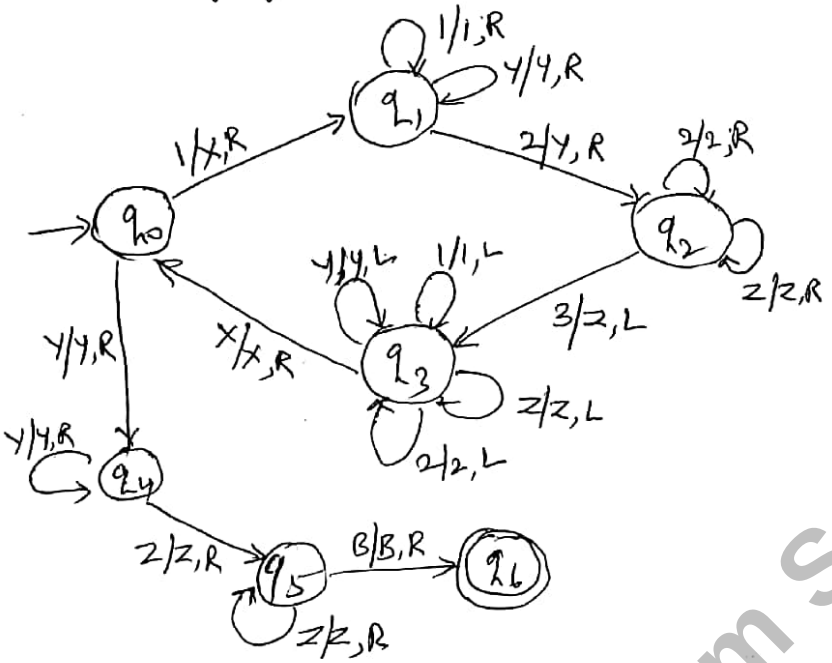
Turing machine has reached final state q_4, 0011 is accepted

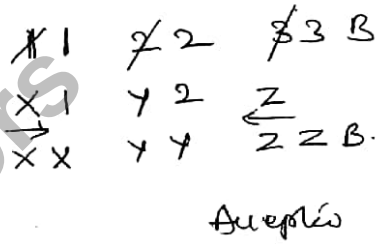
Problem 2. Design Turing Machine to recognise the language $\{1^n 2^n 3^n / n \geq 1\}$.

Let $Q = \{q_0 q_1 q_2 q_3 q_4 q_5 q_6\}$ $\Sigma = \{1, 2, 3\}$ $\Gamma = \{1, 2, 3, X, Y, Z, B\}$

and $F = \{q_6\}$

The strings of this language are 123, 112233, 111, 222, 333.





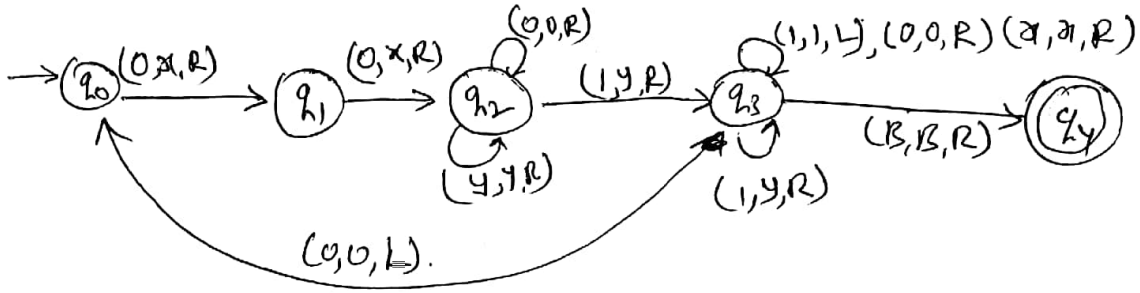
The function δ is.

State	1	2	3	X	Y	Z	B.
→ q_0	(q_1, X, R)	—	—	—	$(q_4 Y, R)$	—	—
q_1	$(q_1, 1, R)$	$(q_2 Y R)$	—	—	$(q_1, Y R)$	—	—
q_2	—	$(q_2 2 R)$	$(q_3 Z L)$	—	—	$(q_2 Z R)$	—
q_3	$(q_3 1, L)$	$(q_3 2, L)$	—	$(q_0 X R)$	$(q_3 Y, L)$	$(q_3 Z L)$	—
q_4	—	—	—	—	$(q_4 Y R)$	$(q_5 Z R)$	—
q_5	—	—	—	—	—	$(q_5 Z R)$	$(q_6 B R)$
(*) q_6	—	—	—	—	—	—	—

Problem 3: Design a TM that recognises the lel- $\{0^{2n}1^n / n \geq 0\}$ Ex: 000011

The tuple representation of TM is

$TM = (\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, \{0,1,x,y,B\}, \delta, q_0, B, \{q_4\})$



State	0	1	x	y	B
→ q_0	(q_1, x, R)	–	–	–	–
q_1	(q_2, x, R)	–	–	–	–
q_2	$(q_2, 0, R)$	(q_3, y, R)	–	(q_2, y, R)	–
q_3	$\{(q_3, 0, L), (q_0, 0, R)\}$	$\{(q_3, 1, L), (q_3, y, R)\}$	(q_3, x, R)	(q_3, y, L)	(q_3, B, R)
(q_4)	–	–	–	–	–

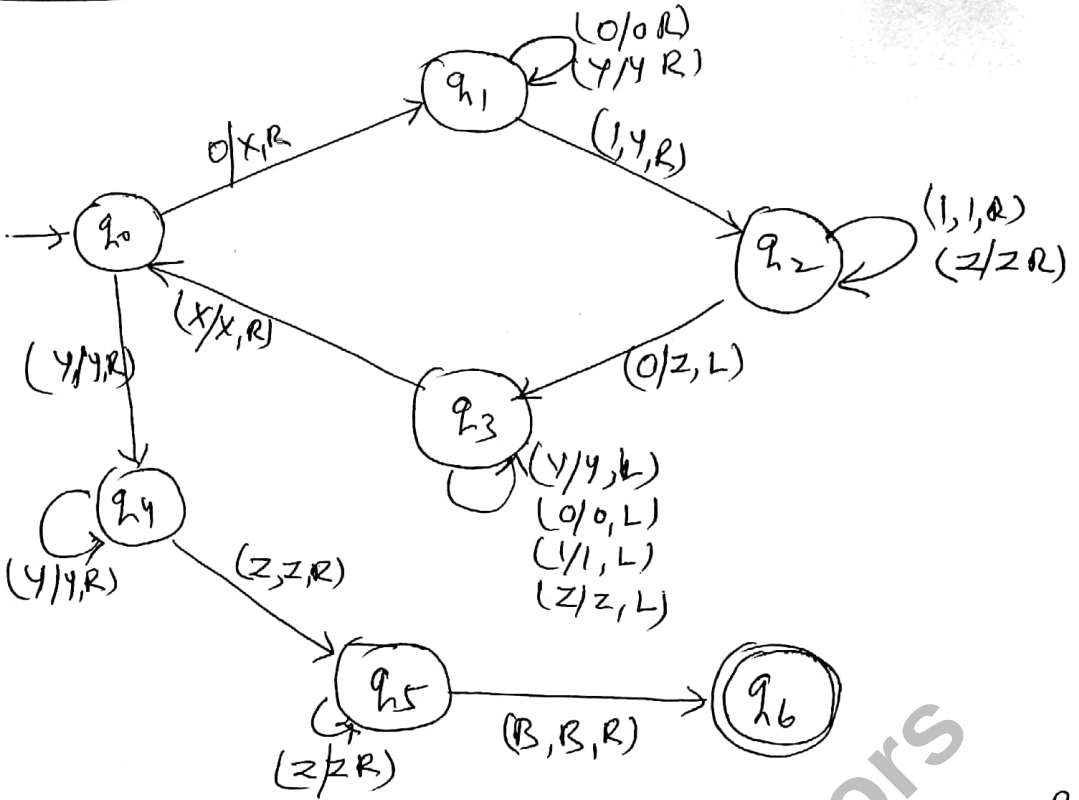
Problem 4: Design Turing machine to recognise the language

$L = \{0^n 1^n 0^n / n \geq 1\}$ Eg: 001100

Let $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ $\Sigma = \{0,1\}$ $\Gamma = \{0,1,X,Y,Z,B\}$

$F = \{q_6\}$.

