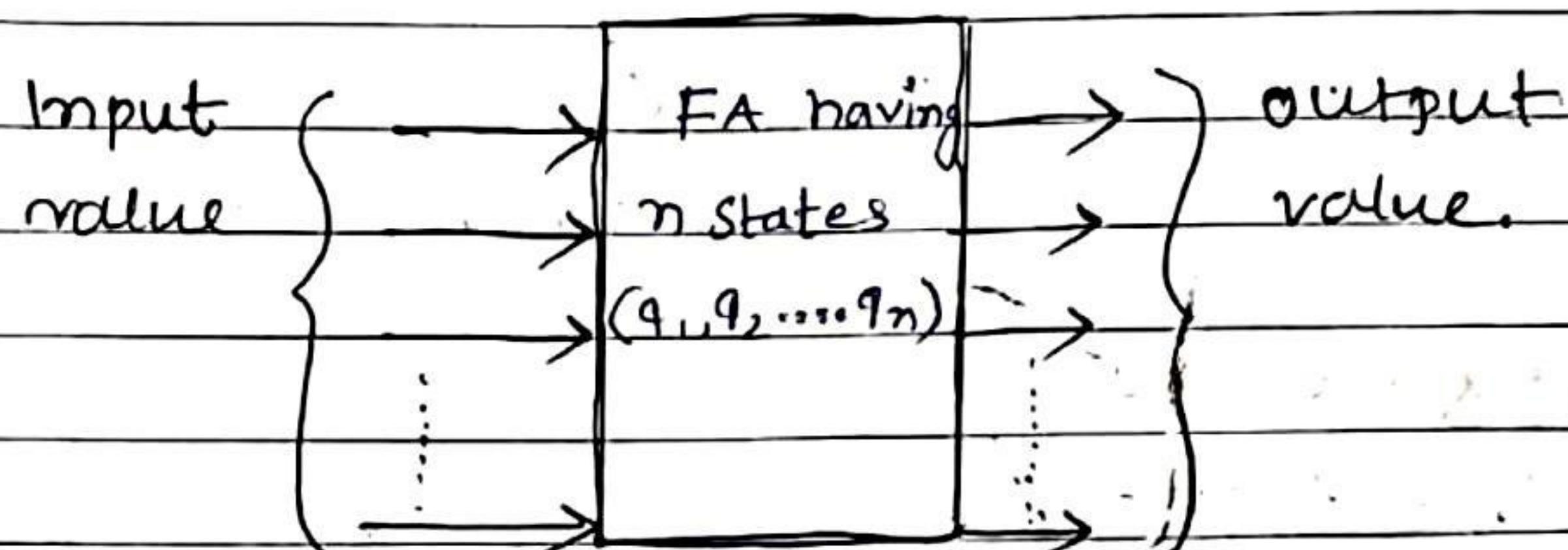


Regular Expression and Finite Automata.

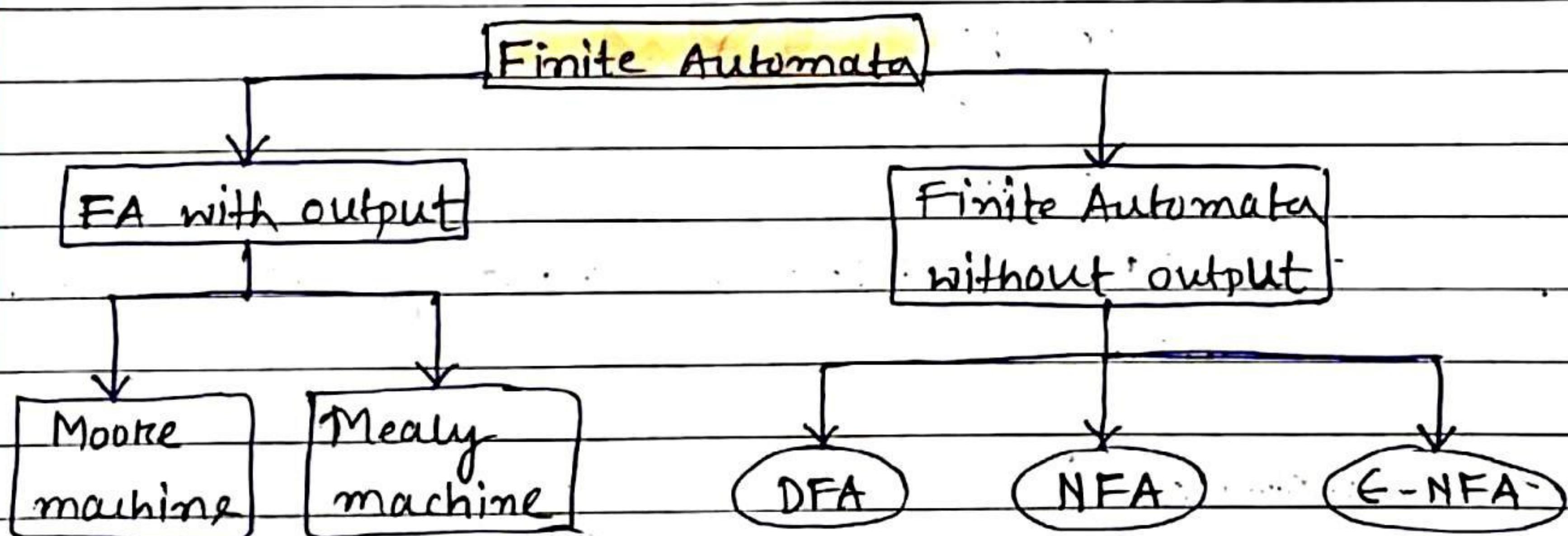
- Finite Automata :-

→ A finite state machine (FSM) or finite state automata is an abstract machine used in the study of computation and language that has only a finite, constant amount of memory.



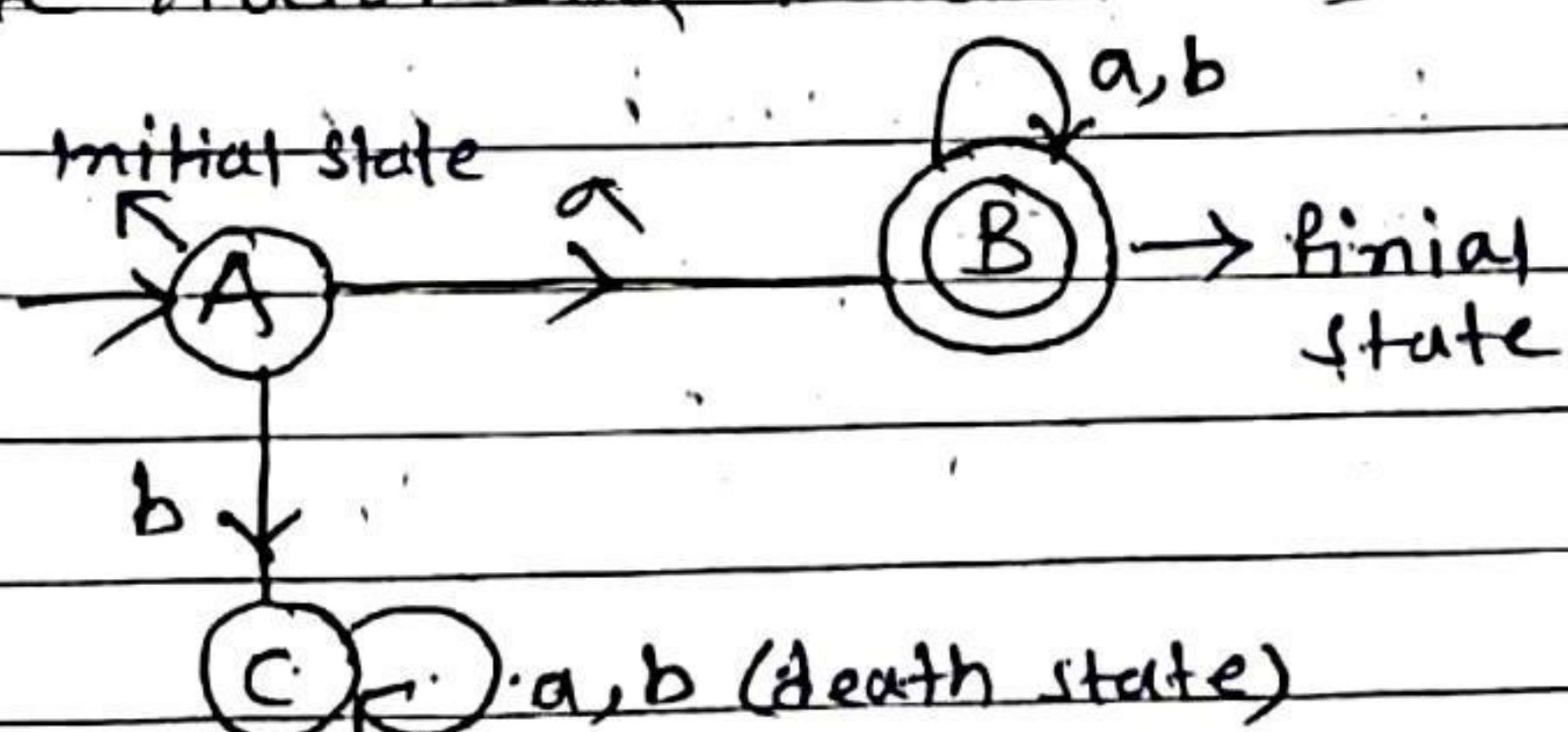
Model of finite automata

- Types of Finite Automata :-



- DFA (Deterministic Finite Automata) :-

→ A finite automata is defined as 5-tuples $(Q, \Sigma, \delta, q_0, F)$.



