Personal Data

Name : _____

School /College : _____

_____

Section/Branch : _____

Roll No:/Reg.No : _____

Subject : _____

Address : _____

_____

Mobile : _____

E-Mail : _____

facebook : _____

Contact Person Name & No. on Emergency:

_____

We all need people who will give us feedback. That's how we improve.

-Bill Gates

ideas inside

REDUCE

Lowering the amount of waste Produced

REUSE

Using materials reapeatedly

RECYCLE

Using materials to make New Products

RECOVERY

Recovering Energy from Waste

LANDFILL

Safe disposal of waste to Landfill

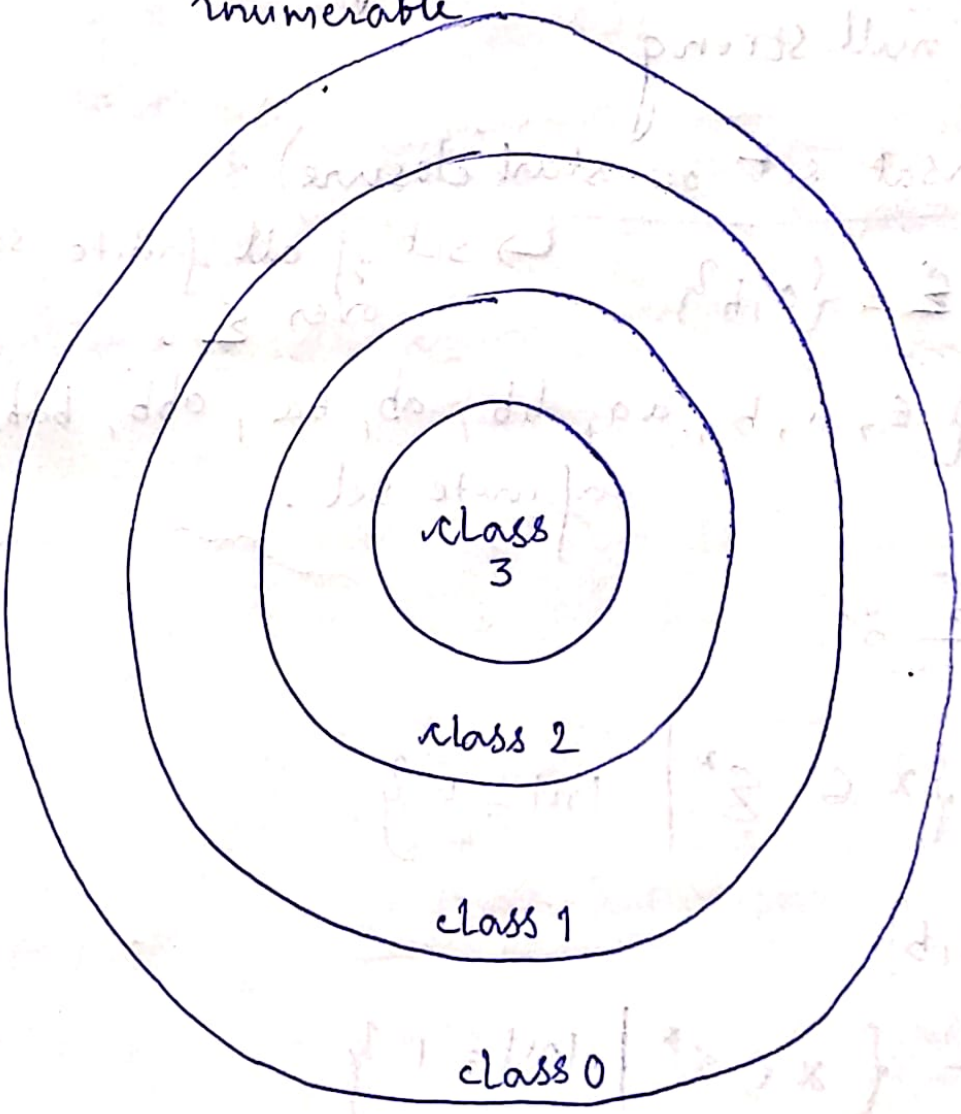
INDEX

NAME: Shreyansh STD: SEC: ROLL NO: SUB: TOC

S.No.	Date	Title	Page No.	Teacher's Sign/ Remarks
		first term - 30	5	
		End Sem - 50	5	
		Surprise quiz - 20 (or online test)	5	
			5	
		Reference Book -		
		① Peter Linz - Introduction to formal & Automa Language		
②		M. Sipser - An Introduction to ~~formal~~ theory of computation		

Chomsky Hierarchy

Class	Language	Grammers	Automata
Type 3	Regular	Regular	Finite
Type 2	Context free	context free	Push down
Type 1	Context sensitive	context sensitive	Linear bounded
Type 0	Recursive Enumerable	Unrestricted	Turing Machine



Symbols - fundamental blocks of a language.

Alphabets - finite set of symbols.

String - A string over Alphabets is a finite set of symbols.

Consider a string, $x = aab$

$|x| = |aab| =$ length of string

$n_a(x)$ = the no of 'a' in string x.

λ or ϵ ⇒ null string

Σ^* (universal set or start closure) :-

↳ set of all finite strings over Σ.

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

$\Sigma^* = \{\epsilon, a, b, aa, bb, ab, ba, abb, bab \ldots\ldots\}$

infinite set.

Power of Σ :-

$$\Sigma^k = \{x \in \Sigma^* \mid |x| = k\}$$

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

$\Sigma^1 = \{x \in \Sigma^* \mid |x| = 1\}$

$= \{a, b\} = 2^1$ elements

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

Positive closure :- $\boxed{\Sigma^+ = \Sigma^* - \{\in\}}$

→ Language over Σ is a subset of Σ^*

L1 = Set of all strings over $\{a, b\}$ of length 2.

↳ This is informal description of language.

$L1 = \{aa, ab, ba, bb\}$

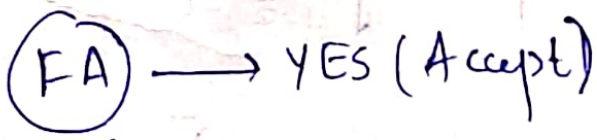
$L = \{x \in \{a, b\} \mid |x| = 2\}$

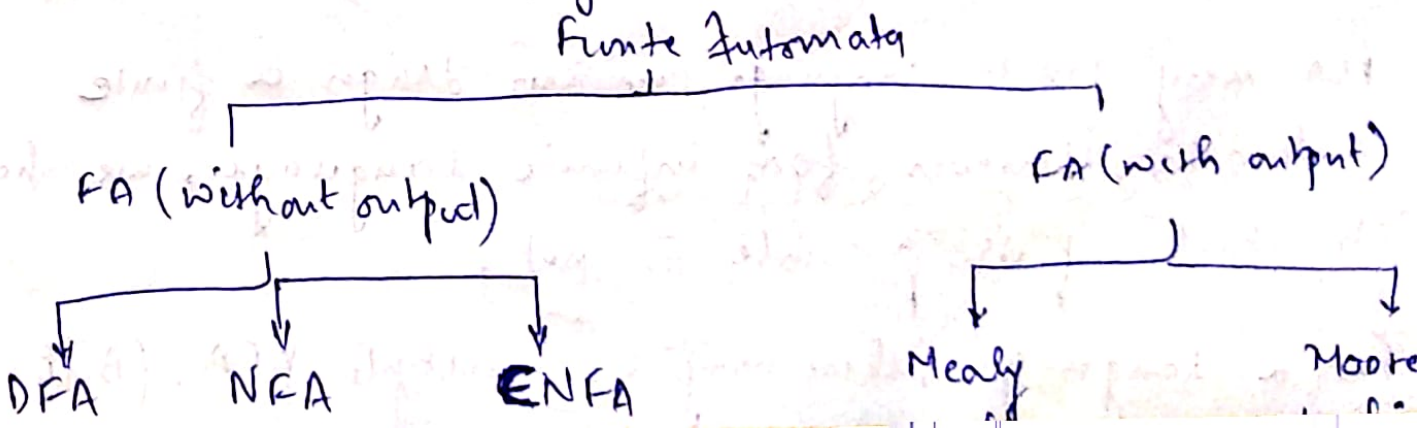
↳ formal description

⇒ Language of real Java identifiers.

⇒ Language of numerical literals

Finite Automata :-





DFA → deterministic FA

NFA → Non deterministic FA

∈-NFA = epsilon NFA

Finite Automata without output (FA):-

DFA → can be defined by 5-tuples.

$\{Q, Q_0, Q_F, \Sigma, \delta\}$ // formal definition.

Q = set of finite states

Q_0 = initial state ($Q_0 \in Q$)

Q_F = set of accepted state. ($Q_F \in Q$)

Σ = Alphabet

δ = transition function

DFA:-

For a particular input symbol, machine goes to one state only.

For transition function $\delta(Q, \Sigma) \rightarrow Q$

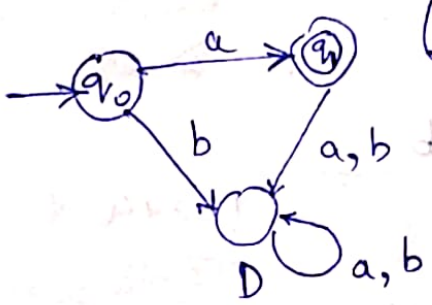
NULL(∈) move is not allowed.

For every finite language, we can design a finite automata. However, for infinite language, we have to check if its possible or not.

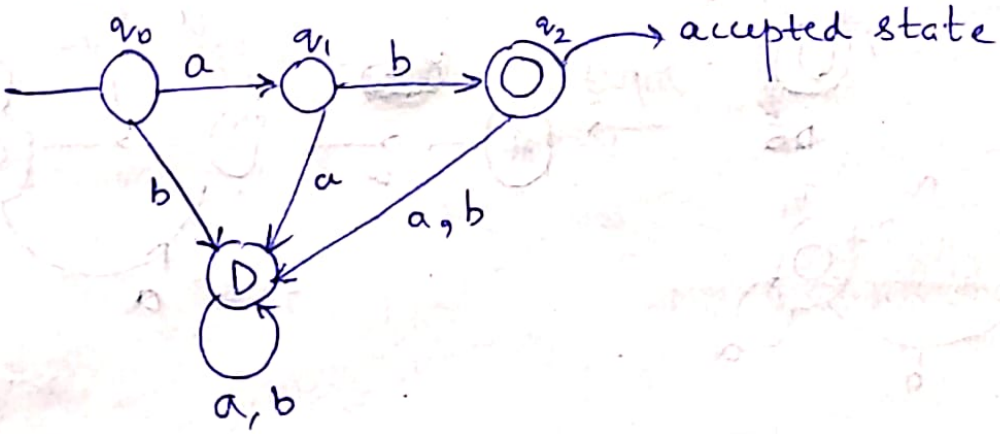
For a language, there may be multiple DFA. (But minimal DFA will be unique)

From every stage, machine must have outgoing transition for each alphabet.

a) Design DFA for a language containing string ab only over {a, b}.



(only for A)
D = Dead state



Design a DFA for a language containing strings starting from a?