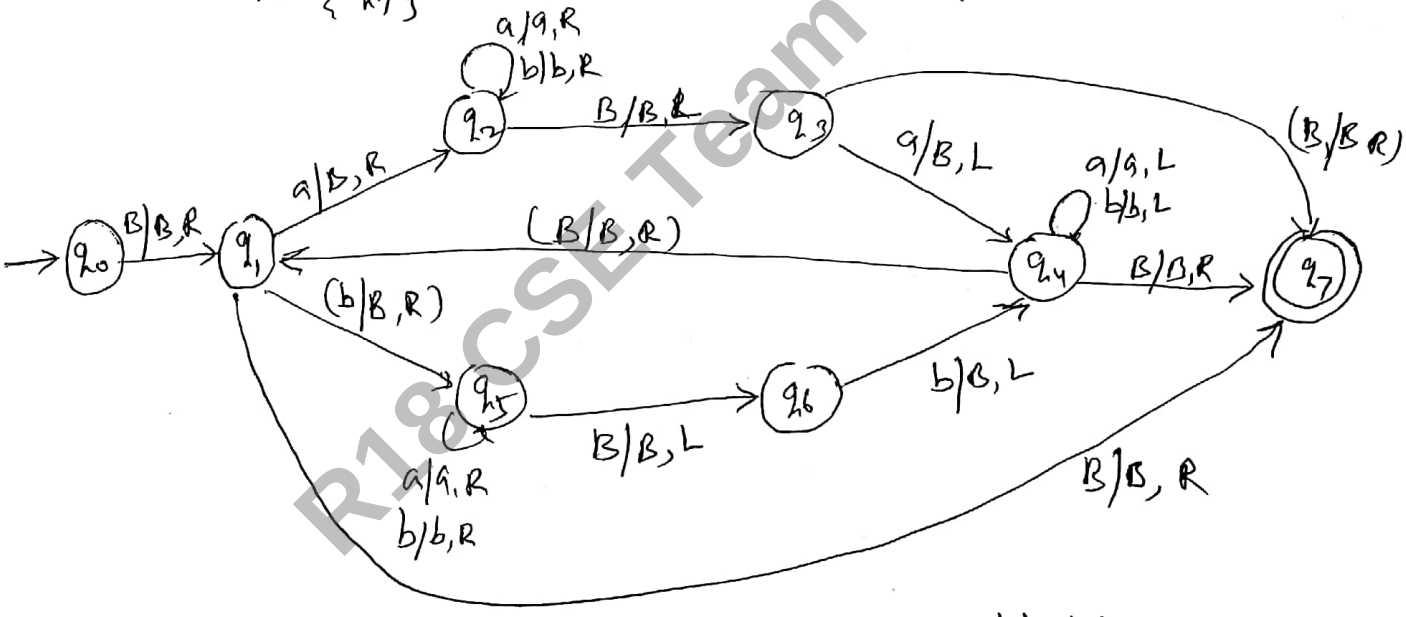
State	0	1	X	Y	Z	B
→ q_0	(q_1, X, R)	–	–	(q_4, Y, R)	–	–
q_1	$(q_1, 0, R)$	(q_2, Y, R)	–	(q_1, Y, R)	–	–
q_2	(q_3, Z, L)	$(q_2, 1, R)$	–	–	(q_2, Z, R)	–
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, X, R)	(q_3, Y, L)	(q_3, Z, L)	–
q_4	–	–	–	(q_4, Y, R)	(q_5, Z, R)	–
q_5	–	–	–	–	(q_5, Z, R)	(q_6, B, R)
(q_6)	–	–	–	–	–	–



Problem 5. Design TM that recognises any palindrome ($ww^R / w \in (a+b)^*$)

Let $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$ $\Sigma = \{a, b\}$ $\Gamma = \{a, b, B\}$