Where,

$Q = \text{set of all states} \Rightarrow \{A, B, C\}$

$\Sigma = \text{Input} \Rightarrow \{a, b\}$

$q_0 = \text{Initial state} \Rightarrow 'A'$

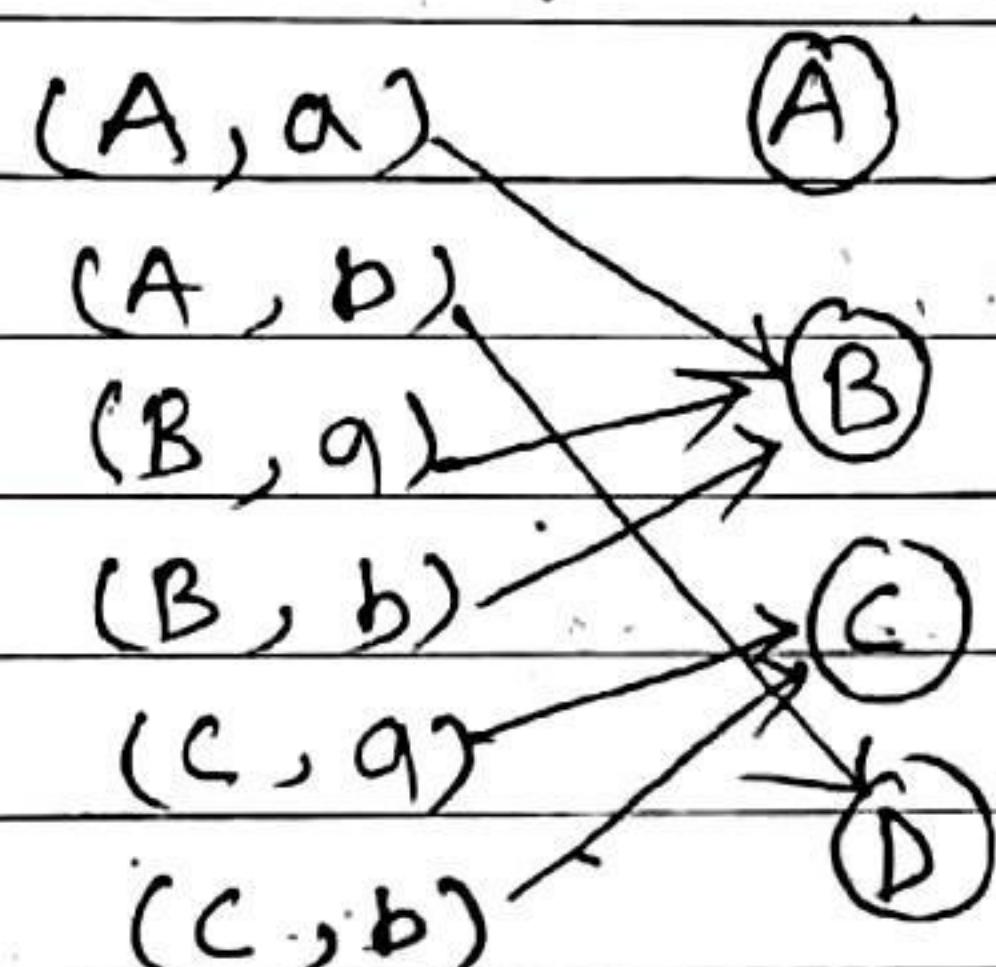
$F = \text{final state} \Rightarrow \{B\}$
↳ set of.

$\rightarrow Q$ is the superset of F . ($F \subseteq Q$)

$\delta = \text{transition function which maps } Q \times \Sigma \text{ into } Q$.

$$\delta: Q \times \Sigma \rightarrow Q$$

$$\{A, B, C\} \times \{a, b\}$$



\rightarrow In DFA, a state with 1 input go into another one state.

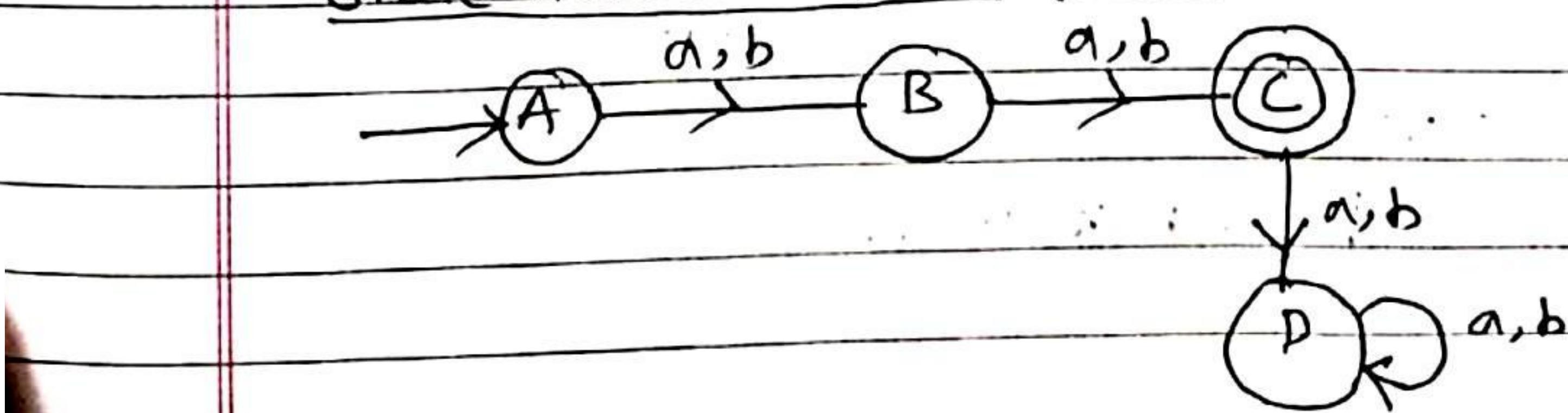
(Example-1)

construct a DFA that accept set of all string over $\{a, b\}$ of length '2'.

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

language, $L_1 = \{aa, ab, ba, bb\}$.

State transition Diagram -



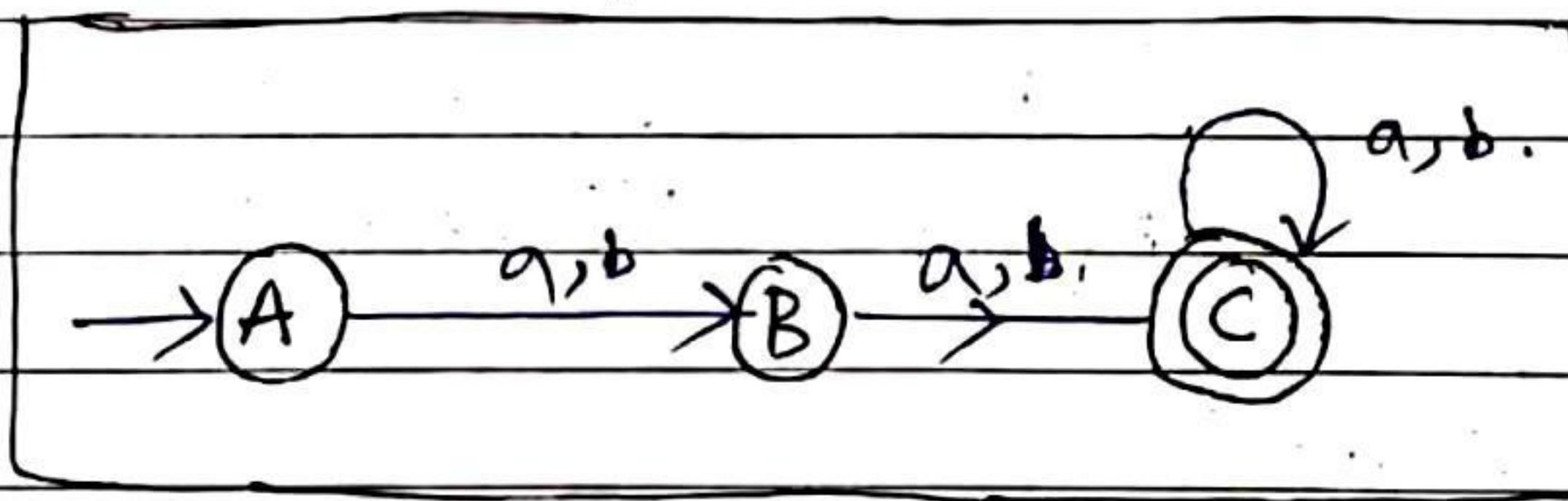
- String accept \rightarrow Scan the entire string, if we reach in a final state from initial state. Then string will be accepted by FA.
- Language accept \rightarrow A Finite Automata is said to accept a language if all the string in the language are accepted and the string not in the language are rejected.

Ex-2

Construct a DFA which accepts all the string $\{a, b\}^*$ where the string length is ^{at least} '2'. $|w| \geq 2$