$L = \{a, ab, aab \cdots\}$

Basic String = {a}

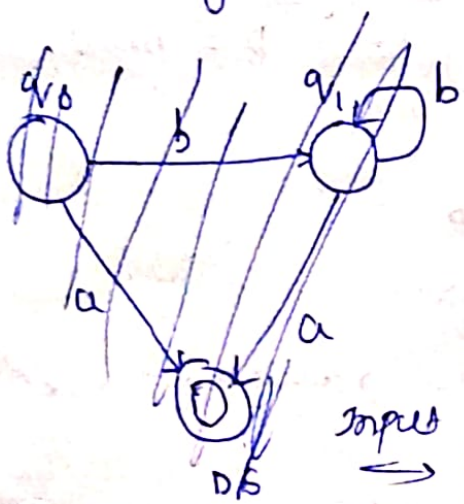
q_1 ⇒ Accepted state

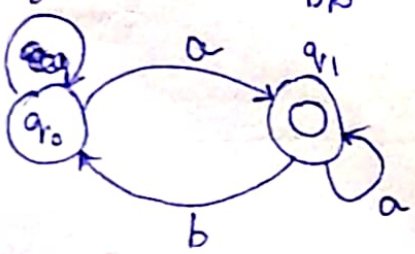
D = Dead state.

when the language contains ε, ... start will became final state.

a) Design a DFA for a language containing strings ending with a. {a,b}

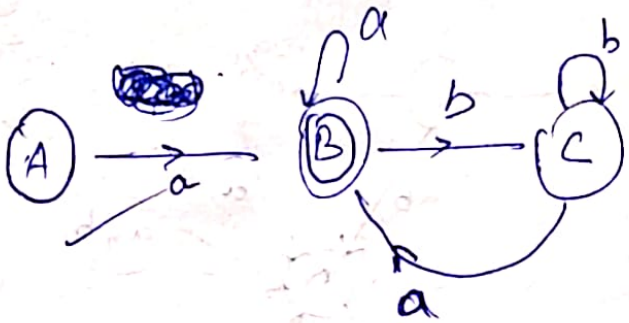
Basic string = {a}





wa
a
a

abaaa



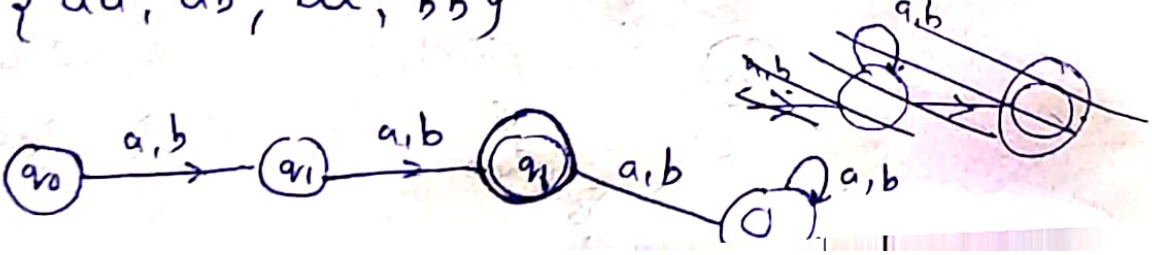
a) Design DFA

① $L = \{ w \in (a,b)^* \mid |w| = 2 \}$

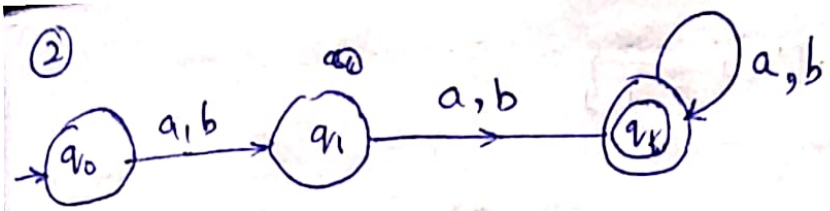
② $L = \{ w \in (a,b)^* \mid |w| \geq 2 \}$

③ $L = \{ w \in (a,b)^* \mid |w| \leq 2 \}$

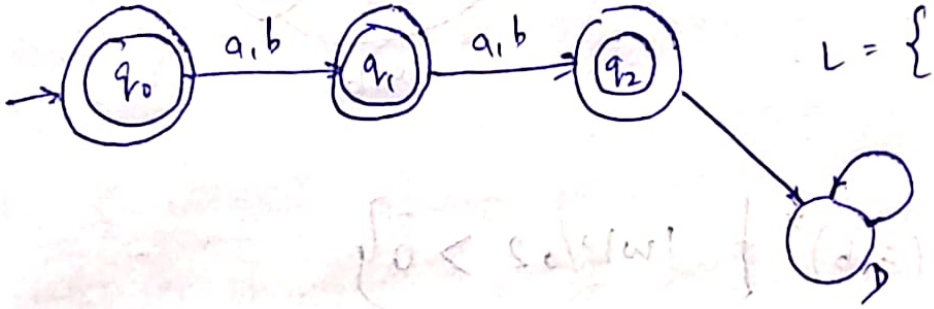
① $L = \{aa, ab, ba, bb\}$



(2)



(3)



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

for (i), $|w| = 2$

~~no of steps = 2+2 = 4~~

no of states = 2+2 = 4 $(n+2)$

↳ in general

for (ii) $|w| = n$

~~no of steps~~

no of states = $n+1$

for (iii) $|w| = n$

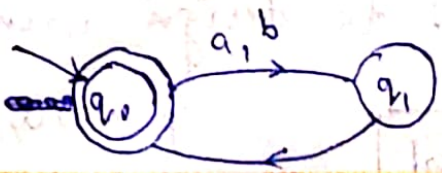
no of states = $n+2$

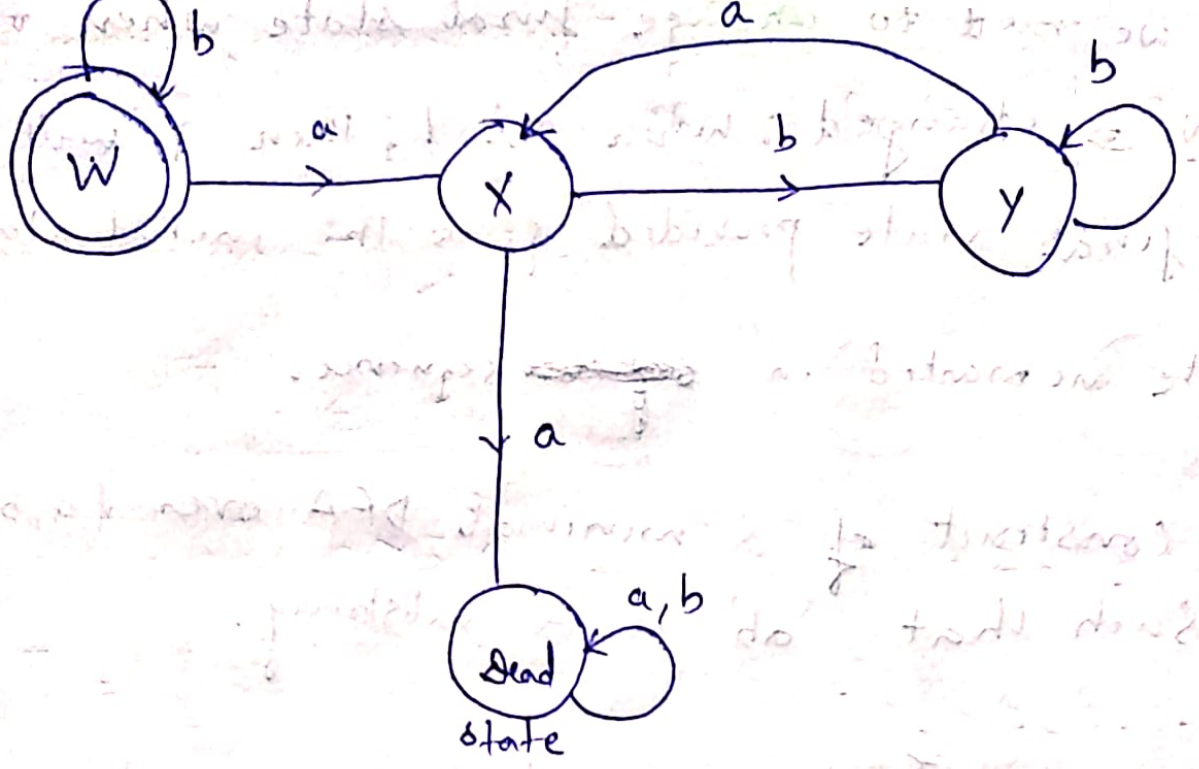
DFA based on modulus :-

(1) $L = \{ w \in \{a,b\}^* \mid |w| \bmod 2 = 0 \}$

(2) $L = \{ w \in (a,b)^* \mid |w| \bmod 2 = 1 \}$

(1) $L = \{\epsilon, aa, ab, ba, bb\}$





(a) Construct a DFA in which 'a' is never followed by 'b' for set $\{a,b\}$

↳ Prefix / substring / suffix :-

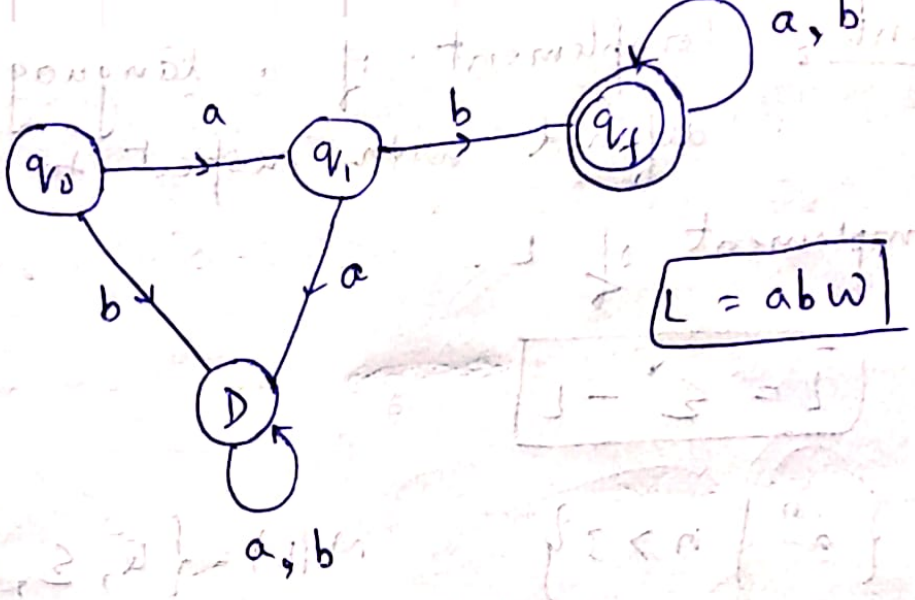
$w = abab\underline{aa}$

Design a DFA such that there is a prefix 'ab'.

$$L = \{ w \in (a,b)^* \mid w \text{ has prefix 'ab'} \}$$

Basic string = ab

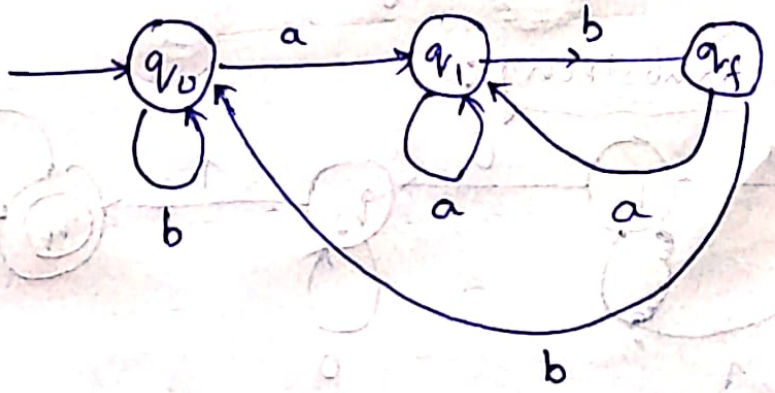
$L = \{ ab, abaaa, abbbb, abbaba, \ldots \}$



$L = abw$

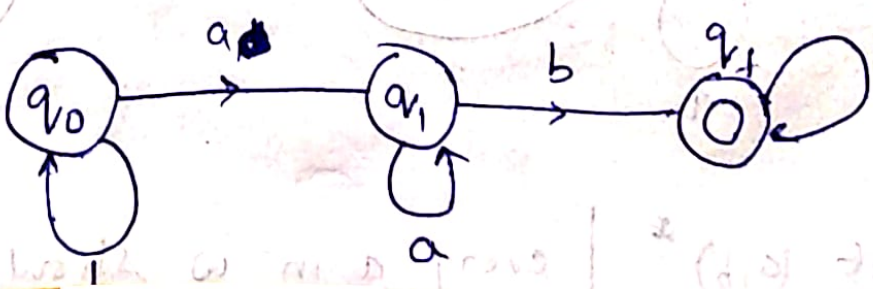
Q) Design a DFA such that there is a suffix 'ab'

$L = \{ w \in (a,b)^* \mid w \text{ has suffix 'ab'} \}$



a) Design a DFA such that w has substring 'ab'

$L = \{ w \in (a,b)^* \mid w \text{ has substring 'ab'} \}$



Complement :- Complement of a language is defined with respect to Σ^*, that is the complement of L.