$F = \{q_7\}$



abbbba

BbbbbB

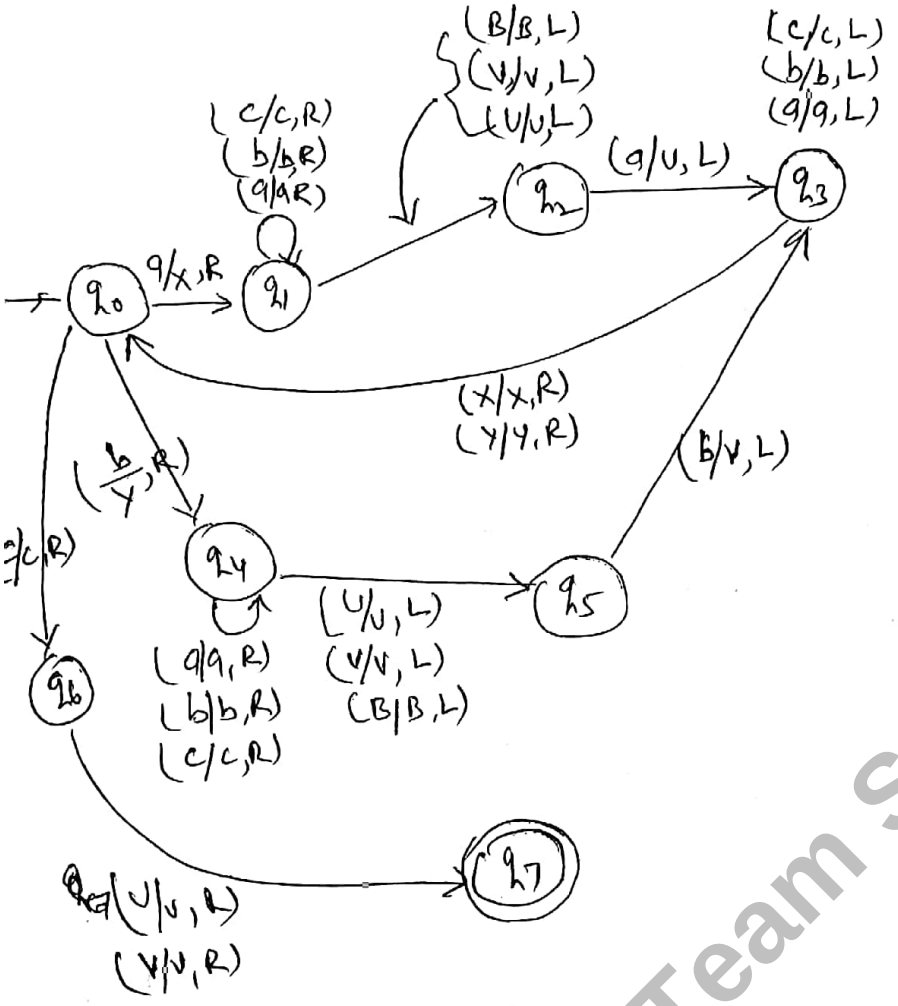
BbbbB

BbbB

BbB

BBB Final state

8. Design Turing machine to accepts the language $L = \{wcw^R / w \in (a+b)^*\}$.



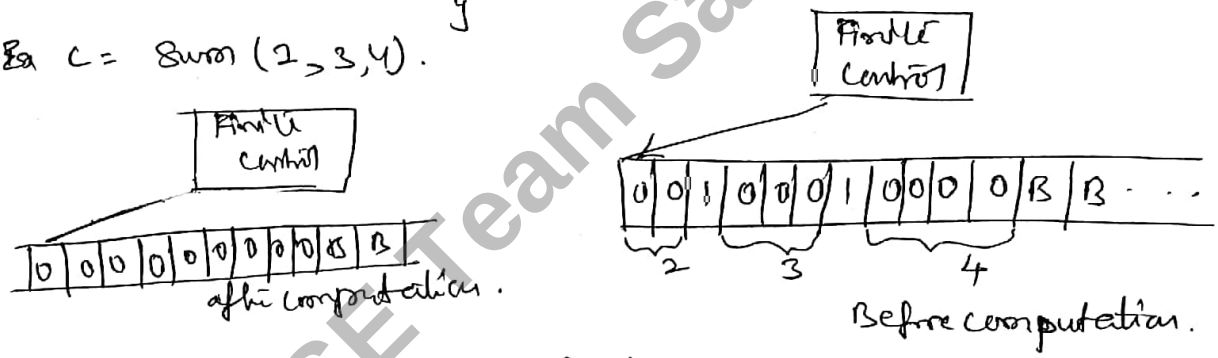
a b b C b b a B
X b b C b b a B
X b b C b b U B
X Y b C b b U B
X Y b C b V U B
X Y Y C b V U B
X Y Y C V V U B accepted.

Turing machines Computable Functions:

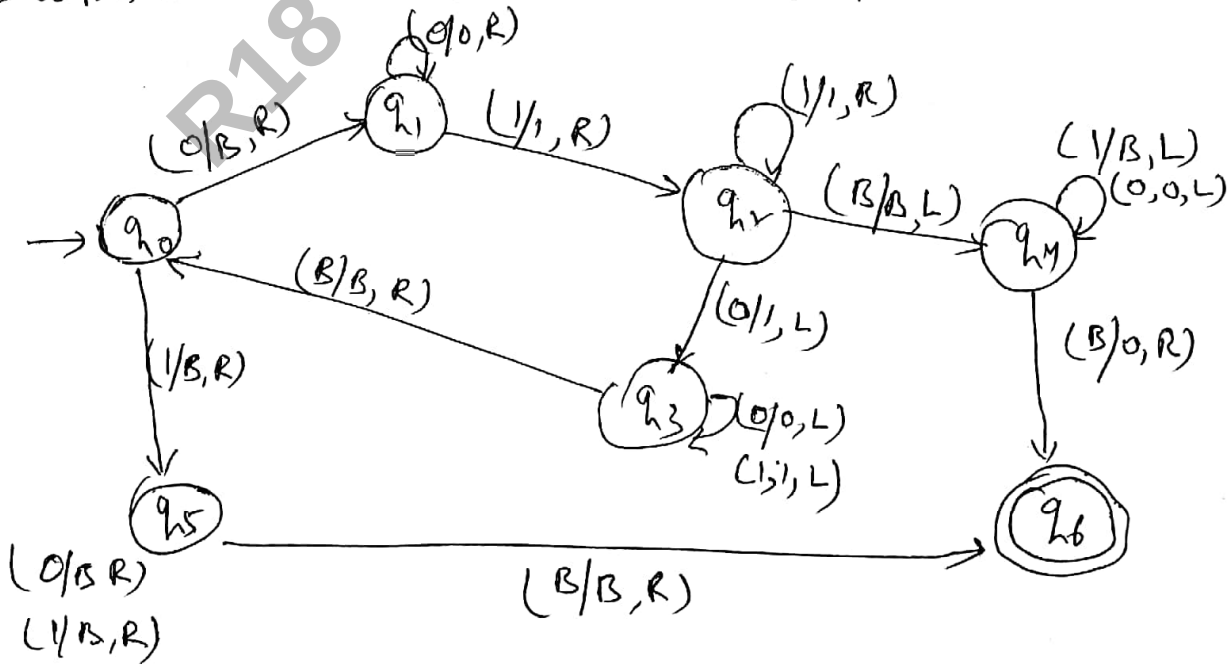
- TM is a language acceptor which checks string 'x' $\in$ L or not.
- It also perform computations of functions from integers to integer.
- Integers is represented in unary, an integer $i \geq 0$ is represented by string 0^i Ex 2 represented 0^2.
- If function have k arguments $i_1, i_2 \ldots i_k$ then these integers are initially placed on the tape separated by 1's i.e $0^{i_1} 1 \; 0^{i_2} 1 \ldots\ldots 1 0^{i_k}$.
- Ex: function in 'C'

```
Int sum(int i, int j, int k)
{
    s = i+j+k;
    return(s);
}
```

Ex c = sum(2,3,4).



Problem 1 Construct Turing Machine to find proper subtraction m−n is defined to be m−n for $m \geq n$, and zero for $m < n$.



- Assume $0^m 1 0^n$ on state and machine halts with 0^{m-n} on its tape.
- Turing machine M' its leading 0 by blank then searches right for a 1 followed by a zero and changes zero to one. M moves left until it encounters a blank and then repeats the cycle.

consider TM $M = (\{q_0 q_1, \ldots q_6\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \phi)$.

δ defined as: $\delta(q_0,0) = (q_1, B, R)$ — Replace 0 by B.

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

$\delta(q_1,1) = (q_2, 1, R)$ — search right looking first 1

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

$\delta(q_2,0) = (q_3, 1, L)$ — looking right part 1's until encounter 0 change 0 to 1

$\delta(q_3,0) = (q_3, 0, L)$

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

$\delta(q_3,B) = (q_0, B, R)$ — move left to a blank. Enter state q_0 to repeat the cycle.

$\delta(q_2,B) = (q_4, B, L)$

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

$\delta(q_4,0) = (q_4, 0, L)$

$\delta(q_4,B) = (q_0, 0, R)$

Ex:

ϕ 0 0 1 0 0

→ B 0 0 1 1

B 0 0 1 1

↳ B 0 1 1 1

B 0 1 1 1

↳ B 1 1 1 B

0 B B B

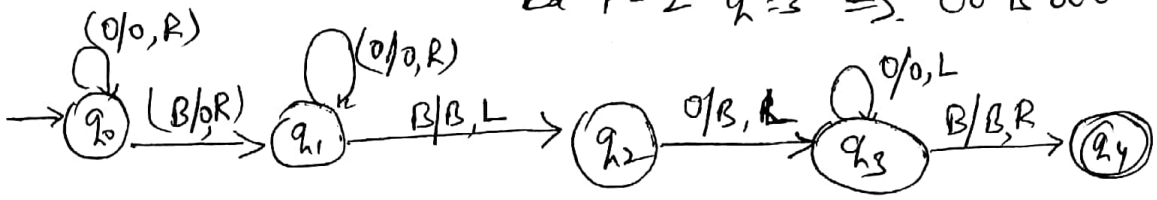
$m=3$, $n=2$, $m-n=1$

Accepts $0^1 = 1$

Problem: Design Turing machine to add two given integers (4-6)

The language add two integers $L = \{ 0^p B 0^q / p \geq 1 \text{ and } q \geq 1 \}$.