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

$$L = \{\underline{aa}, ab, ba, bb, aaa, aba, baa, bba, \dots\}$$



State transition Diagram,

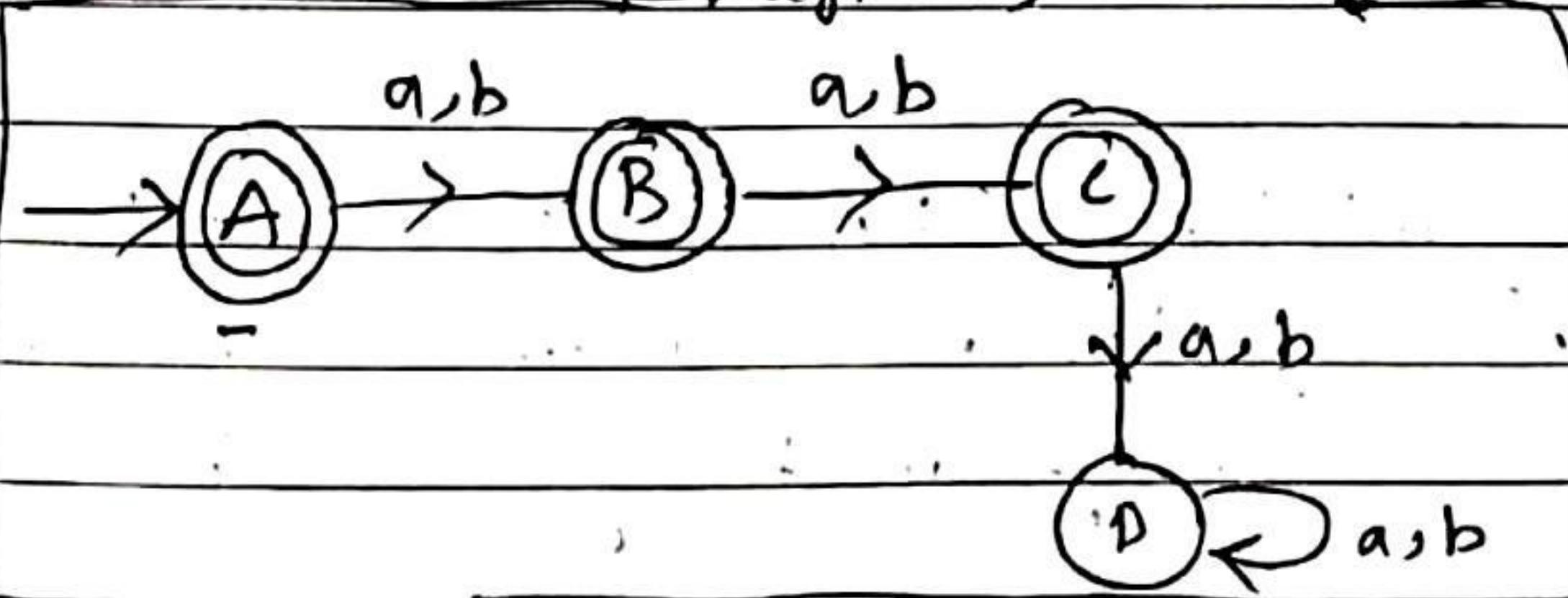
Ex-3

Construct a DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$, $|w| \leq 2$.

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

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

State transition Diagram,



Q8/9 $w \rightarrow \text{string}$

$\textcircled{*} |w|=2, |w| \geq 2, |w| \leq 2.$ (gate)

→ When, string length $|w|=n$, then
no of state required ($n+2$).

When, $|w| \geq n$, then

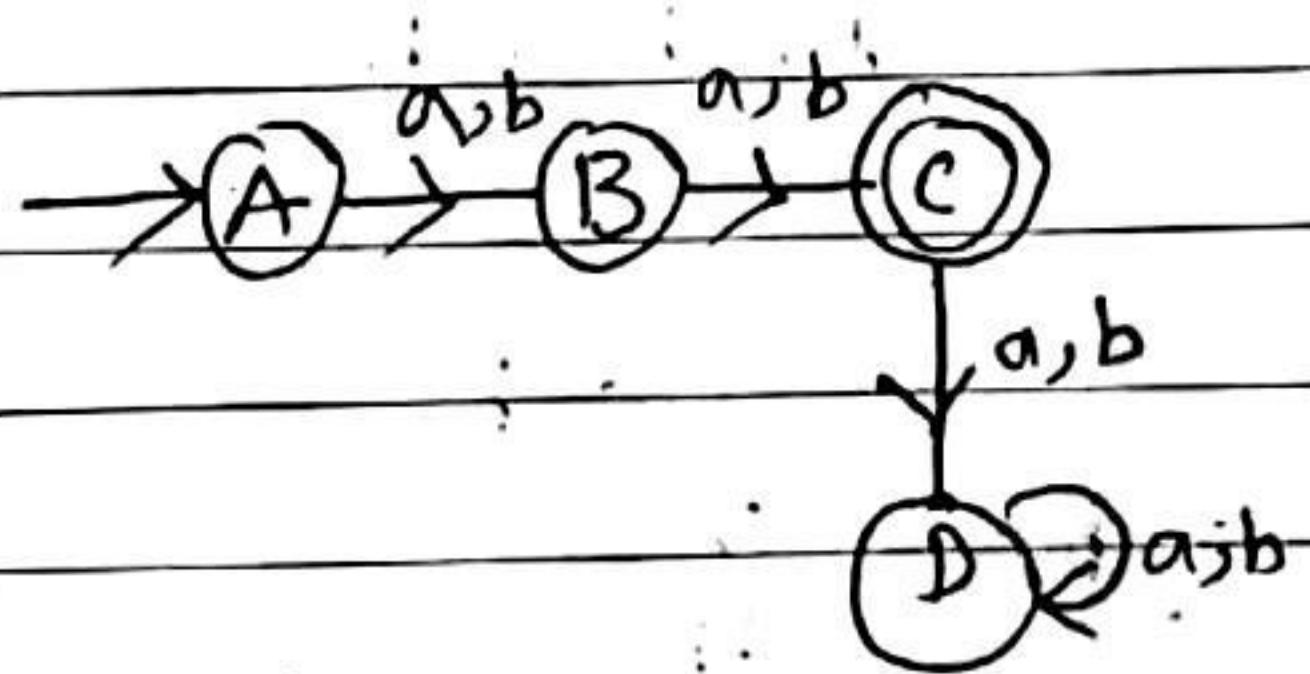
no of state required ($n+1$)

When, $|w| \leq n$, then

no of state required ($n+2$)

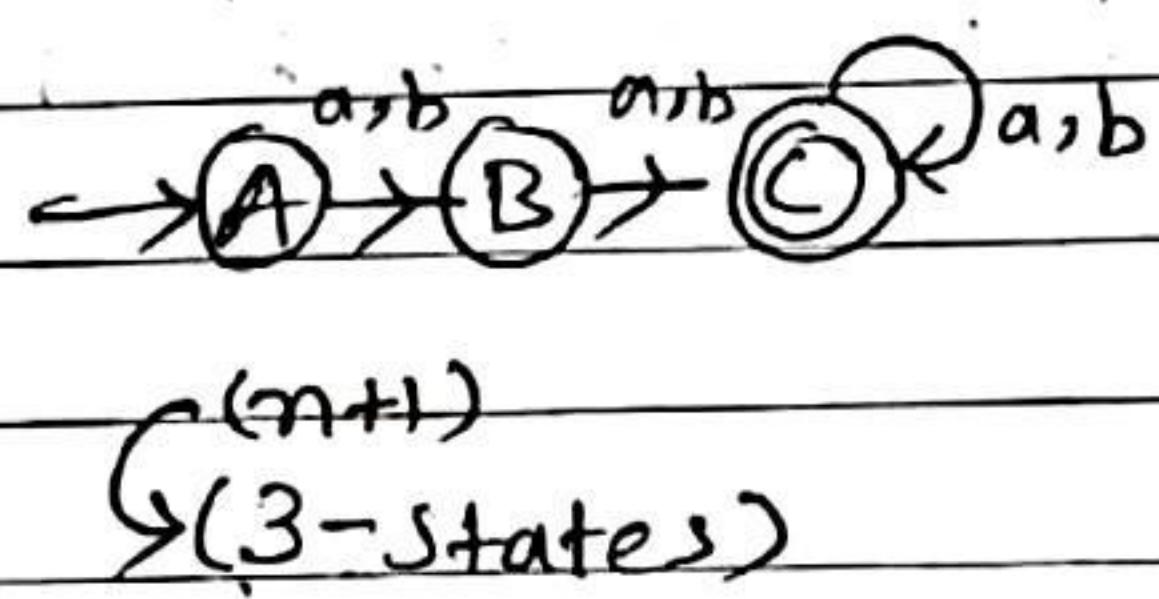
• $|w|=2$

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



• $w \geq 2$

$$L = \{aa, ab, auaa\ldots\}$$

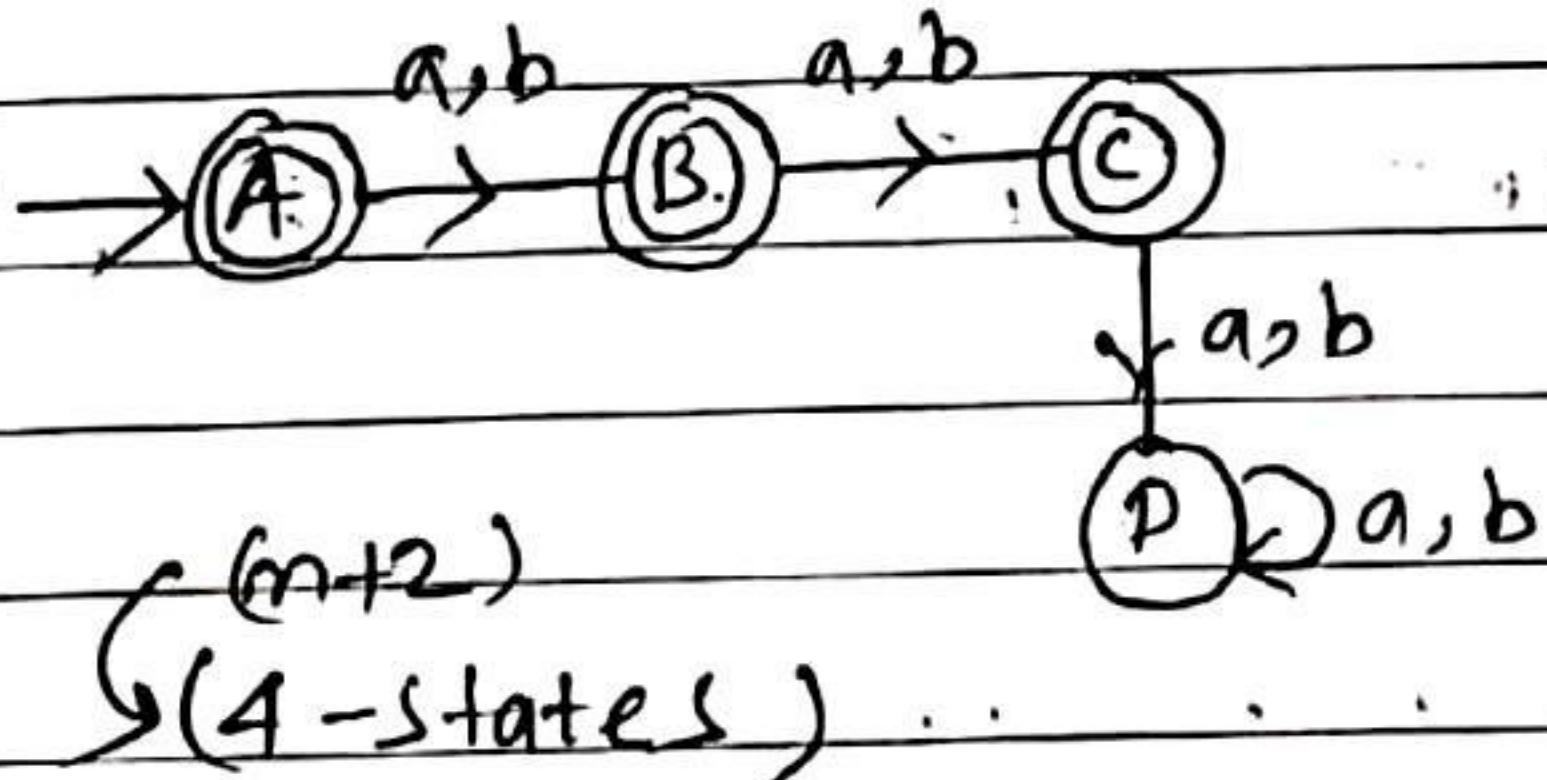


($n+2$)