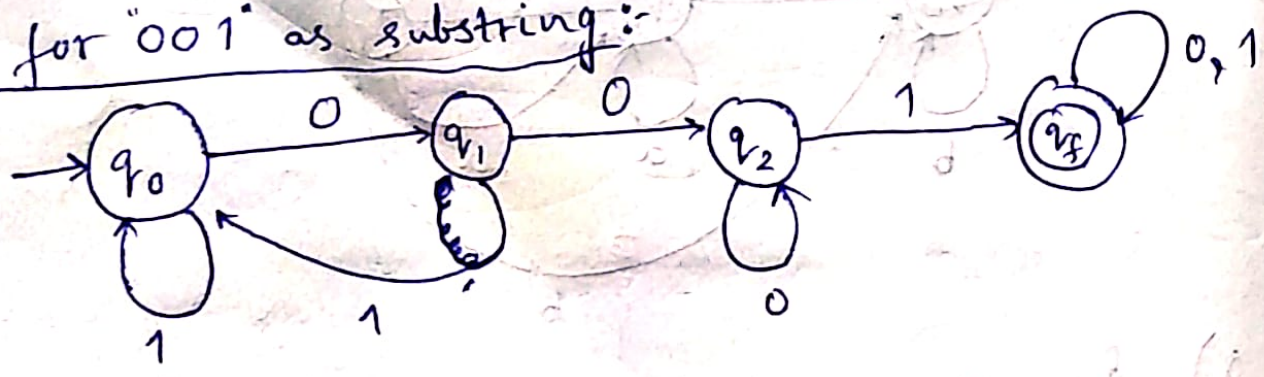
$$\bar{L} = \Sigma^* - L$$

Eg :- $L = \{ a^n \mid n > 3 \}$ $\quad M(L) = \{Q, \Sigma, \delta, q_0, q_f\}$

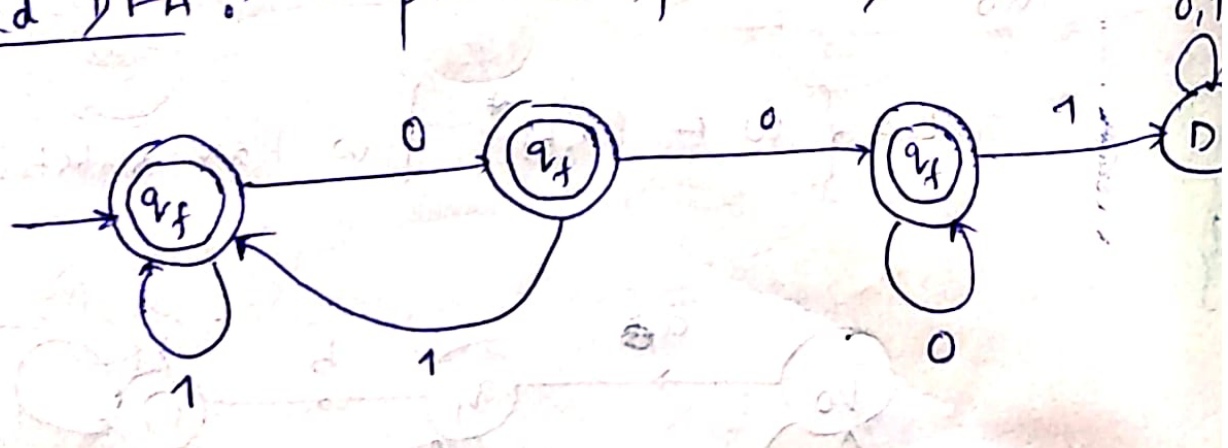
$\bar{L} = \{ a^n \mid n <= 3 \}$ $\quad M(\bar{L}) = \{Q, \Sigma, \delta, q_0, Q-q$

Q) Design a DFA that accepts all strings over $\{0, 1\}$ except those containing "001" as substring

DFA for "001" as substring :-

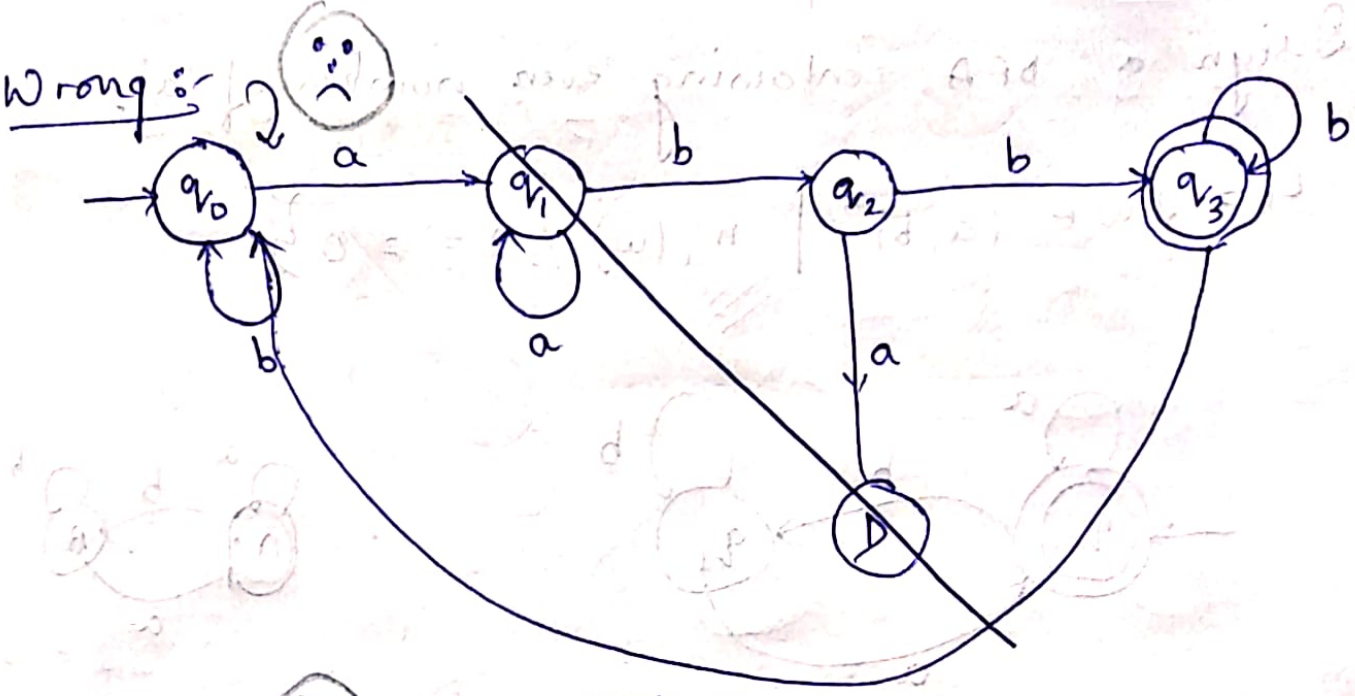


Required DFA :- Complement of above DFA.

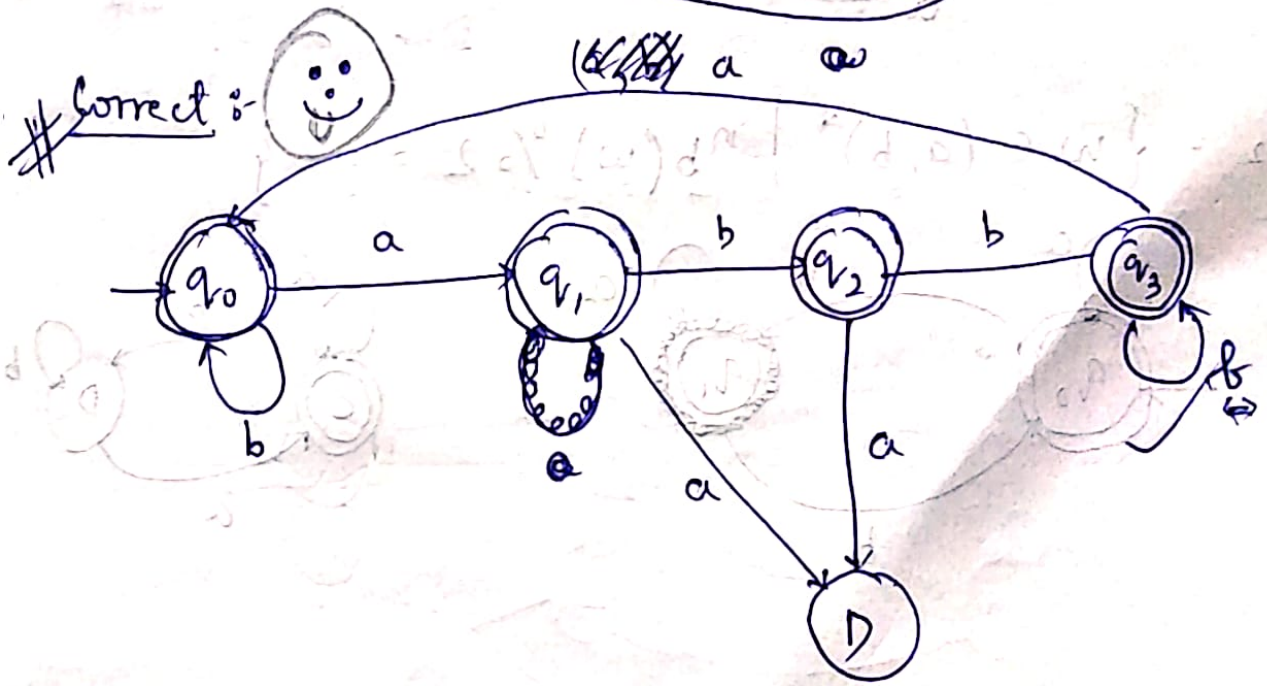


$L = \{ w \in (a,b)^* \mid$ every a in w should be followe

Wrong :-



Correct :-



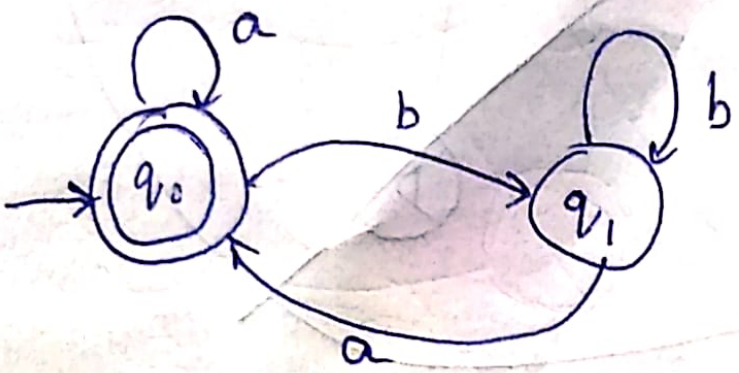
Note :- If L_1 and L_2 are compliment to each other then no string must be common in them.