Ex $p = 2$ $q = 3$ $\Rightarrow$ 00 B 000



00B000B

→ 000 00Ø B

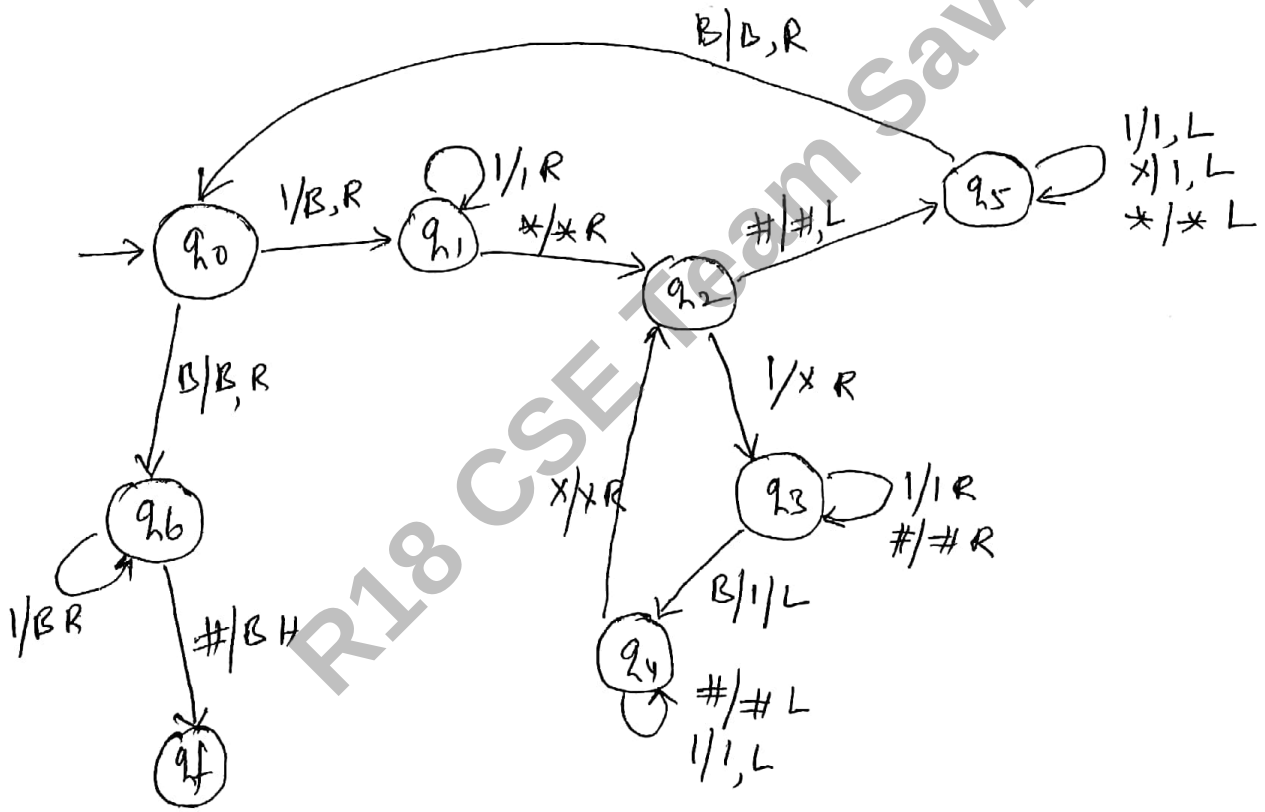
Ø 000 00B ←

B↳ =

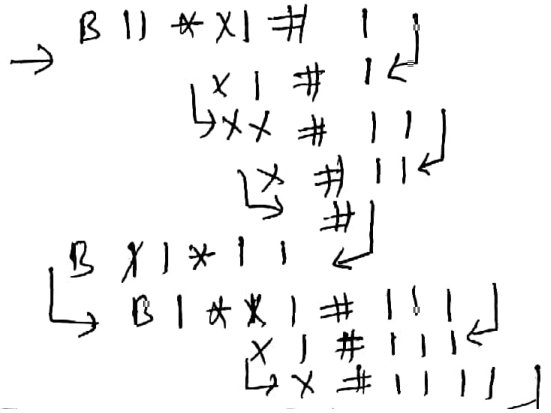
q_4

Problem: Design TM to implement the total recursive function multiplication

Language $L = 0^m 1 0^n$ (multiplication of given integers)

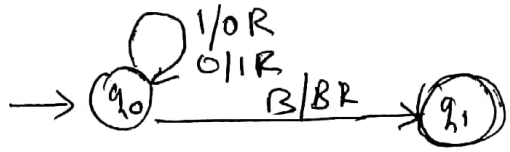


Ex: 11 *11 # BBBBBBB.

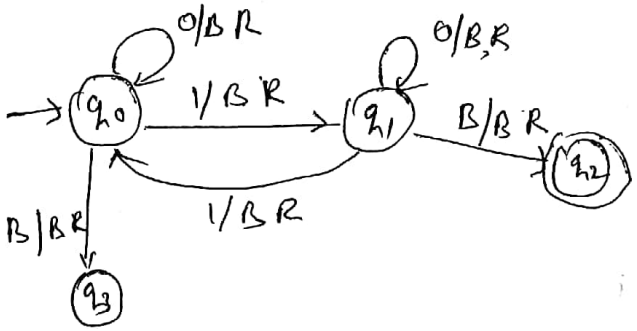


Problem : Design turing machine that computes ones complement of a binary number

Ans: complement of 100110 is 011001



problem: Design Turing machine to recognise all strings if string contains odd number of 1's over {0,1}



X 0 1 1
B B B B Accepts

UNIT - V

Chomsky Hierarchy of Languages :-

Types of Grammer

Prescriptive
- Authorative norms

Descriptive
- Actual usage (No norms)

Formal
- Defined grammer (CFG)

Generative
- Natural Language expressions

Formal →

Generative
- Formulate algorithm, that generates valid string.
- It describes how to write a language

Analytic
- A set of rules to reduce input string to boolean result that indicates validity of string.
- It describe how to read/parser it.

Closure properties of Languages :

Regular Language :- Union, concatenation, Kleen star, intersection, complement, homomorphism Inverse homomorphism.

Context Free :- concatenation, Kleen star, Union

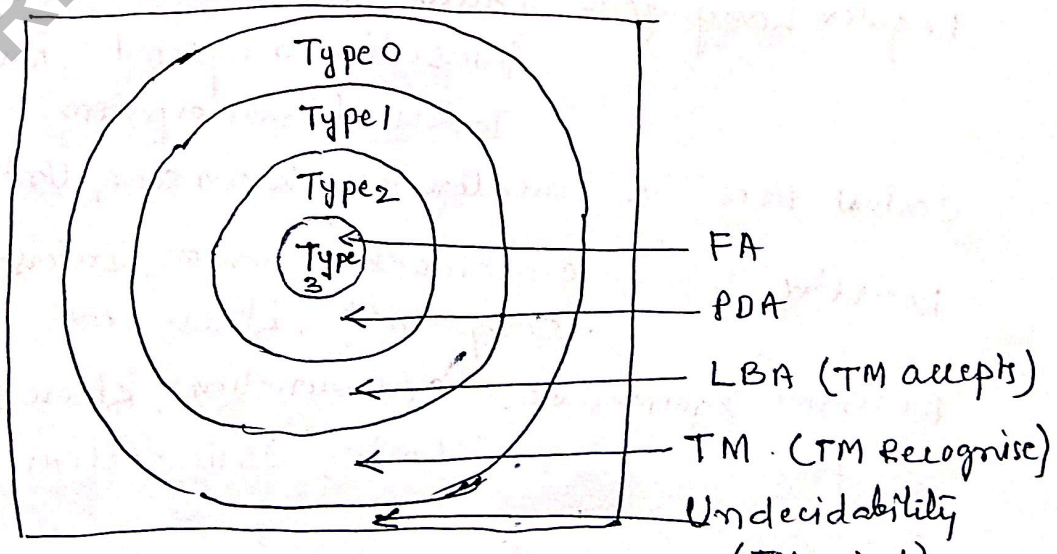
Recursive - concatenation, Union, Intersection, complement, Kleen star

Recursive Enumerable :- concatenation, kleen star, Union, Intersection.