↙ (4-states)

• $|w| \leq 2$

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



($n+2$)

↙ (4-states)

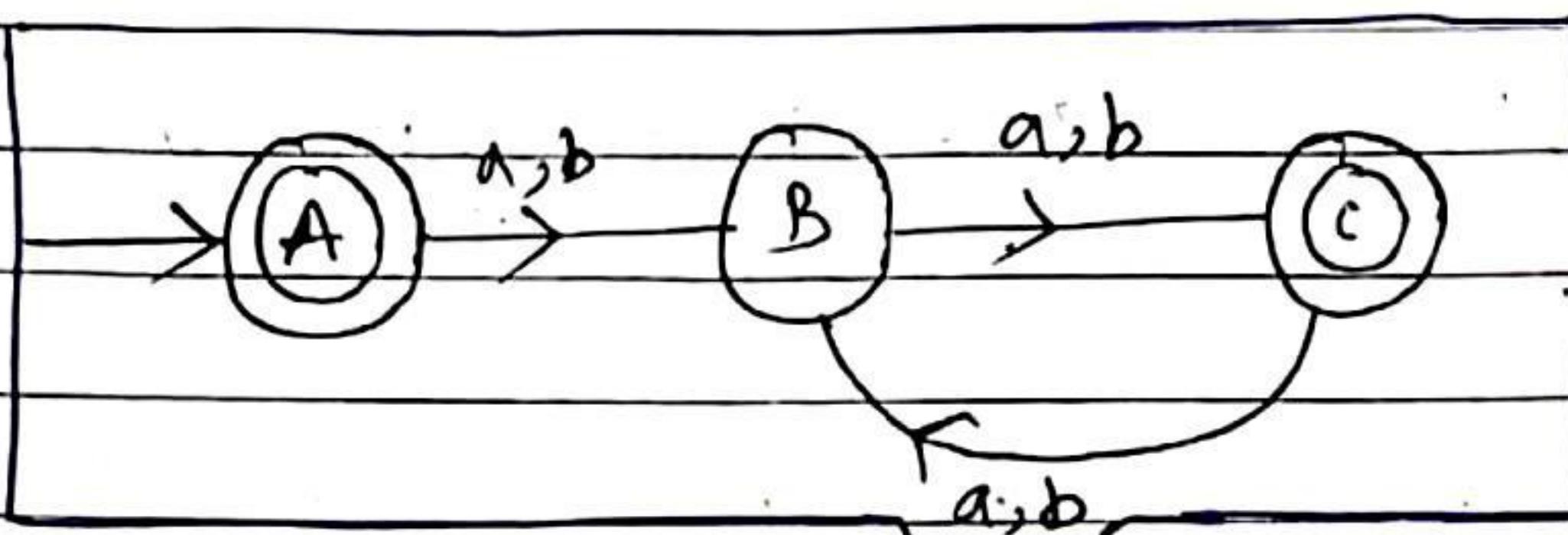
Example] - 4

construct a minimal DFA which accept all string over $\{a, b\}$, $|w| \bmod 2 = 0$ (means the length of the string
length of string will be even)

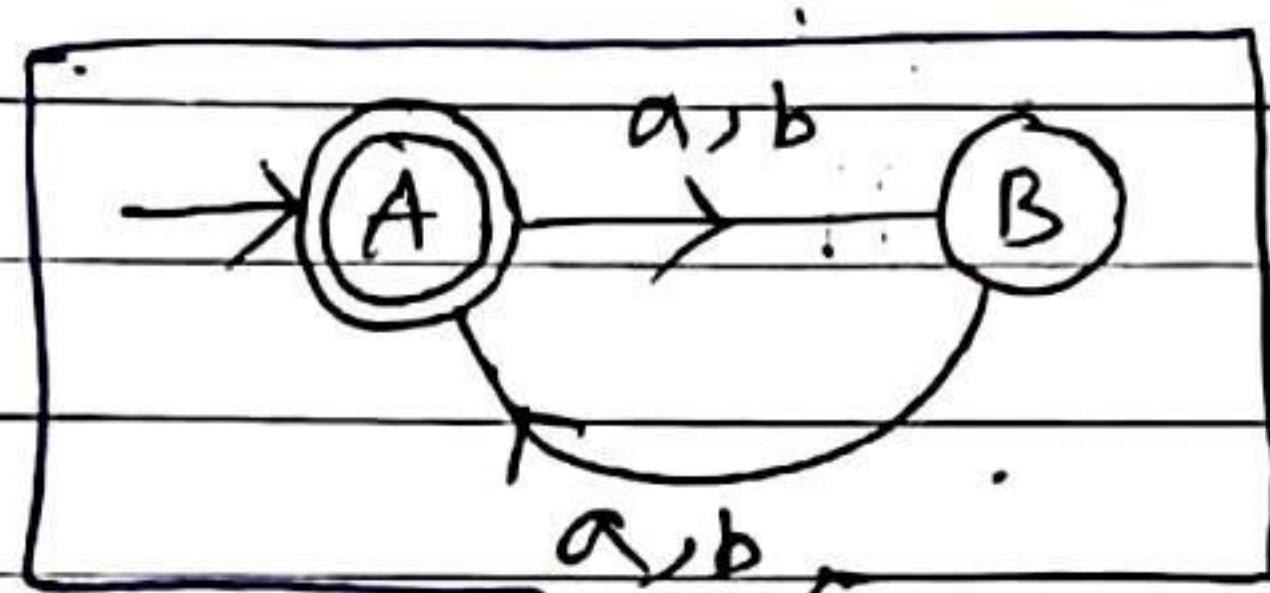


$$\Sigma = \{a, b\}$$

$$L = \{aa, ab, ba, bb, aaaa, \dots, bbbb, \dots\}$$



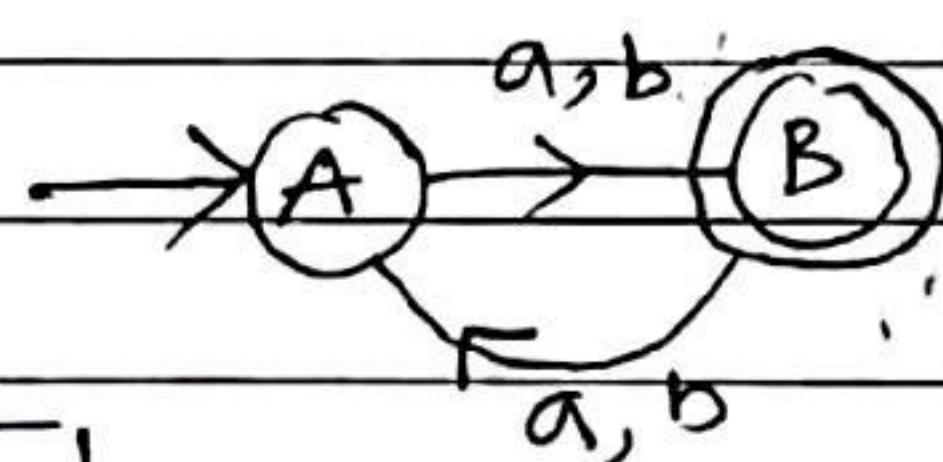
Q5



(State transition Diagram)