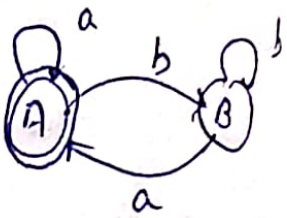
For compliment :- $L_1 \cap L_2 = \{\epsilon\}$.

$L_1 = \{ w \in (ab) \mid$ every a in w should never be followed by b $\}$

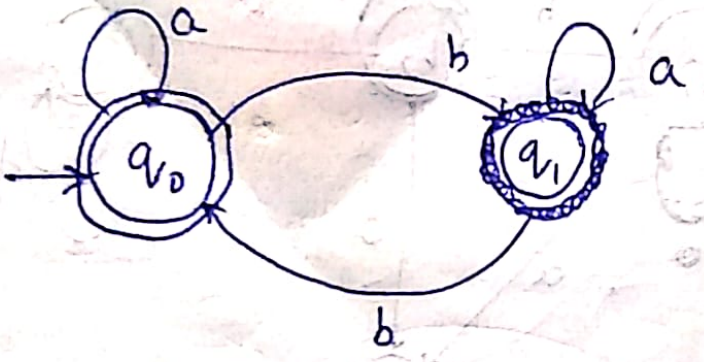
Design a DFA containing even number of 'a'.

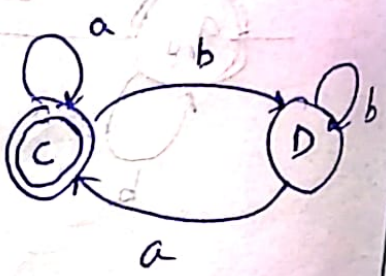
$L_1 = \{ w \in (a,b)^* \mid n_a(w) \% 2 == 0 \}$



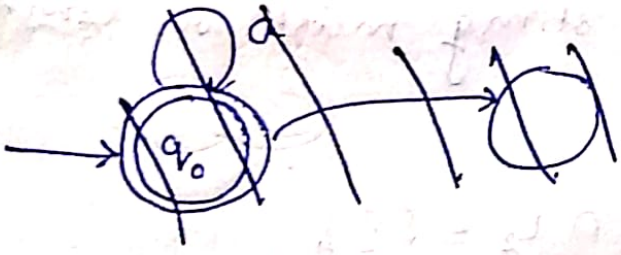


$L_2 = \{ w \in (a,b)^* \mid n_b(w) \% 2 == 0 \}$





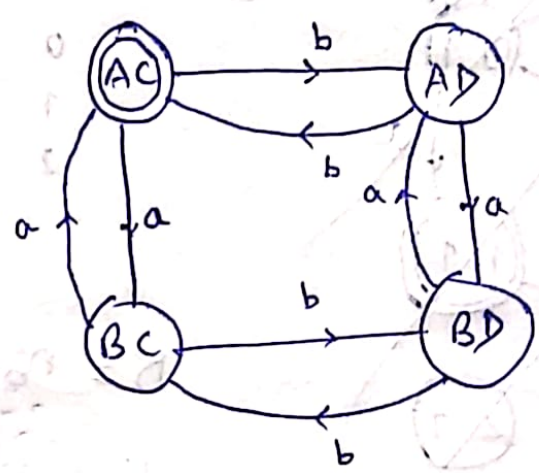
$L_3 = \{ w \in (a,b)^* \mid n_a(w) \% 2 == 0 \text{ \&\& } n_b(w) \% 2 == 0 \}$



e e
e o
o e
o o

Cross Product :-

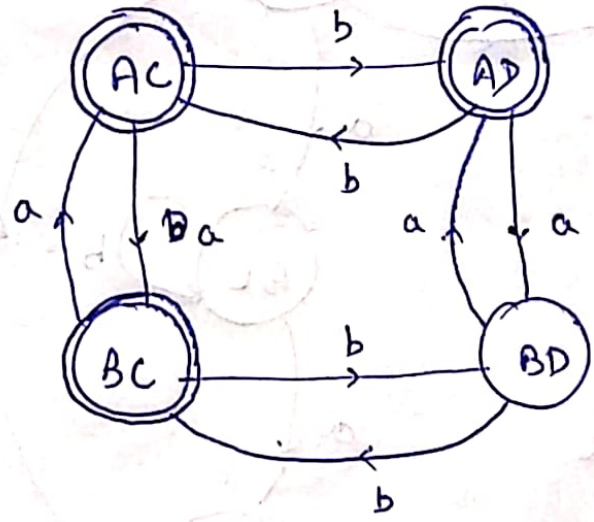
$\{(A,B)\} \times \{(C,D)\} = \{AC, AD, BC, BD\}$



Minimal DFA

	a	b
AC	A → B	A → A
	C → C	C → D
AD	A → B	A → A
	D → D	D → B
BC	B → A	B → B
	C → C	C → D
BD	B → A	B → B
	D → D	D → C

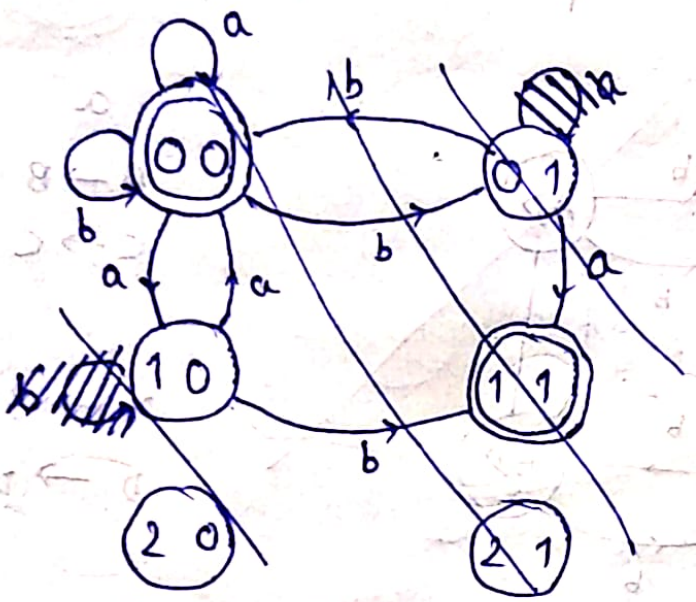
$L_4 = \{w \in (a,b)^* \mid n_a(w)\%2 == 0 \;||\; n_b(w)\%2 == 0\}$



Q) Design minimal DFA

i) $L = \{w \in (a,b)^* \mid n_a(w) \bmod 3 \geq n_b(w) \bmod 2\}$

ii) $L_2 = \{ w \in (a,b)^* \mid n_a(w) \% 3 == n_b(w) \% 2 \}$



0 0
1 1
2

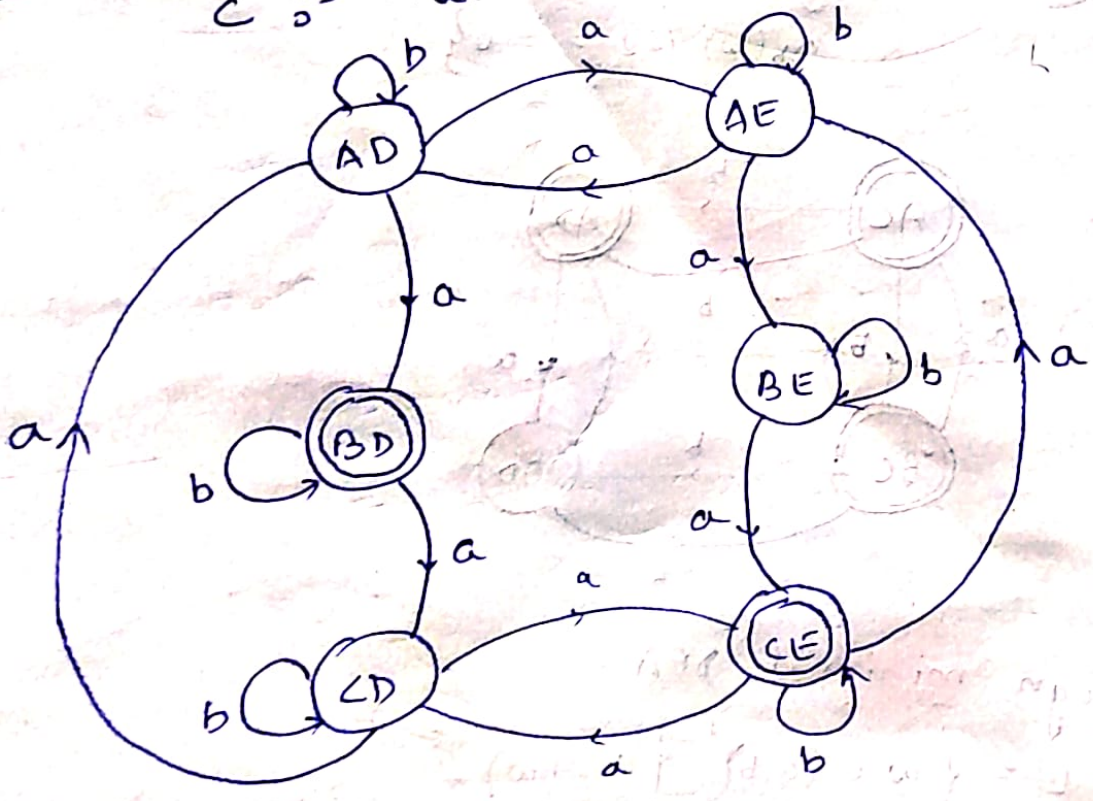
D = $n_a(w) \% 2 =$

E = $n_a(w) \% 2$

A :- $n_a(w) \% 3 == 0$

B :- $n_a(w) \% 3 == 1$

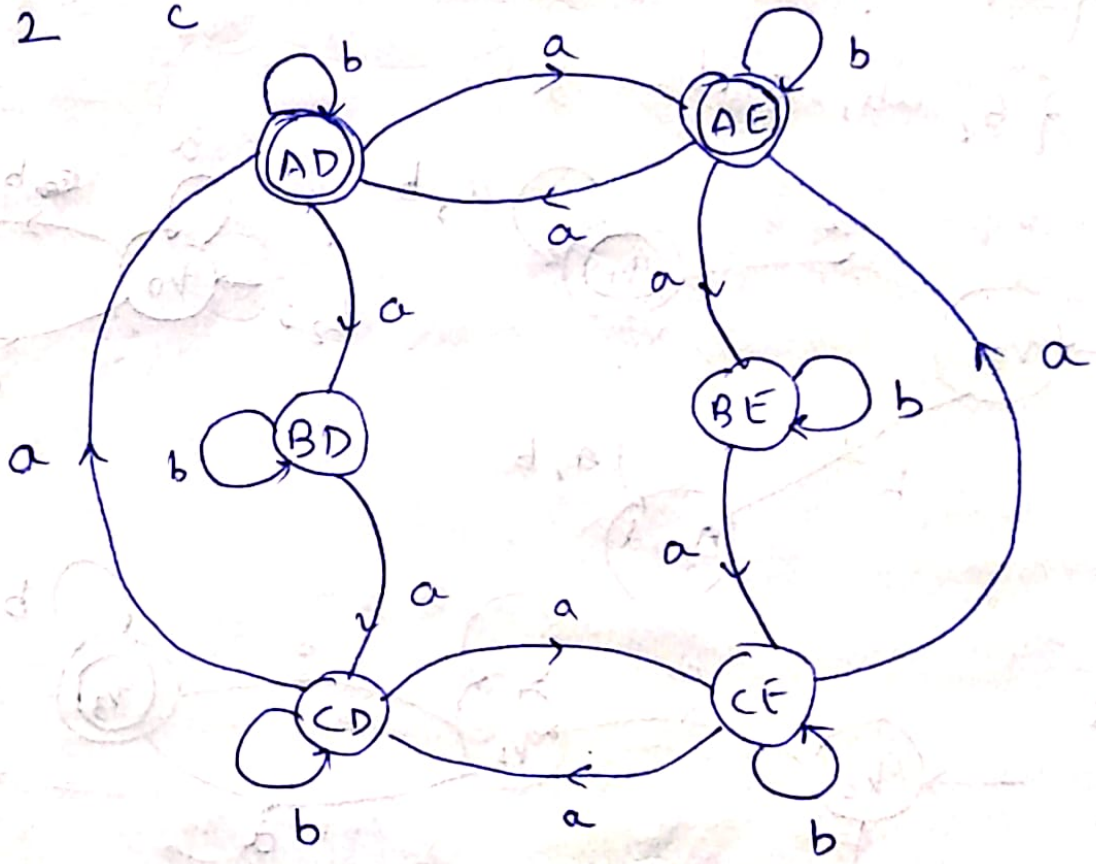
C :- $n_a(w) \% 3 == 2$



i) $n_a(w) \% 3 \geq n_b(w) \% 2$.