Hierarchy of Languages :-

Type	Grammer	Language	Recognizer	Rules.
Type 0	Unrestricted $G=(V,T,P,S)$	Recursive Enumerable	Turing machine (TM)	$\alpha \to \beta$ $\alpha, \beta \in (V \cup T)^*$
Type 1	Context Sensitive $G=(V,T,P,S)$	Context Sensitive.	Linear Bounded Automata (LBA)	$\alpha \to \beta$ $\lvert\alpha\rvert \leq \lvert\beta\rvert$ $\alpha, \beta \in (V \cup T)^*$
Type 2	Context Free Grammar $G=(V,T,P,S)$	context free Language	Push Down Automata (PDA)	$A \to \alpha$ $\alpha \in (V \cup T)^*$ $A \to \epsilon$ $A \in V$
Type 3	Regular Language	Regular Language	Finite Automata (FA)	$A \to \alpha B$, $A \to \alpha$ $A, B \in V$ $\alpha \in T^*$

Type 3 $\subseteq$ Type 2 $\subseteq$ Type $\subseteq$ Type 1 $\subseteq$ Type 0.



UNIT-V

Church's Hypothesis :-

Question: what does 'Computable' mean ?

Church (1903-1995)
- Lambda calculus

A. Turing (1912-1954)
- Turing machine

- Turing machine and Lambda calculus are equivalent in power
- Algorithmically computable is computable by T.M.

Problems → Decidable (Recognise by TM)
Problems → Undecidable (Not Recognise by TM).

→ A functional on natural number is computable if and only if it is computable by Turing machine.

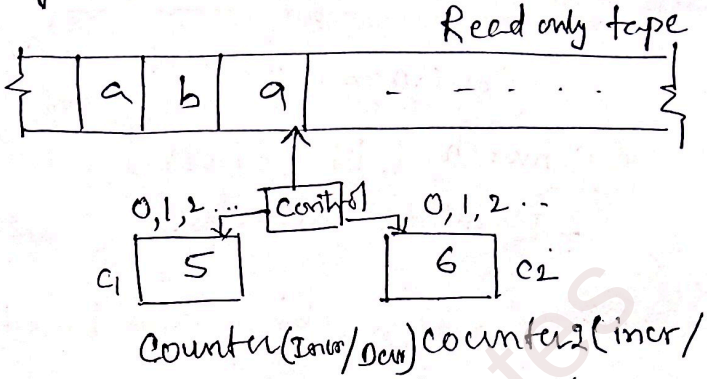
(i) Any Thing that can be done by current digital computer can also be done by T.M.

(ii) Currently, there is no problem which can be solved by a digital computer and cannot be solvable by T.M

(iii) Many mathematical models are suggested but, no one of them is more powerful than the T.M.

Counter Machine :-

- It have Read only tape with counters.



Counter(Inc/Dec) counter2(inc/Dec)

- Each counter can represent non-negative integers
- It have semi infinite tape
- The counter may incremented or decremented.
- If it have only one counter, it will not achieve T.M. capacity hence it should be more than one counter, but more than two counters is not much efficient.
- operations:
  1. check counter is zero
  2. increment / Decrement by 1.
  3. Decrement by 1 provided counter is +ve.
- move of counter machine:

present state	I/p symbol	c_1	c_2	next state	operation c_1 c_2
q	a	0	0	p	

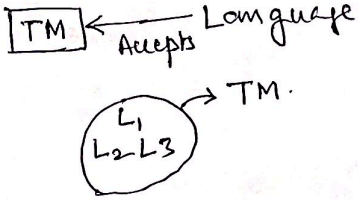
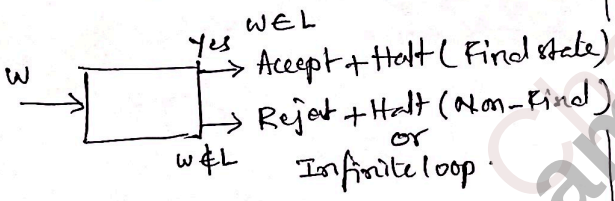
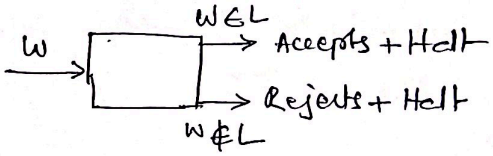
$\cdots(q, a, \underset{c_1}{\text{posive}}, \underset{c_2}{\text{zero}}$ next state c_1 c_2 p incre incre ~~decrement~~)

- These ~~are~~ can represent with multi-stack T.M i.e replace stack with counter.

Unrestricted Grammar

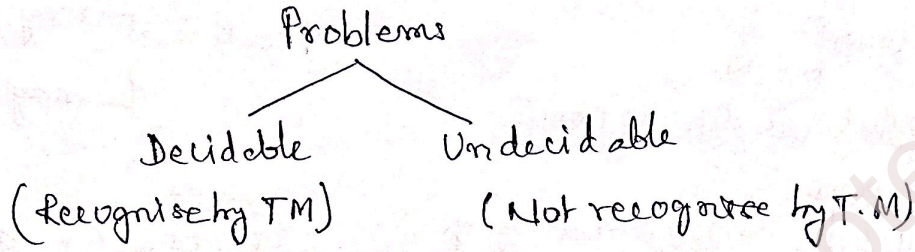
→ Recursively Enumerable, Recursive sets:

- Turing machine recognises Recursively Enumerable Language.
- T.M. have : (1) Halt & accept the I/p
  (2) Halt & reject the I/p
  (3) Never halt / Loop.

Recursively enumerable Language:	Recursive Languages
- Accept every string or not.	- T.M halt every string.
	- Accept & halt (Reach Final state) - Reject + halt (Reach Non-final state)
	- Recursive ⊂ RE.  
- Membership function undecidable.	- Membership function decidable
→ Properties - REL not closed under complementation - REL closed under union, Intersection, concatenation, kleen closure positive closure Reversal	- RL is closed under complementation - RL closed under Union, Intersection - concatenation - Kleen closure - positive closure - reversal

UNIT-V
Undecidability

8. A Language that is Not Recursively Enumerable :-



Problems that computer cannot solve:

1. program for print 'Hello world'

(i)
```
main()
{printf(hello world);
}
```

(ii) program with like $x^n + y^n = z^n$.
- given in if it finds 1 = prints hello world.
- never finds integers x, y, z satisfy the equation, then it continues searching never prints hello, world.

As per Church-Turing-Thesis :- Any thing computable can be computable by Turing machine.

Problems

Decidable problems (Solvable by TM)

- Solution: Yes / No
- Solution in definite time
- Example:
  - (i) L_1, L_2 are Regular language they are closed under Union, concatenation, kleen closure.
  - 2, Does FA accept R.L
- Algorithm will solve it

Undecidable problems (Unsolvable by TM)

Solution: may 'yes', may 'no'

Example

1, For CFG, G is ambigious.