$$[Ex-5], |w| \bmod 2 = 1$$

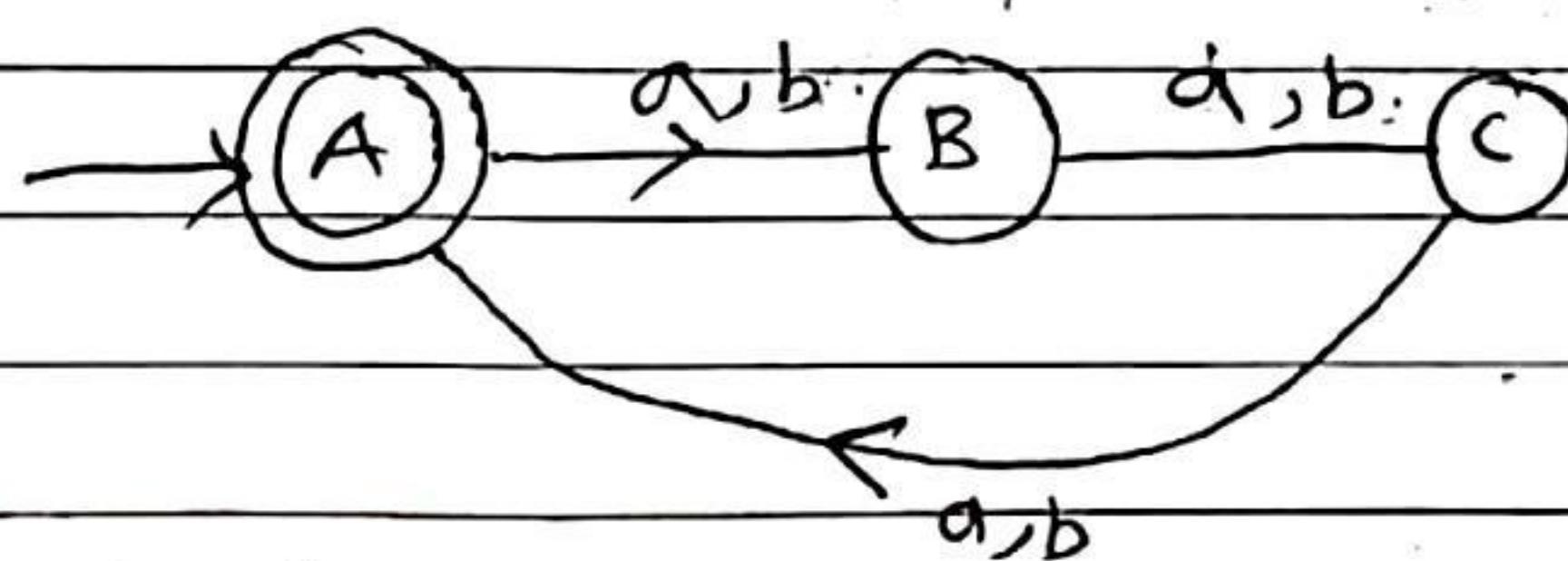
$$\rightarrow L = \{a, b, aaa, bbb, \dots, aaaa, \dots\}$$



[Ex-6]

$$|w| \bmod 3 = 0$$

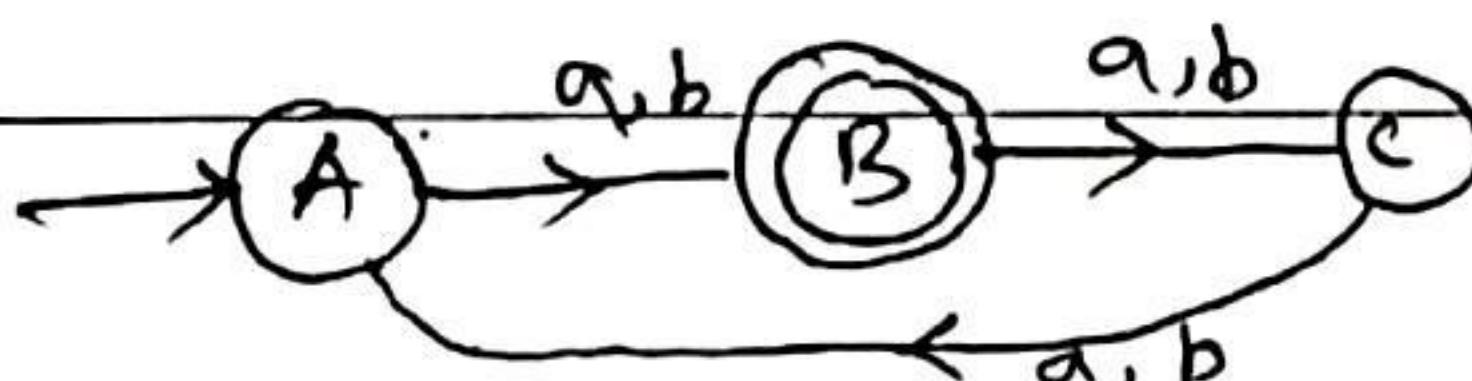
$$\rightarrow L = \{a, b, aaa, aba, bbb, \dots\}$$



final state 'A'

$$[Ex-7], |w| \bmod 3 = 0$$

$$\rightarrow L = \{a, b, aaaa, bbbb, \dots\}$$



* If $|w| \bmod n = 0$, then no of states are ' n '.