0	A	D, E
1	B	
2	C	

AD, AE ⇒ accepted states



Concatenation of 2 languages L_1 & L_2 is the set of all strings obtained by concatenating elements of L_1 with any element of L_2.
Specifically $L_1 L_2 = \{xy ; x \in L_1, y \in L_2\}$

// We can club final state of L_1 & initial state of L_2 provided L_1 has only one final state. But we have to additionally take care of clubbing point.

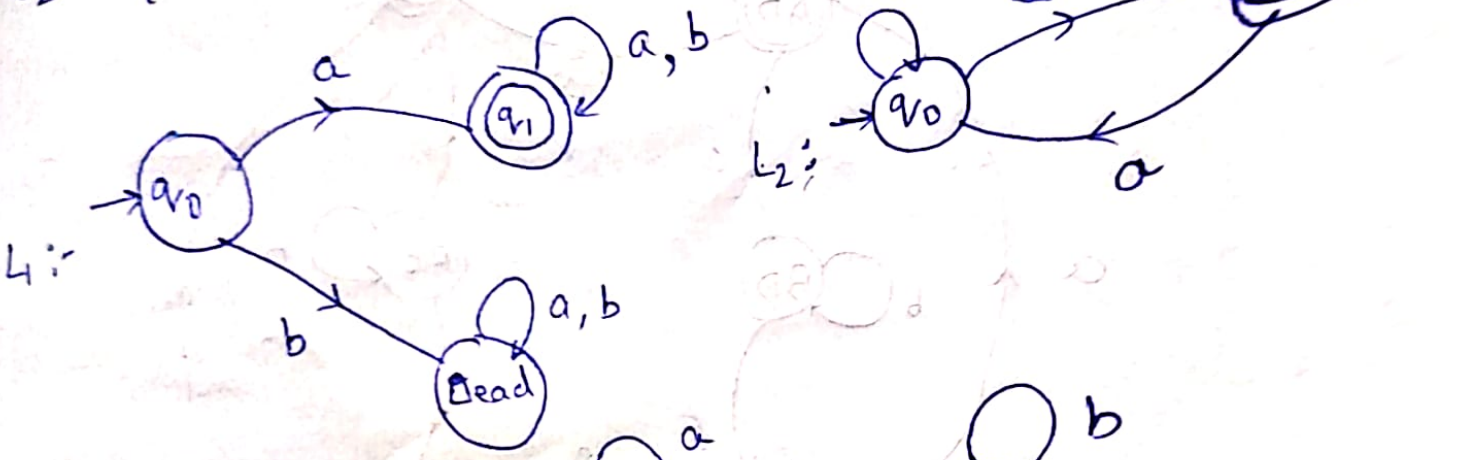
// Concatenation is very useful concepts in NFA.

Q) $L = \{ w \in (a,b)^* \mid w \text{ starts with } a \}$

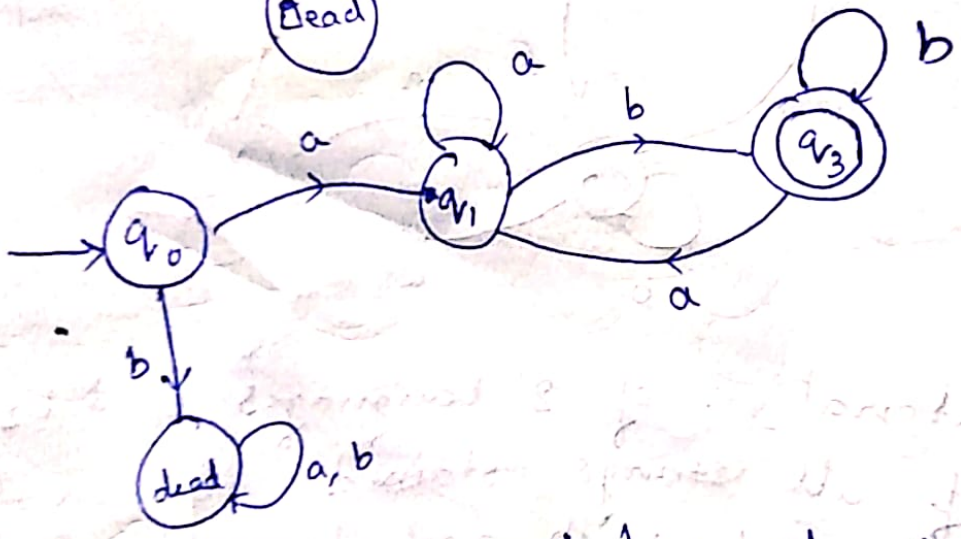
$L_2 = \{ w \in (a,b)^* \mid w \text{ end with } b \}$

$L_1 = \{ a, ab, aa \ldots \}$

$L_2 = \{ b, ab, aab \ldots \}$



Ans :-

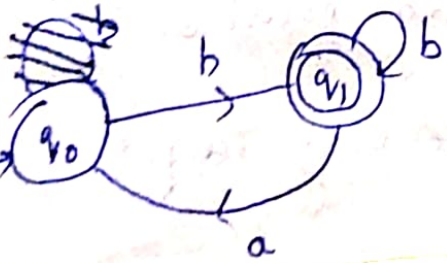


* Reversal :- $L = \{ w \in (a,b)^* \mid w \text{ end with } b \}$

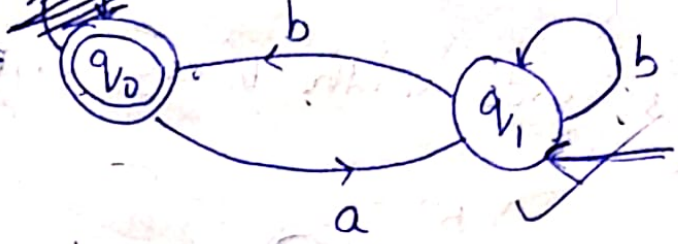
$= \{ b, ab, aab \ldots \}$

$L^R = \{ b, ba, baa \ldots \}$

DLA (L^R) $\Rightarrow$



DFA(L^*) =



Non Deterministic Finite Automata (NFA / NDFA) :-

A NFA is defined by $M = (Q, \Sigma, \delta, q_0, Q_F)$

δ for NFA

$\delta : Q \times \Sigma \rightarrow 2^{|Q|}$

power set of Q

δ for $\in$ NFA

$\delta : Q \times (\Sigma \cup \{\in\}) \rightarrow 2^{|Q|}$

For a set $|Q|$ states there are $2^{|Q|}$ subsets.
Hence Q, q_0, Q_F, Σ are defined as DFA.

Note :-

Multiple transactions over an alphabet are possible.

There is no concept of dead state.

The class of DFA & NFA are equally powerful. every NFA has equivalent NFA / DFA.

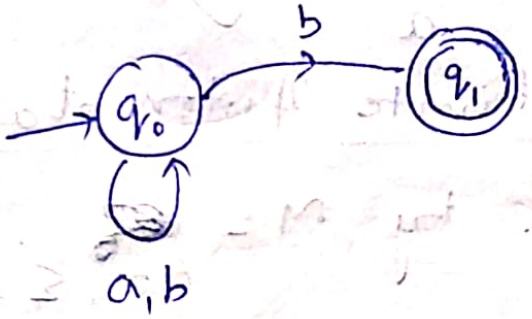
NFA for respective language must accept all strings in the language & reject all which are not part of language.

Avoid unecessary transition.

In $\in$ NFA even without input we can change

↳ L = {strings ends with b over {a,b}}

NFA :-



$q_0 \xrightarrow{a,b} \{A\}$

$B \xrightarrow{a,b} \{\phi\}$

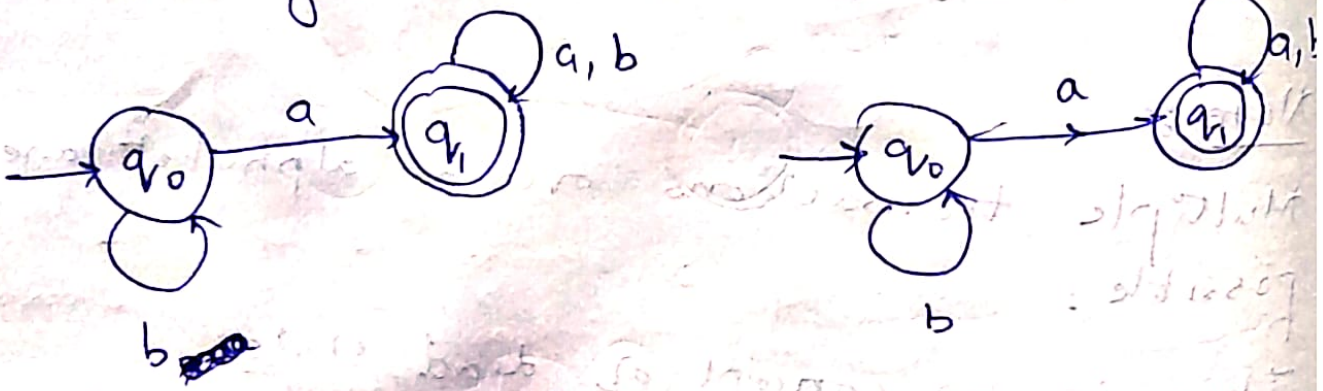
↳ a null state

$Q = \{q_0, q_1\}$

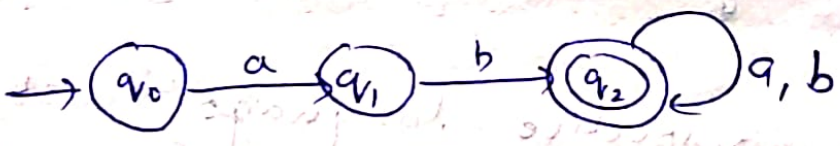
$2^n = \{\phi, q_0, q_1, \{q_0, q_1\}\}$

Design NFA.