2, For CFG L_1 and L_2 whether $L_1 \cap L_2$ is in CFL.?

- No Algorithm solve it.
- If decision problem

UNIT-V

Codes for Turing machine :-

We can represent TM $M=(Q,\{0,1\},\tau,\delta,q_1,B,F)$ in binary bits $\{0,1\}$.

Example:

$TM=\{(q_0,F),\{0,1\},\{0,1,B\},\delta,q_0,B,F\}$.

Now: δ = $\delta(q_0,0)=(q_0,0,R)$

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

$\delta(q_0,B)=(F,1,L)$

- the above Turing machine convert into binary bits.

States $q_0=1$ $\quad$ $\Sigma=$ $0=1$

$q_F/F=11$ $\quad$ $1=11$

$\tau=$ $0=1$ $\quad$ Directions $L=1$

$1=11$ $\quad$ $R=11$

$B=111$

- each one separated by '0' (each symbol separated).
- each transition separated by 00 (each term separated)

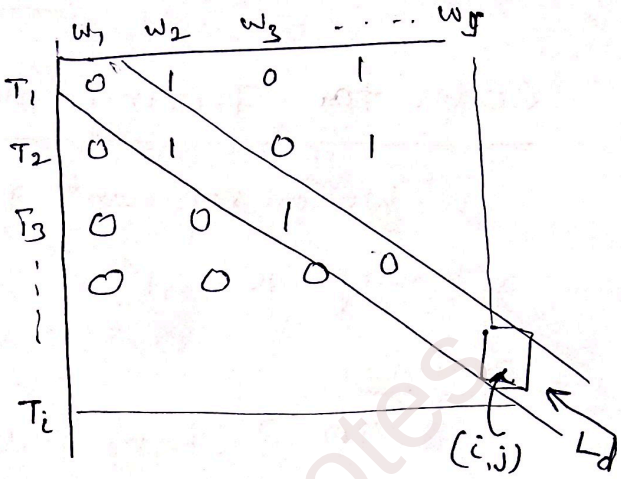
$\therefore TM=\{\underbrace{1011}_{Q}\ 00\ \underbrace{1011}_{\Sigma}\ 00\ \underbrace{10110111}_{\tau}\ 00\ \underbrace{1010101011}_{\delta_1}\ 00$

$\underbrace{1011010111011}_{\delta_2}\ 00\ \underbrace{1011101101101}_{\delta_3}\ 00\ \underbrace{1}_{q_0}\ 00\ \underbrace{111}_{B}\ 00\ \underbrace{11}_{F}\}$

Example: code the T.M.

$M=(\{q_1,\cdots q_6\},\{0,1\},\{0,1,B\},\delta,q_0,B,\{q_2\})$.

8. Diagnolization Language :-



If $(i,j)^{th} = 1$, T_i accepts w

If $(i,j)^{th} = 0$, T_i doesnot accept w

L_d = Diagnoligation of Language

$\therefore L_d$ = corresponding value = 0
(not accepts)

$\overline{L_d}$ = corresponding value = 1
(accepts string w).

$\therefore L_d = \{w / w = w_i$ for some i and $(i,j)^{th}$ is $= 0$
not accepted by $T_i\}$

$\overline{L_d} = \{w / w = w_i$ for some i and $(i,j)^{th}$ is $= 1$
accepted by $T_i\}$.

$\therefore$ Here L_d = is not Recursively Enumerable

$\overline{L_d}$ = is Recursively Enumerable but not Recursive set.

Universal Turing Machine (UTM):-

A UTM is given as input a string composed of two parts 1) Encoding program 2) a string that will be called data.

1) Encoding scheme:- provide information regarding M. The information includes no. of states used in M, no. of tape symbols, Input Alphabet, starting state, final state, end marker, blank symbol and transition function.

Ex: $Q = \{p, q, r\}$

tape symbols = $\{a, b\}$

I/p symbols = $\{a, b\}$ Initial state p, final state r.

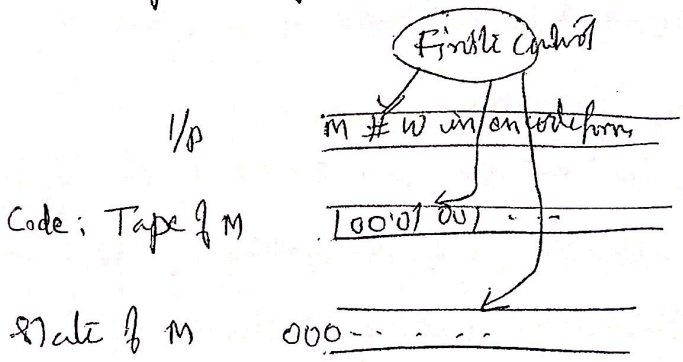
p encoded by 00, a encoded 00

q " by 000 b encoded 000

r " by 0000 end marker = #

blank = B

Description of M = $\underbrace{0^3}_{\text{no. of states}} 1 \underbrace{0^2}_{\text{tape symbols}} 1 \underbrace{0^2}_{\text{I/p symbols}} 1 \underbrace{00}_{p} 1 \underbrace{0000}_{r} 1 \underbrace{0}_{\text{blank symbol}} 1 0^6$

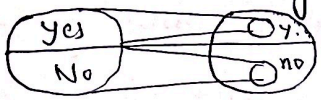


UNIT-5

Undecidable Problem about Turing machine :-

- Reduction :

It is a technique in which if a problem P_1 is reduced to a problem P_2 then any solution of P_2 solves P_1.



If we have an algorithm to convert an instance of a problem P_1 to an instance of a problem P_2 that have the same answer, then it is called as P_1 reduced to P_2.

Hence, if P_1 is not recursive $\rightarrow$ P_2 also not recursive

if P_1 is not recursively enumerable $\rightarrow$ P_2 also not recursively enumerable.

Theorem: If P_1 is reduced to P_2 then

(i) If P_1 is undecidable, then P_2 also undecidable

(ii) If P_1 is non-R.E, then P_2 also non-RE.

7. Turing machine that accept empty Language :

- These are undecidable problems, it involves reduction problem

Let Le = Language accept empty language (empty)

Lne = Language does not accept empty language (non-empty language)

- investigate Le, Lne languages.
- Le, Lne consist of binary strings = $\Sigma = (0+1)^*$

  If w is binary string then it represent T.M, M_i.
- If $L(M_i) = \phi$ then M_i does not accept I/p then w is in Le

  if $L(M_i) \neq \phi$ then w is in Lne.

Hence we can say that

$$Le = \{ M / L(M) = \phi \}$$
$$Lne = \{ M / L(M) \neq \phi \}$$

Both Le and Lne are the complement of each other.

8. Rice's theorem :-

- This theorem shows undecidability of infinite set of R.E languages / Language.
- A property of Language is a function.

$$P\{ \text{set of all languages} \} \rightarrow \{0,1\}$$

$P(L) = 1$ if associate -1 → Language satisfy property

$P(L) = 0$ if associate -0 → " does not satisfy.

- Non-trivial property of Languages of T.M, if exists TMs M_1 and M_2 such that

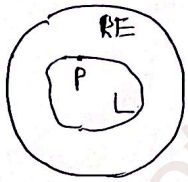
$$P(L(M_1)) = 1$$
$$P(L(M_2)) = 0$$

— Rice's theorem :-
(cont.).

Let P be a non-trivial property of languages of T.M (RE) then the language.

$$L_P = \{ \langle M \rangle \mid P(L(M)) = 1 \}$$ is undecidable.

if P = RE set or empty ⟶ trivial property
else non-trivial.