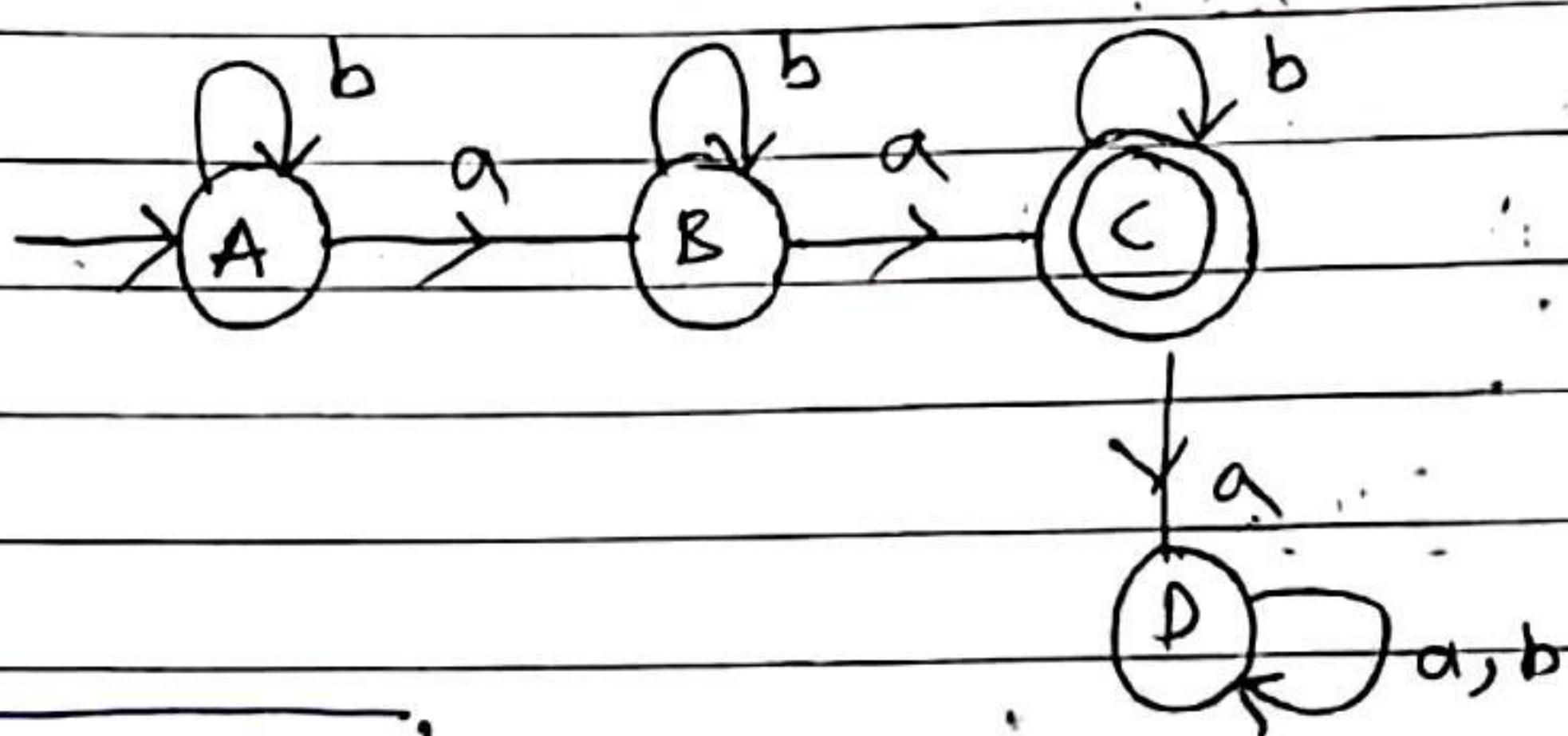
Example - 8

construct a minimal DFA, that accept $w \in \{a, b\}^*$

$$1. [n_a(w) = 2]$$

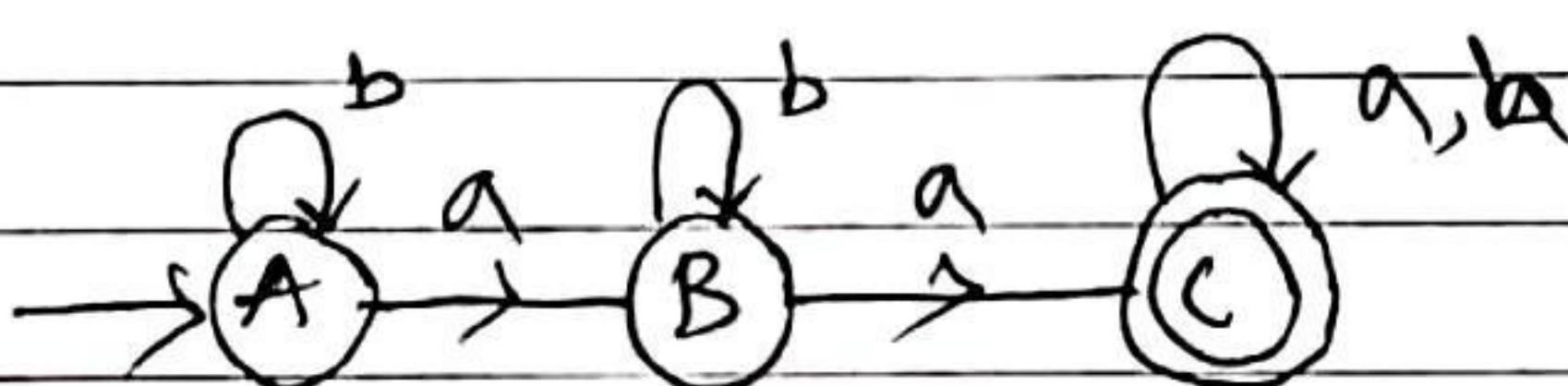


$$L = \{aa, aab, baa, aba, bbaa, \dots\}$$



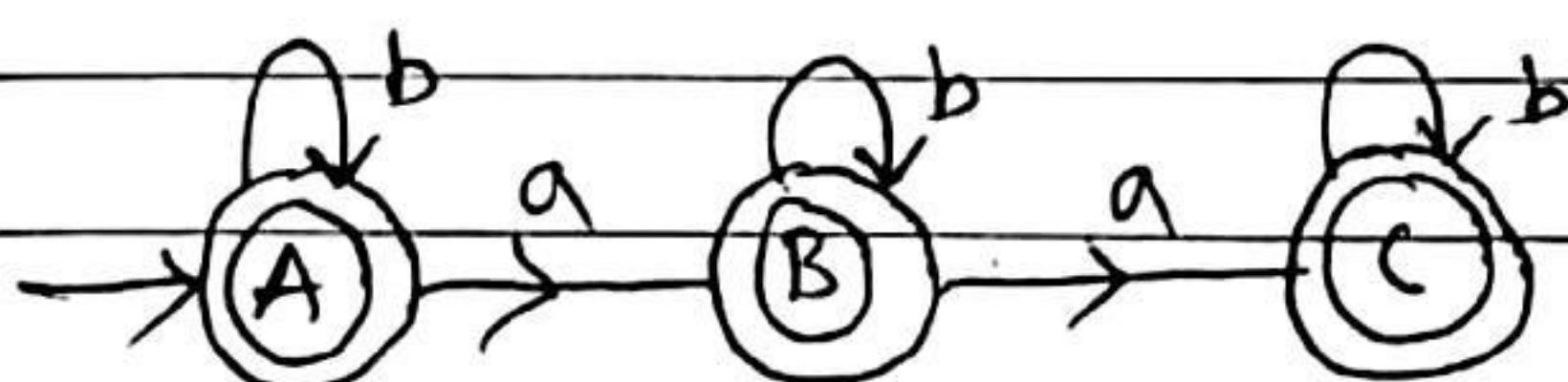
$$2. [n_a(w) \geq 2]$$

$$\rightarrow L = \{aa, aaab, aaaaba, \dots\}$$



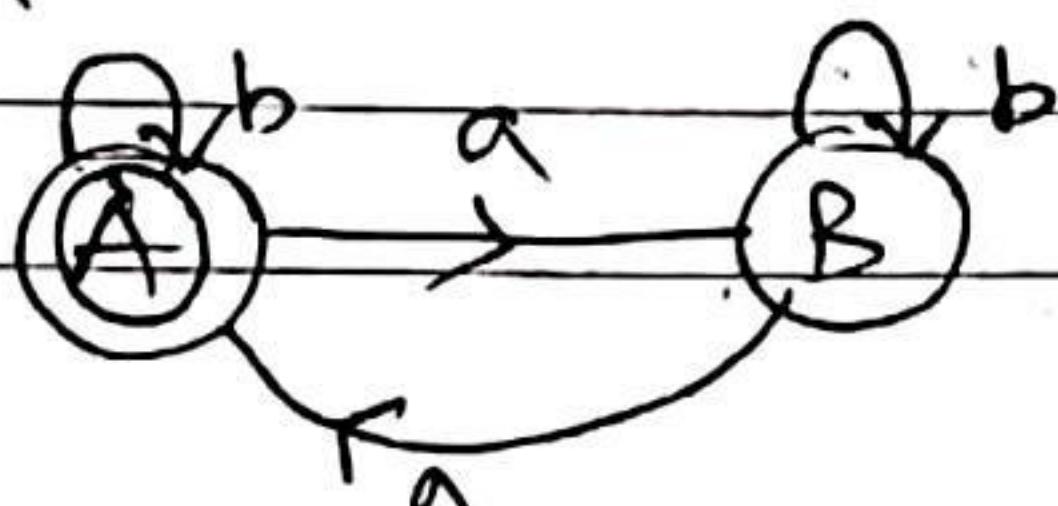
$$3. [n_a(w) \leq 2]$$

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



$$4. [n_a(w) \bmod 2 = 0]$$

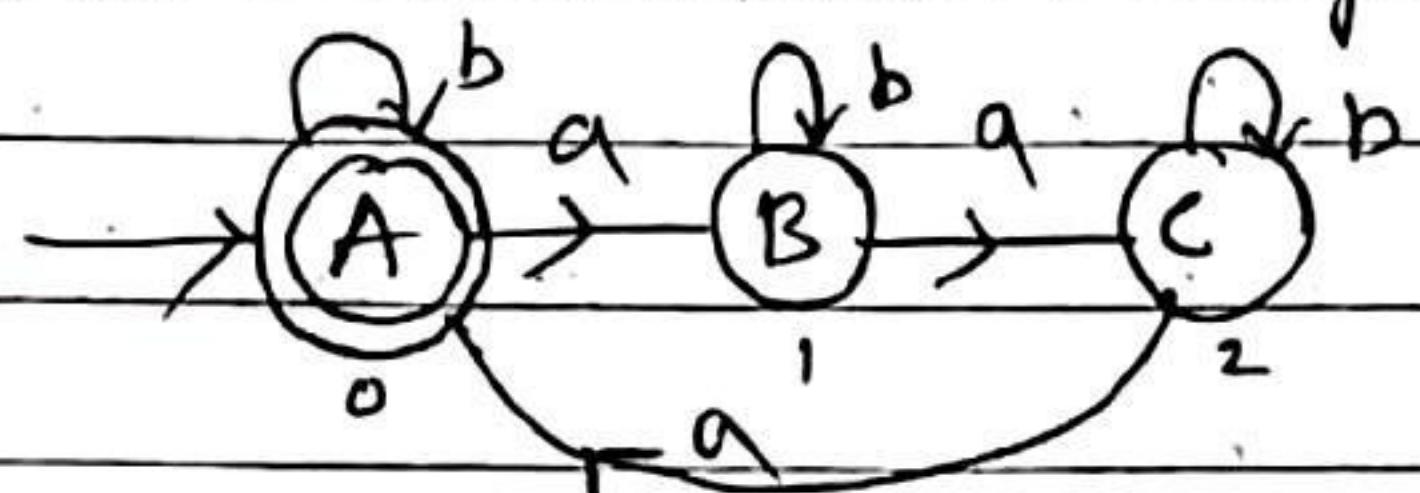
$$\rightarrow L = \{aa, aab, aabb, aaaa, aaaaaa, \dots\}$$



5. $n_a(w) \bmod 3 = 0$

$\rightarrow L = \{aaa, \dots aaaaaaa \dots\}$

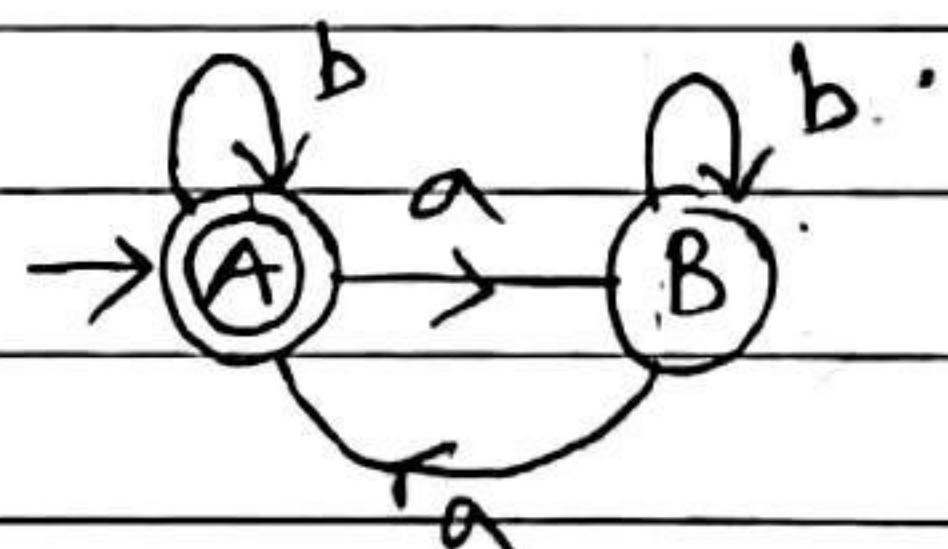
state transition diagram -



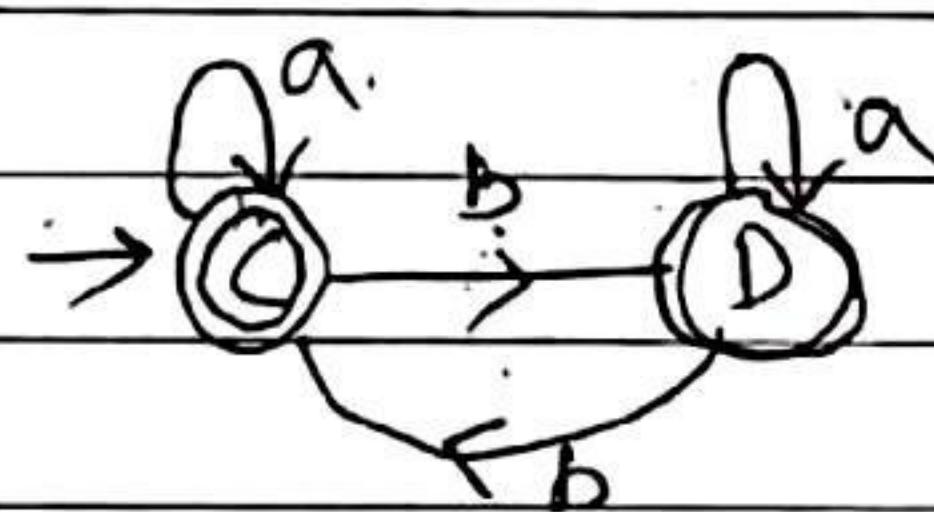
6. construct a minimal DFA, where $w \in \{a, b\}^*$

$$\left. \begin{array}{l} n_a(w) \equiv 0 \pmod{2} \\ n_b(w) \equiv 0 \pmod{2} \end{array} \right\}$$