L_1 = {string contains 'a'}



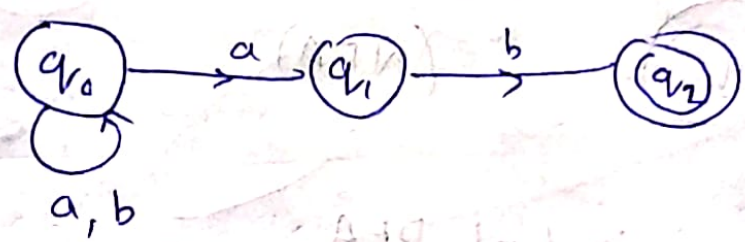
L_2 = string starts with 'ab'



// DFA can be considered as NFA, just remove dead state.

// If any possible transformation is not specified it will go to dead state.

L_3 = { string ends with 'ab' }



Procedure to convert NFA to DFA :-

Identify initial state of NFA as initial state of equivalent DFA.

Repeat following steps until no more edges are missing.

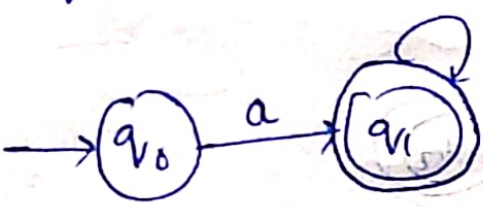
Take an vertex $\{q_i, q_j, q_k\}$ in equivalent DFA that has no outgoing edge for some $a \in \Sigma$ and compute $\delta^*(q_i, a) \cup \delta^*(q_j, a) \cdots \cup \delta^*(q_k, a) = \{q_l\, q_m \ldots q_n\}$

Make $\{q_l\, q_m \ldots q_n\}$ as new state if it doesn't exist in equivalent DFA, then complete the outgoing transition over Σ.

Every state in equivalent DFA whose state contain any $q_f \in F_{NFA}$ (final states of NFA) is identified as final state.

If NFA accepts $\in$, then make initial state of

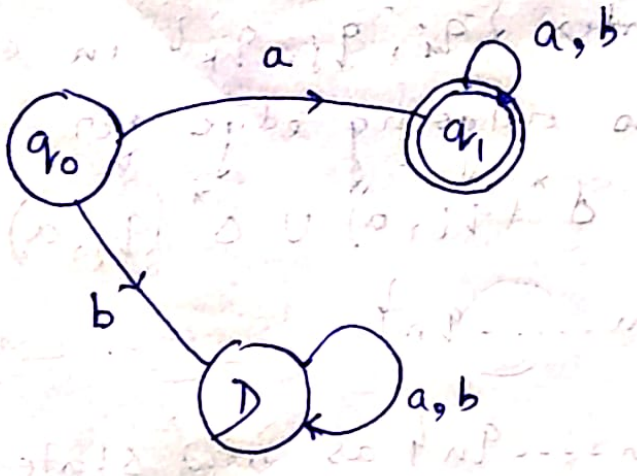
Q) Design DFA for L = {string starting with 'a'}



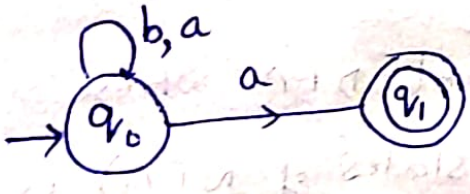
(NFA)

NFA to equivalent DFA

	a	b
→ q_0	$\{q_1\}$	dead
→ q_1	$\{q_1\}$	$\{q_1\}$
Dead	Dead	Dead



Q) Design DFA for L = {string ending with 'a'}

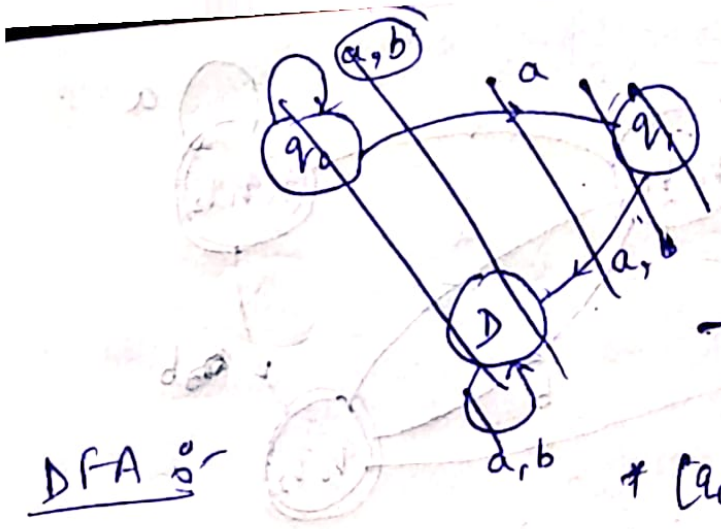


(NFA)

NFA

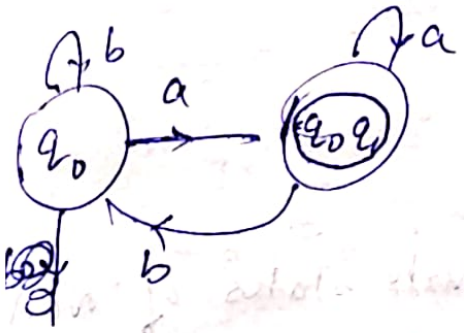
	a	b
→ q_0	$\{q_1, q_0\}$	$\{q_0\}$
→ q_1	$\{\}$ Dead	Dead $\{\}$
→ Dead	Dead	

aba



DFA :-

	a	b
$[q_0]$	$[q_0 q_1]$	$[q_0]$
* $[q_0 q_1]$	$[q_0 q_1]$	$[q_0]$



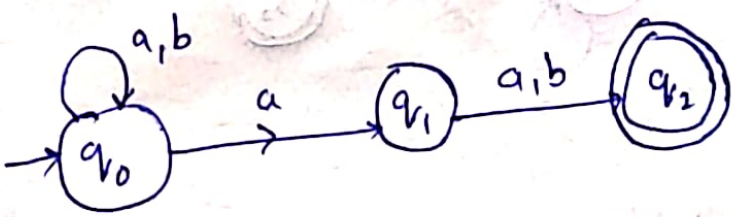
$[q_0 q_1] = \delta^*(q_0, a) \cup \delta^*(q_1, a)$

$\delta^*(q_0, a) = [q_0, q_1] \cup \{\} = [q_0 q_1]$

$\delta^*(q_1, a) = [q_0] \cup \{\} = [q_0]$

Design NFA for L = {string contains second symbol from RHS 'a'}

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



	a	b
q_0	$\{q_0 q_1\}$	$\{q_0\}$
$\{q_0 q_1\}$	$\{q_0 q_1 q_2\}$	$\{q_0 q_2\}$
$q_0 q_1 q_2$	$q_0 q_1 q_2$	$q_0 q_2$

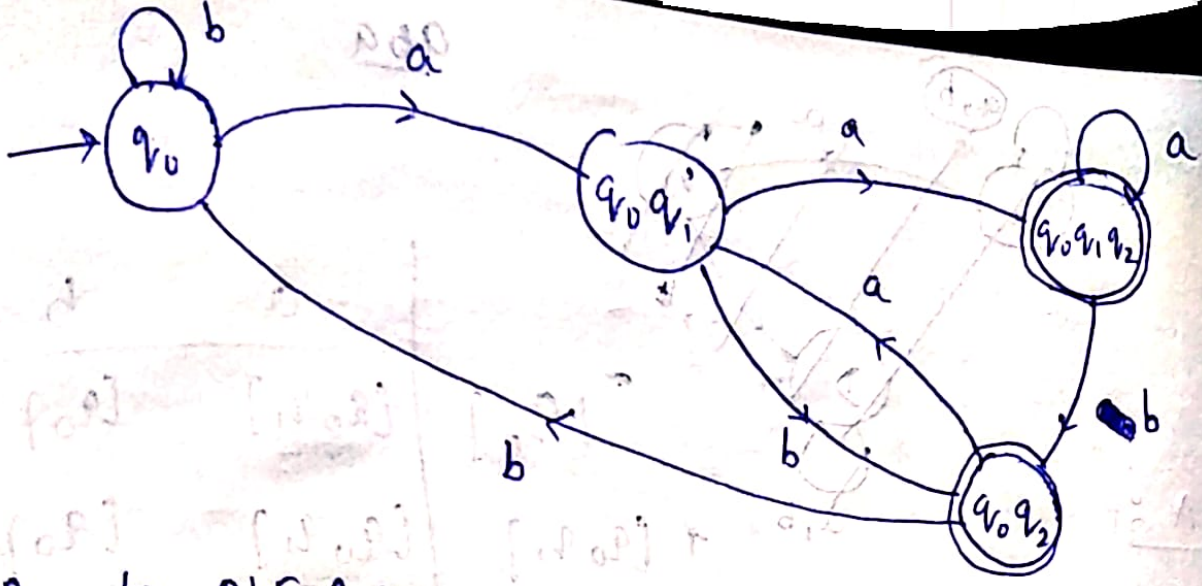
$\delta^*(q_0, a) \cup \delta^*(q_1,$

$q_0 q_1 \cup q_2$

$\delta^*(q_0, b) \cup \delta^*(q_1, b$

$q_0 q_2$

q_0 q_2

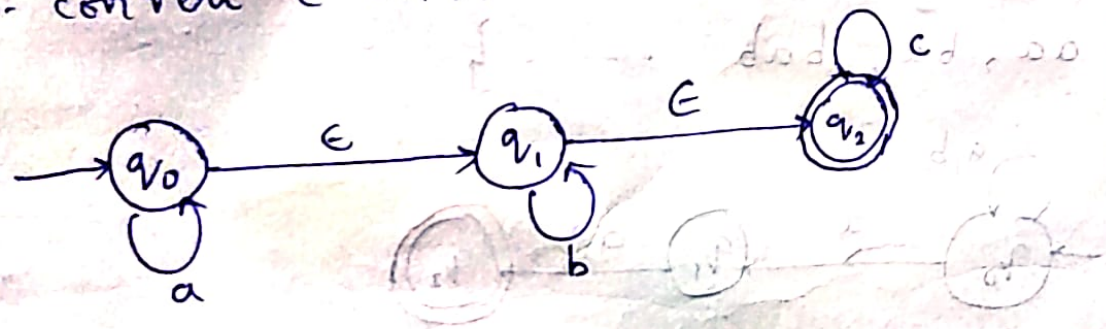


∈-NFA to NFA

function : ∈-closure

∈-closure (q) = set of all those states of NFA (with ∈-transitions) which can be reached from q on a path labeled by ∈ (i.e., without consuming an input symbol), including q itself.

Eg :- convert ∈-NFA to NFA



transition table

δ | ∈	a	b	c	∈
→ q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
			q_2	ϕ

// Do not perform compliment operation on NFA.

closure $(q_0) = \{q_0, q_1, q_2\}$.

closure $(q_1) = \{q_1, q_2\}$.

closure $(q_2) = \{q_2\}$.

Initial state of NFA (without ϵ-closure) will be as below (Here q_0 is initial state for ϵ-NFA).