∴ L has all properties of P.
L → M (Language by M)
L = L(M) if L = L(M) ∴ $L_P = \{ \langle M \rangle / L(M)$ is in P. $\}$

Rice's theorem can be stated as follows:

Let 'P' be any non trivial property of regular expression languages then 'P' is undecidable.

By this theorem, we can determine, whether a property 'P' of regular expression is decidable or undecidable.

By using Rice's theorem, we can able to determine the following properties of the regular expressions are undecidable.

(1) context freedom (2) Finiteness (3) Emptiness
(4) Regularity.

Undecidable :- Given grammar is undecidable if its
(i) its corresponding language is non-recursive
(ii) it has no solution / answer / Algorithm.

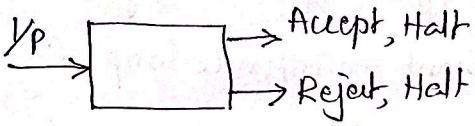
8. Halting Problem of T.M

For every T.M. the final results are

(1) Accept, Halt
(2) Reject, Halt
(3) Loop.

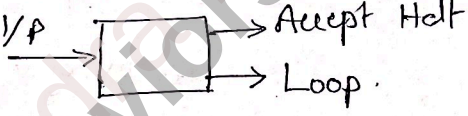
According to above results T.Ms are two types.

(1) type



- with this type TM, no problem.

(2) type.



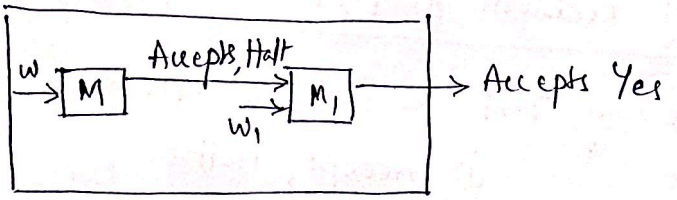
- with this type :-
  (i) if it accept/Halt → No problem.
  (ii) if it is Loop →
    (a) wait — decided not accept
    (b) wait — waiting for accept.

→ Halting problem is undecidable problem of T.M.

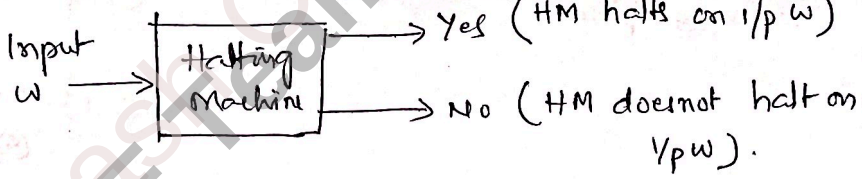
: Problem of Reduction:

- To prove undecidable problems, use concept of problem of reduction.
- Reduction is used to solve new problem (P_1) interms of another already solved problem (P_2) (using rules, theorems used for P_2).
- ∴ if P_1 is hard → P_2 also hard.
  if P_2 unsolvable — P_1 is unsolvable.
  if P_2 undecidable — P_1 undecidable.

- Prove that Halting problem is undecidable.



- Here we have two problem. 1) Machine M, string w
  2) Machine M1, string w1
- Let take an instance (M,w) and another instance (M1, w1) such that,
  M1 halts on the i/p w, if and only if M accepts w.
- if M reject w then went to infinite loop forms on w. and it doesnot proceed further M1 and halting don't takes place.



Hence, the acceptance behaviour of T.M is undecidable.
So, Halting problem is undecidable.

→ Post's Correspondence Problem (PCP) :-

- It is undecidable problem that was introduced by email post in 1946.
- Let two lists of strings over ~~some~~ some alphabet are Σ.
- The two lists must be equal length.
- $A = w_1 w_2 w_3 \dots w_k$

  $B = x_1 x_2 x_3 \dots x_k$ for some integer k.
- For each i the pair $(w_i x_i)$ is said to be corresponding pair.
- there exist non empty set of integers $i_1 i_2 i_3 \dots$ in such that $w_1 w_2 w_3 \dots w_n = x_1 x_2 x_3 \dots x_n$.

Example:1.

Given instance of PCP, tell whether this instance has a solution.

	List A	B
i	w_i	x_i
1	10	101
2	01	100
3	0	10
4	100	0
5	1	010

From this instance.

Consider List A:

$w_1 w_5 w_2 w_3 w_4 w_4 w_5 w_4$

$= 10101001000100$

consider List B.

$x_1 x_5 x_2 x_3 x_4 x_4 x_3 x_4$

$= 10101001000100$

∴ Here, Both are same, In this instance this PCP holds correspondence problem $i = i_1 i_5 i_2 i_3 i_4 i_4 i_3 i_4$

Example 2 In above problem. if consider list

$A = w_2 w_3 = 010$

$B = x_2 x_3 = 10010$ } Not equal

for this PCP has no solution. $i = i_2 i_3$.

Example

i	A: w_i	B: x_i
1	110	110110
2	00011	00
3	0110	110

Let instance $A = w_2 w_3 w_1 = 00011\ 0110\ 110$

$B = x_2 x_3 x_1 = 00\ 110\ 110\ 110$

equal Hence PCP have a solution for $i = 2, 3, 1$.

Example: List $A = (b, bab^3, ba)$

$B = (b^3, ba, a)$ Find solution ?

A solution is consider A list $= w_2 w_1 w_1 w_3$

$= bab^3\ b\ b\ ba = bab^3 b^3 a$

B list $= x_2 x_1 x_1 x_3$

$= ba\ b^3\ b^3\ a$

Hence, both are same PCP have a solution $i = i_2 i_1 i_1 i_3$.

Example:

i	A: w_i	B: x_i
1	011	101
2	11	011
3	1101	110

Check whether there is a solution for PCP or not ?

Ans: No solution for any combination of instances i.

8. Modified PCP :- In modified PCP, there is additional requirement on a solution that the first pair on the A and B lists must be the first pair in the solution.

Instances of MPCP is two lists $A = w_1 w_2 \ldots w_k$

$B = x_1 x_2 \ldots x_k$

and solution is a list of 0 (zero) or more integers $i_1, i_2 \ldots i_m$ such that

$w_1 w_{i_1} w_{i_2} w_{i_3} \ldots w_{i_m} = x_1 x_{i_1} x_{i_2} x_{i_3} \ldots x_{i_m}$

Notice that the pair (w_1, x_1) is forced to be in the begining of the two strings.

Example on MPCP

i	A: w_i	B: x_i
1	110	110110
2	0011	00
3	0110	110

For this problem, as an instance of MPCP it has no solution.

because, partial solution begin with index 1.

like :- $A = \boxed{w_1}\ w_2\ w_3$

$B = \boxed{x_1}\ x_2\ x_3$ ← you can take any sequence

↑ must be same (i.e. index with 1 only)

Ex. $A = w_1\ w_3\ w_2 = 110\ 0110\ 0011$

$B = x_1\ x_3\ x_2 = 110110\ 110\ 00$ } not same

(or) $A = w_1\ w_2\ w_3 = 110\ 0011\ 0110$

$B = x_1\ x_2\ x_3 = 110110\ 00\ 110$ } Not equal.

Hence, there is no solution for MPCP.

UNIT-V

Intractable Problems :-

Problems

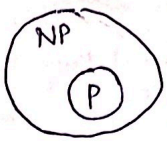
- solvable
- unsolvable

Problems

- Group 1
  - Solutions time within polinomial time Alg.
  - Ex: Linear search $O(n)$
    Binary search $O(\log n)$
    Bubble sort $O(n^2)$
    merge sort $(O \log n)$
  - Tractable problems
- Group 2
  - solution time in non-polynomial Algorithm. (Exponential time).
  - Ex: 0/1 Knapsack $O(2^n)$
    TSP $O(n^2 2^n)$
    sum of subsets $O(2^n)$
    Graph coloring $O(2^n)$
  - Intractable problems.