$\rightarrow L = \{\epsilon, aa, bb, aabb, abab, bbaaaa, \dots bbbb \dots\}$



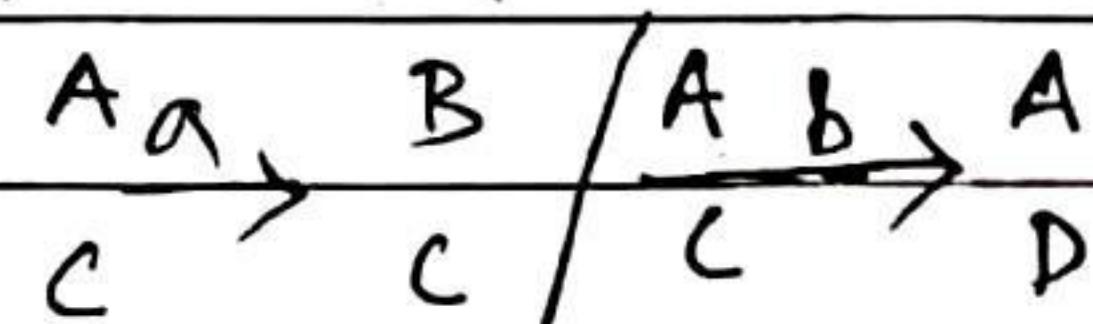
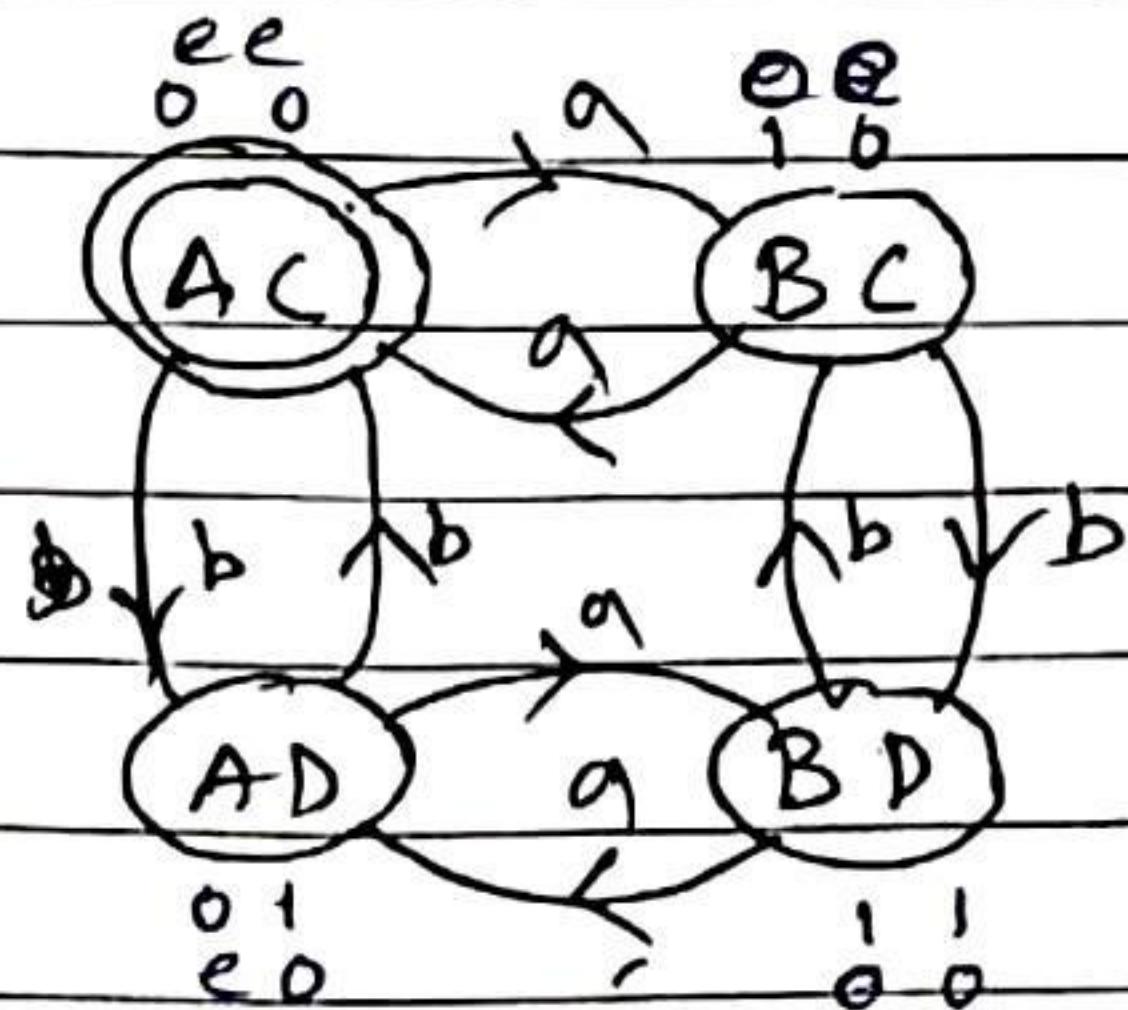
of state $\{A, B\}$



of state $\{C, D\}$

Cross product method -

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



$e \wedge e = \{ee\}$ (initial state)

$e \vee e = \{ee, e0, 0e\}$ state (final state).

**

→ If one Automata contain 'n' state and the other automata contain 'm' numbers of states then their cross product going to contain, (mn) states.

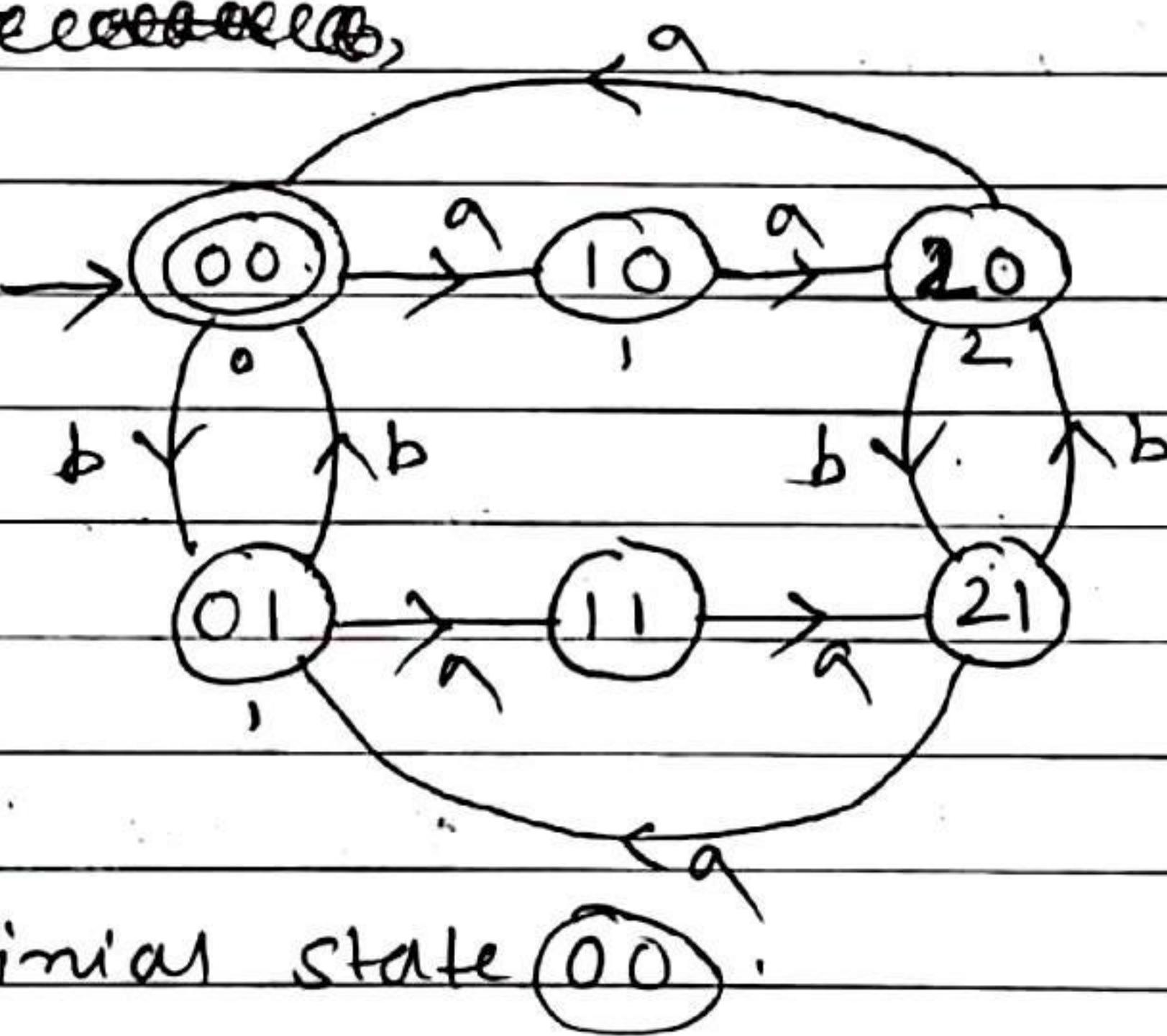
- ⑦ Construct a minimal DFA which accepts set of all strings over $\{a, b\}$ in which no of 'a's are divisible by 3 and no of 'b's are divisible by 2.