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

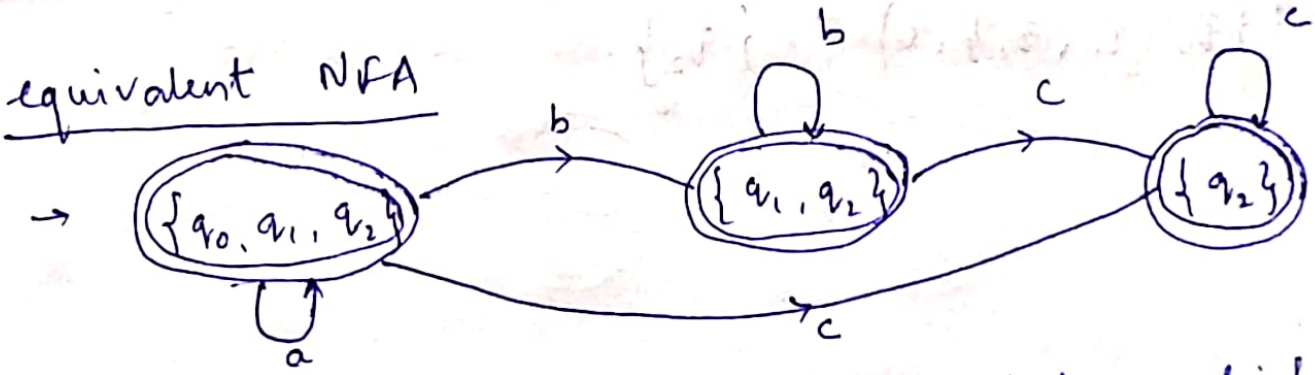
$\Rightarrow$ initial state for NFA $= \{q_0, q_1, q_2\}$

Rest states of NFA are :-

ϵ-closure $(q_1) = \{q_1, q_2\}$ { second state for NFA

ϵ-closure $(q_2) = \{q_2\}$ { Third state for NFA }

equivalent NFA



$\Rightarrow$ Final states of NFA are those new states which contains final state of ϵ NFA as a member.

Do following for transition over each alphabet for each state (δ is the transition function).

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

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

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

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

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

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

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

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

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

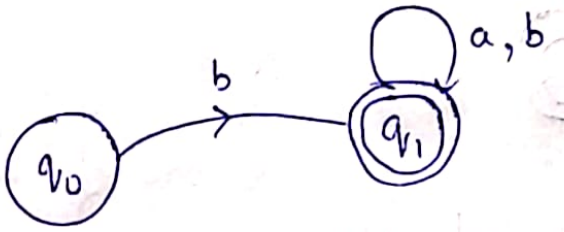
$= \{q_1, q_2\}$

$\delta'(\{q_0, q_1, q_2\}, c) = \{q_2\}$

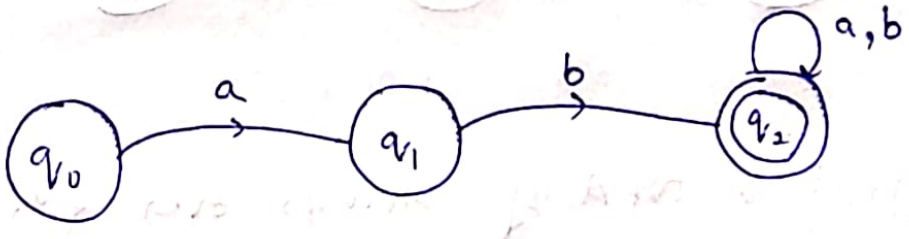
Questions on NFA :-

Q) Construct a NFA for strings not starting with 'a' over {a, b}.

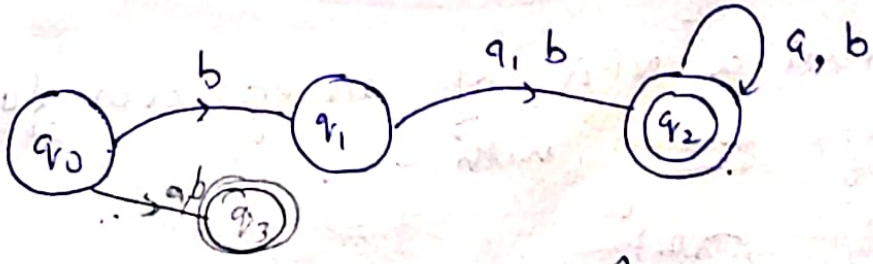
$L = \{baaa, bbaa, \dots\}$



Q) Construct a NFA for strings over {a, b} in which each string of the language starts with 'ab'?

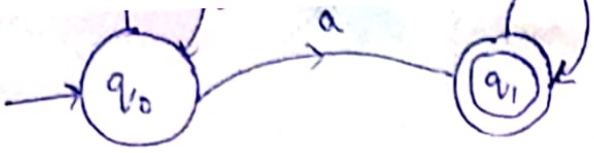


Q) Construct a NFA for strings over {a, b} with strings not starting with 'ab'?

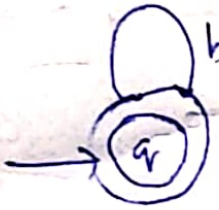


$L = \{bab\dots, baab\dots\}$

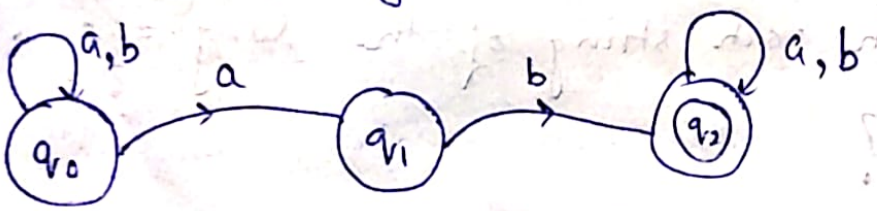
Q) Construct a NFA for strings over {a, b} with 'a' as



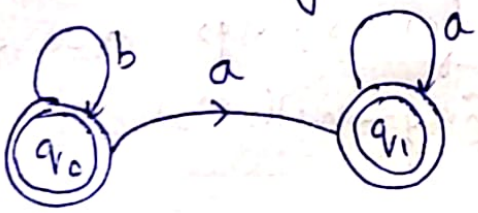
Q) Construct a NFA of strings over {a, b} with str not containing 'a' as substring



Q) Construct a NFA of strings over {a, b} with strings containing 'ab' as substrings.



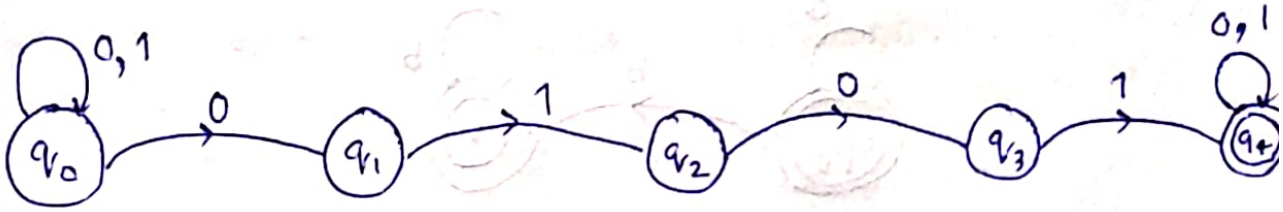
Q) Construct a NFA of strings over {a, b} with strings not containing 'ab' as substring.



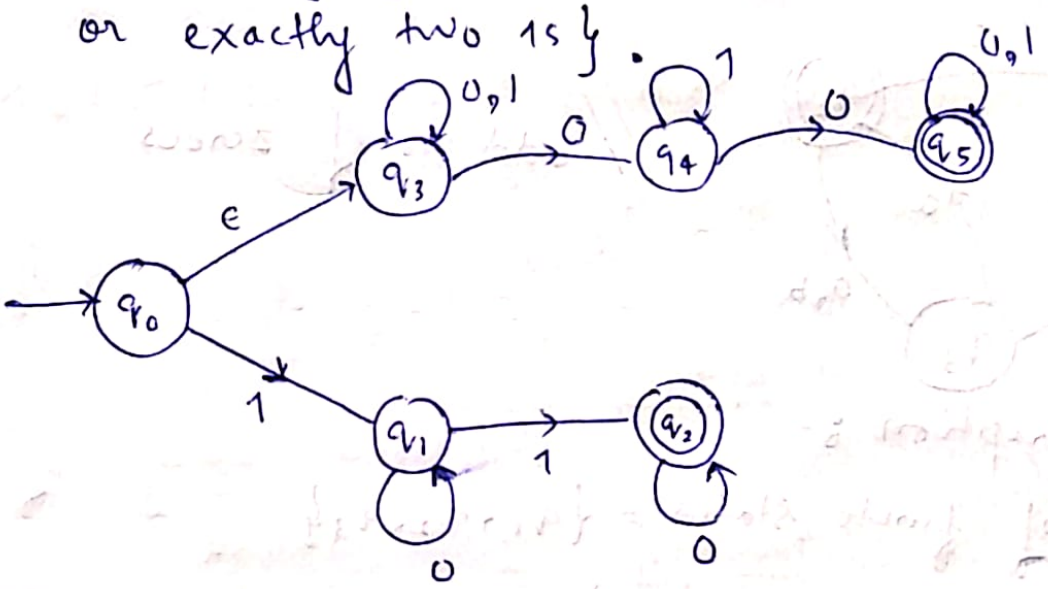
Q) Construct a NFA of strings over {0, 1} with strings ending with 00.

Q) The language $\{w \in \Sigma^* \mid w$ contains the substring 0101, over $\Sigma = \{0, 1\}$.

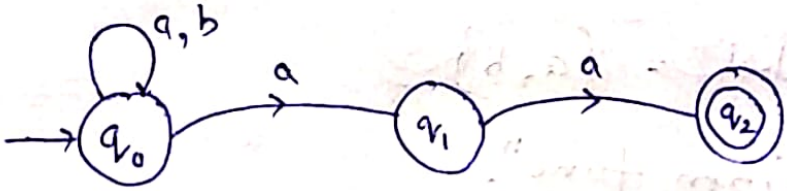
Sol)



Q) The language $\{w \in \Sigma^* \mid w$ contains atleast two 0s or exactly two 1s$\}$.

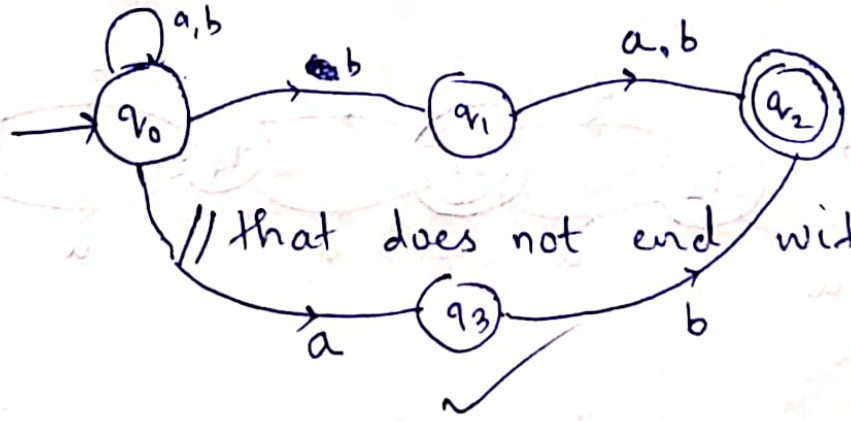


Q) All strings that do not end with 'aa', over $\{a, b\}$



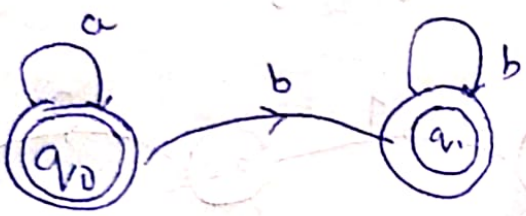
// that end with 'aa'.

Sol)



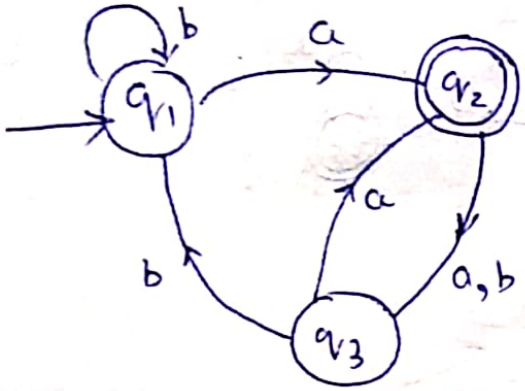
// that does not end with 'aa'

Q) All strings which do not contain substring 'ba'.