Problems :

- P (Tractable)
  - set of all Decision problems solvable by deterministic Alg in polynomial time.
  - Ex: Searching
    Sorting
    Find prime
    Find GCD
    Kruskal Alg
  - Efficient Algorithms.
- $P \subseteq NP$
- NP (Intractable)
  - set of all decision problems solvable by Non-deterministic Alg in polynomial time.
  - Ex: TSP
    Knapsack problem.
    Job scheduling
    n-Queen problem.

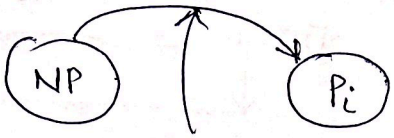


$P \subseteq NP$ $P = NP$?

Case 1: $P = NP$
- Problems can solved in less time
- security Alg. fails

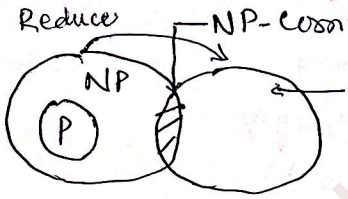
Case 2: $P \neq NP$ – Not correct
- we can solve NP problems in future.
- NP problems can also solve by reduction to P.

- Apply Reduction technique try to solve NP problems.



Conversion time is polynomial time

∴ This P_i is said to be NP Hard if all NP problems are reduces to P_i in polynomial time.



NP Problems

NP-Hard

- A decision problem P_i is NP Hard if every problem in NP is polynomial reducible to P_i
- Problems can be solved in polynomial time then all NP-Complete problems can be solved in polynomial time.

NP-complete

- A Decision problem P_i is NP-complete if it is NP-Hard and is also in the class NP itself.
- A problem NP-complete can solved in polynomial time iff all other NP-complete problems can also solved in polynomial time

Ex: Set-cover problem
Vertex-cover problem
Clique problem
3-colour problem.
TSP, Hamiltonian cycle

Algorithms

Deterministic Alg.

- The result of every operation is uniquely defined.

Ex: All Algorithms.

Non-Deterministic Alg.

- The result of every operation is not uniquely defined but limited to specific set of possibilities.
- Three functions
  - choice ( )
  - failure ( )
  - success ( )

Ex: searching, knapsack } we can write in NDA.

Problems

Decidable problem

- Result is in Yes/No 0/1.

Ex: Given number is prime ?

- Alg to take Decision.

Optimization Problems.

- Result in max/min. (optimal)

Ex: TSP
shortest path

Problems:

Reducibility

- Let L_1 and L_2 be problems, L_1 reduces to L_2, iff there solution to solve L by polynomial time using L_2 (Already solve by polynomial time).

$NP \xrightarrow{\text{Reduce}} P$

satisfiability

- satisfiability is in P iff NP=P
  ∴ P=NP.
- A problem L is NP-Hard iff satisfiability reduces to L
- A problem L is NP-complete, iff L is NP-Hard

§ Context-Sensitive Languages (CSL):-

A Language (L) is said to be context sensitive if there exists a context sensitive grammer G. such that $L = L(G)$ or $L = L(G) \cup \{\in\}$.

- A context sensitive grammer is defined as A grammer $G=(V,T,P,S)$ is said to be context sensitive if all productions are in the form $x \to y$ where $x, y \in (V \cup T)^*$ and $|x| \leq |y|$
- A context sensitive grammer never generate a language containing the empty string $x \to \in$ is not allowed.
- Every CFG without $\in$ can be generated by a special case CSL, say by one in CNF or GNF, both which satisfy the conditions of CSL.

Ex:- The language $L = \{a^n b^n c^n / n \geq 1\}$ is a CFL grammer is

$S \to abc / aABL$
$Ab \to bA$
$AC \to Bbcc$
$bB \to Bb$
$aB \to aa / aaA$

Solution: Derivation tree of $a^3 b^3 c^3$ is looking to be as following

$S \Rightarrow aAbc$
$\Rightarrow abAc$
$\Rightarrow abBbcc$
$\Rightarrow aBbbcc$
$\Rightarrow aaAbbcc$
$\Rightarrow aabAbcc$
$\Rightarrow aabbAcc$
$\Rightarrow aabbBbccc$
$\Rightarrow aabBbbccc$
$\Rightarrow aaBbbbccc$
$\Rightarrow aaabbbccc$

§ Linear Bounded Automata (LBA)

A linear bounded automaton (LBA) is a non-deterministic TM.

It will be denoted as $M = (Q, \Sigma, \tau, \delta, q_0, ¢, \$, F)$

Where $Q, \Sigma, \tau, \delta, q_0$ and F are as for a non-deterministic TM

¢, $ are symbols in Σ

¢ – Left end marker

$ – Right end marker

δ is a function $\delta : Q \times \tau \to 2^{Q \times \tau \times \{L,R\}}$ subjected to restriction Σ must be contain two special symbols ¢, $ such that $\delta(q_i, ¢)$ can contain only elements of the form $(q_j, ¢, R)$ $\delta(q_i, \$)$ contain only elements of the form $(q_j, \$, L)$ LBA satisfies two conditions.

1. Its input alphabet (Σ) includes two special symbols ¢ and $, the left and right end markers respectively.
2. The LBA has no moves left from ¢ or right from $.

§ Instantaneous Description (ID)

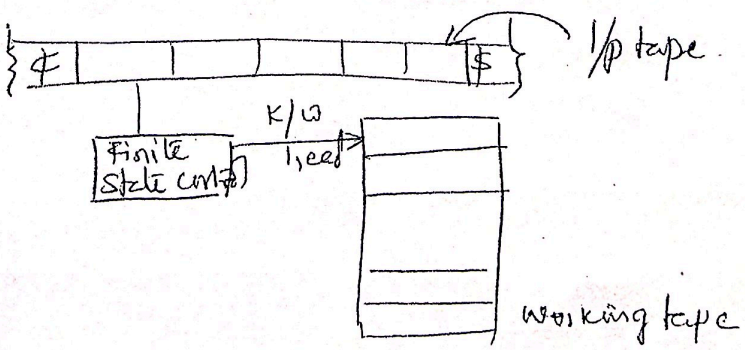
ID of LBA is denoted $\alpha q \beta$ where $\alpha, \beta \in \tau^*$

The initial configuration is given by $q_0 ¢ w \$$. A string w is accepted by LBA if $q_0 ¢ w \$ \vdash^* \alpha q_f \beta$ for some $q_f \in F$, $\alpha, \beta \in \tau^*$.

§ The language accepted by LBA :-

$L(M)$ is $= \{ w / w \text{ is in } (\Sigma - \{¢, \$\})^* \text{ and } q_0 ¢ w \$ \vdash_M^* \alpha q_f \beta \text{ for some } q_f \text{ in } F \}$.

§ Model of LBA



Linear Bounded Automata (LBA)

Applications:

1. To model digital computer
2. Halting problem for LBA is decidable
3. LBA is powerful that PDA.

   Ex: $L = \{a^n b^n c^n / n \geq 1\}$ is not accepted by PDA

   But, we can construct with LBA

- It recognise Context sensitive Languages (CSL) Type 1.

- Properties of CSL

  1) CSL closed under

     - Union, Kleene closure, concatenation, Intersection, Reversal, Substitution Homomorphism, Inverse Homomorphism.