Sipser - Chapter 1

Q1)



// odd no of zeroes

Formal description :-

Q = set of finite states = $\{q_1, q_2, q_3\}$

Q_0 = initial state = $\{q_1\}$

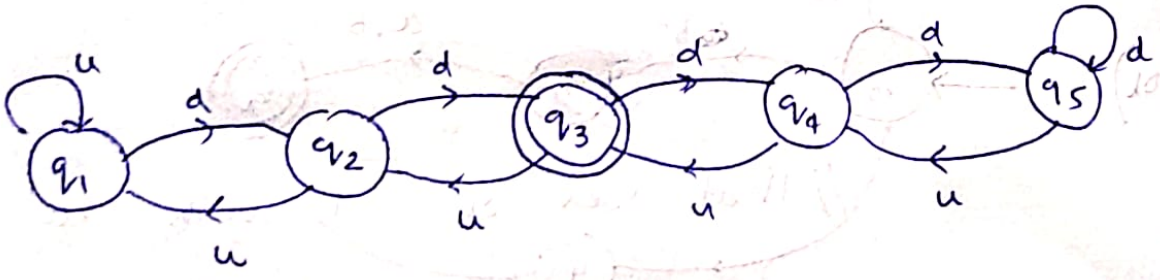
Q_F = final state = $\{q_2\}$

Σ = alphabet = $\{a, b\}$

δ = transition func^n.

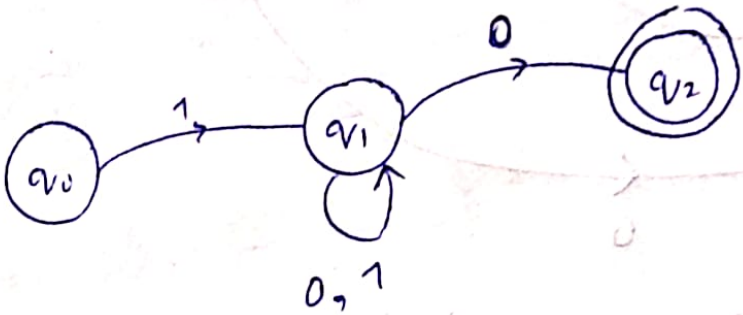
$M_1 = \{\{q_1, q_2, q_3\}, \{q_1\}, \{q_2\}, \{a, b\}, \delta\}$.

Q.3)



Q) i) NFA:

//



	0	1
q_0	D	q_1
q_1	$q_1 q_2$	q_1
D	D	D
$q_1 q_2$	$q_1 q_2$	q_1

$\delta^*(q_1 q_2, 0) = \delta^*(q_1, 0) \cup \delta^*(q_2, 0)$

$= q_1 q_2 \cup \{\}$

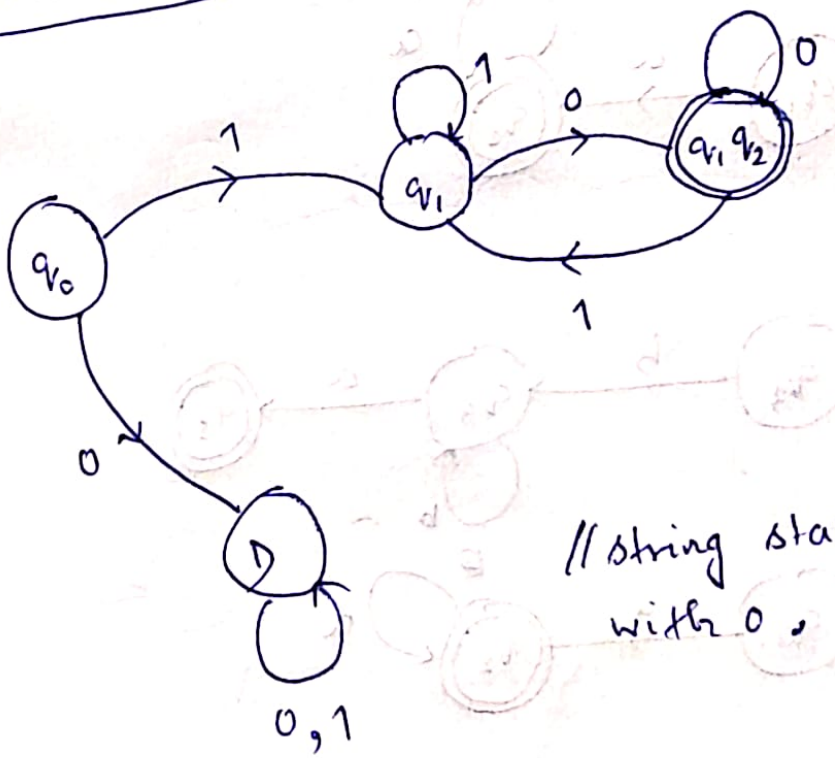
$= q_1 q_2$

$\delta^*(q_1 q_2, 1) = \delta^*(q_1, 1) \cup \delta^*(q_2, 1)$

$= q_1 \cup \{\}$

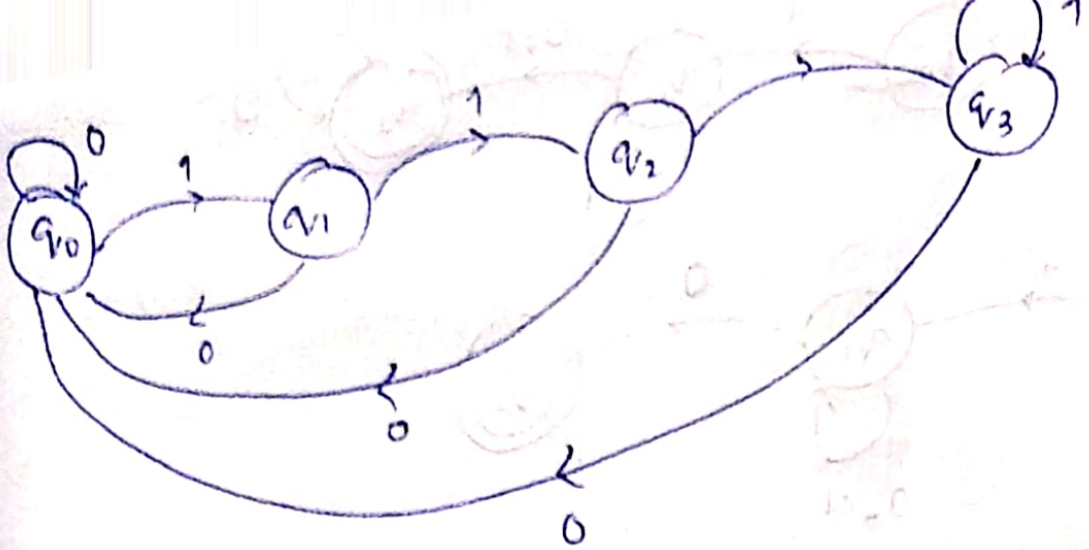
$= q_1$

Equivalent DFA:



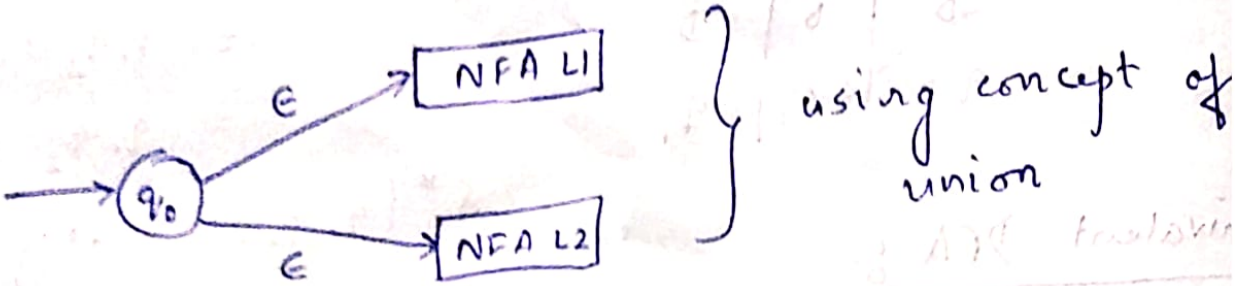
// string starts with 1 & ends with 0.

Q) w/w contains atleast 3 1's.

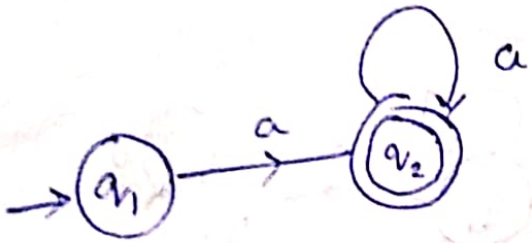


Notes: L4 : Design ϵ-NFA for $L = \{a^n : n \geq 1\}$

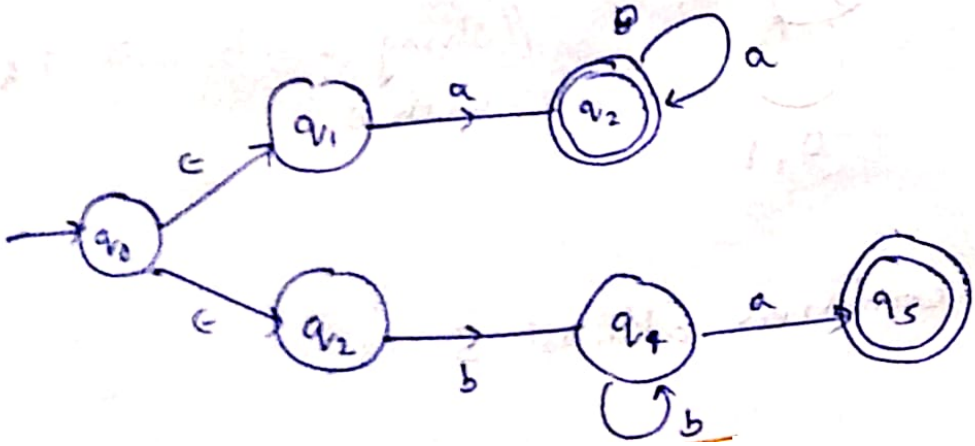
$\{b^m a : m \geq 1\}$



for L1,



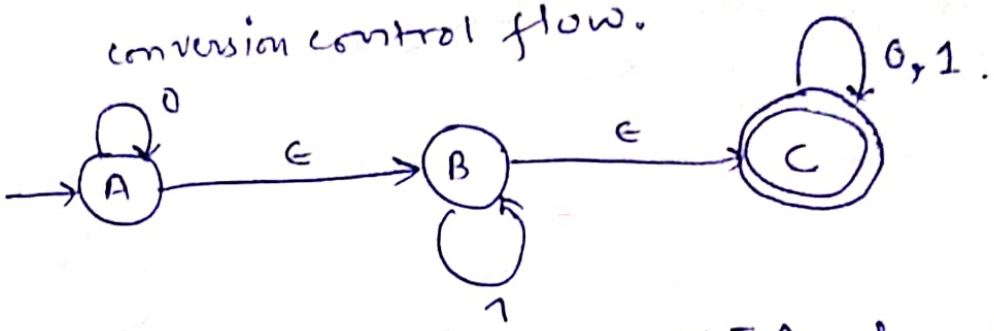
for L2,



// ∈-NFA ⟹ NFA ⟹ DFA

conversion control flow.

Q)



Convert the following ∈-NFA to equivalent NFA ?