$$\rightarrow w \in \{a, b\}^*$$

$$n_a(w) \equiv 0 \pmod{3}$$

$$n_b(w) \equiv 0 \pmod{2}$$

~~Given~~



When, $n_a(w) \bmod 3 > n_b(w) \bmod 2$

final state = $\{10, 20, 11, 21\}$

or

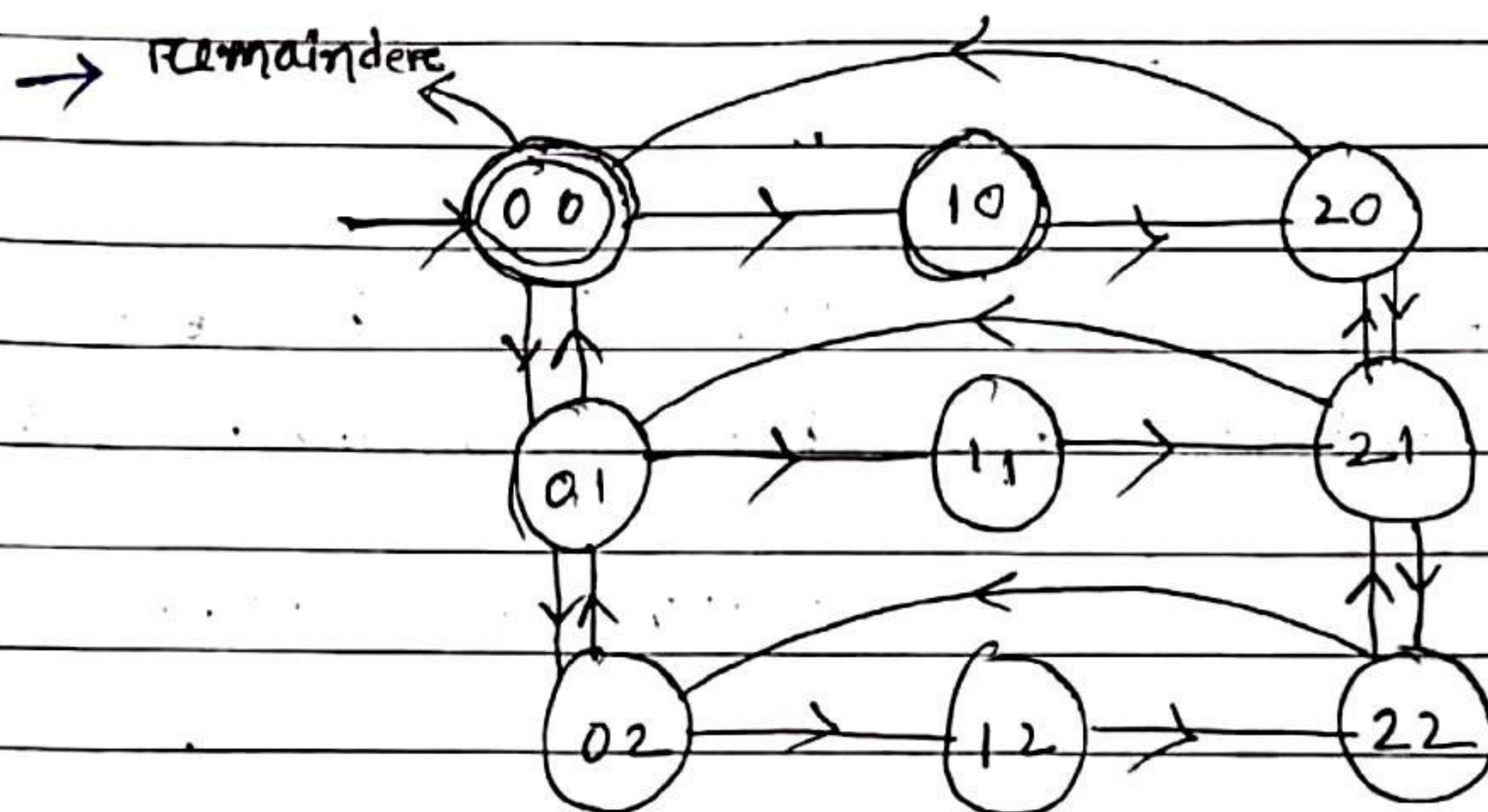
When, $n_a(w) \bmod 3 = n_b(w) \bmod 2$

final state = $\{00, 11\}$

⑧ Construct a minimal DFA, $w \in \{a, b\}^*$,

$$n_a(w) \equiv 0 \pmod{3}$$

$$n_b(w) \equiv 0 \pmod{3}$$



When, $n_a(w) \pmod{3} = 1$

&

$n_b(w) \pmod{3} = 2$, then

final state = $\{1, 2\}$

when, $n_a(w) \pmod{3} > n_b(w) \pmod{3}$, then

final state = $\{10, 20, 21\}$

$\rightarrow n_a(w) \equiv 0 \pmod{n}$

$n_b(w) \equiv 0 \pmod{m}$

Then the minimum no of states in automata is ' $m \times n$ '.

Example-2 - ①

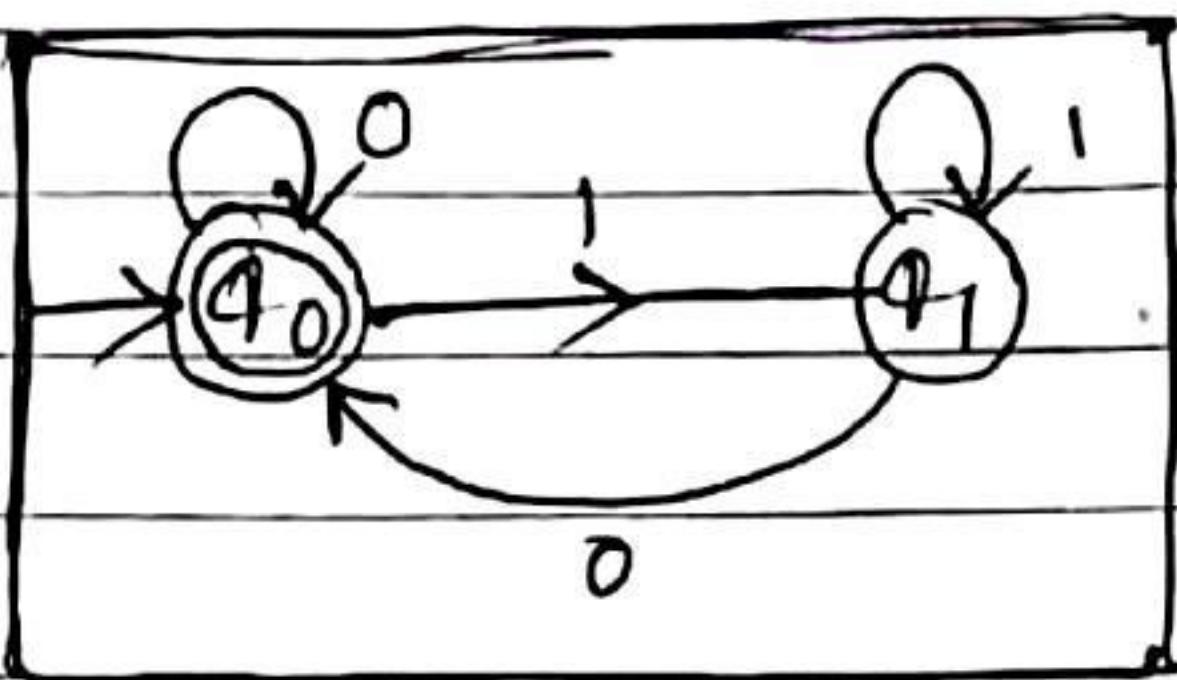
construct a minimal DFA, which accepts set of all strings over $\{0, 1\}$, which when interpreted as binary number is divisible by '2'.

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

$$w \in \{0, 1\}^*$$

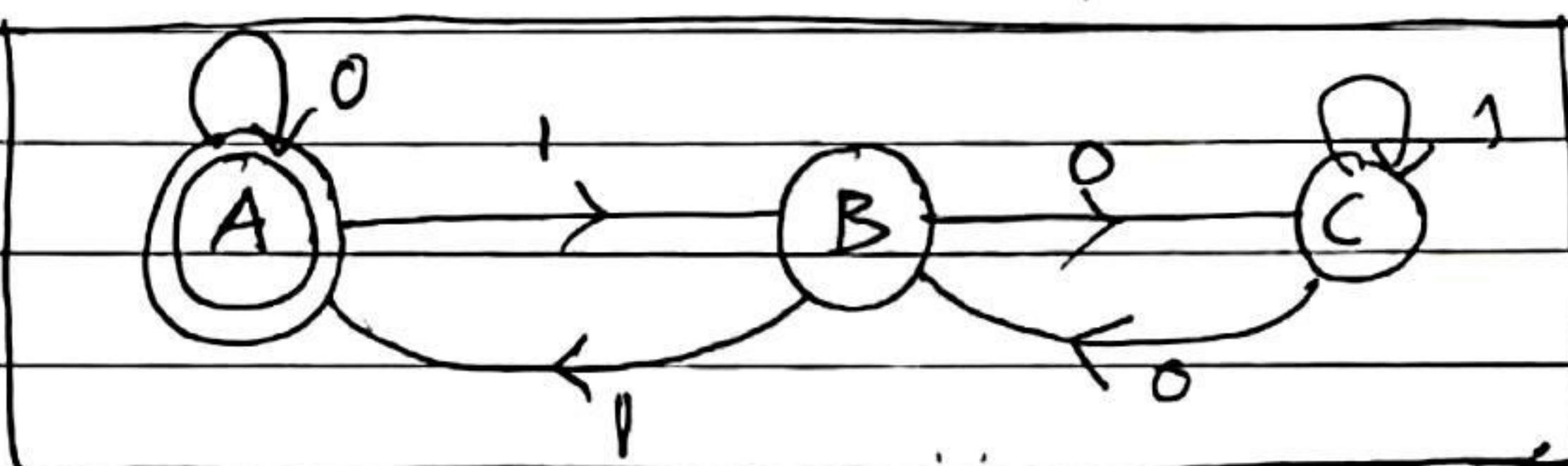
If we divide any no by 2 remainder can be '0' or '1'.

$$L = \{0, 00, 000, 0000, \dots, 10, 100, 110, \dots\}$$



- (2) construct a minimal DFA, $\Sigma = \{a, b\}$, ~~where~~ $w \in \{a, b\}^*$
 when interpreted as binary numbers is divisible by 3.

$$\rightarrow L = \{0, 00, 000, \dots, 11, 110, 1100, 1111, \dots\}$$



→ Finite Automata can represent in two ways
 ↪ State Diagram / transition Diagram.
 ↪ State transition table.

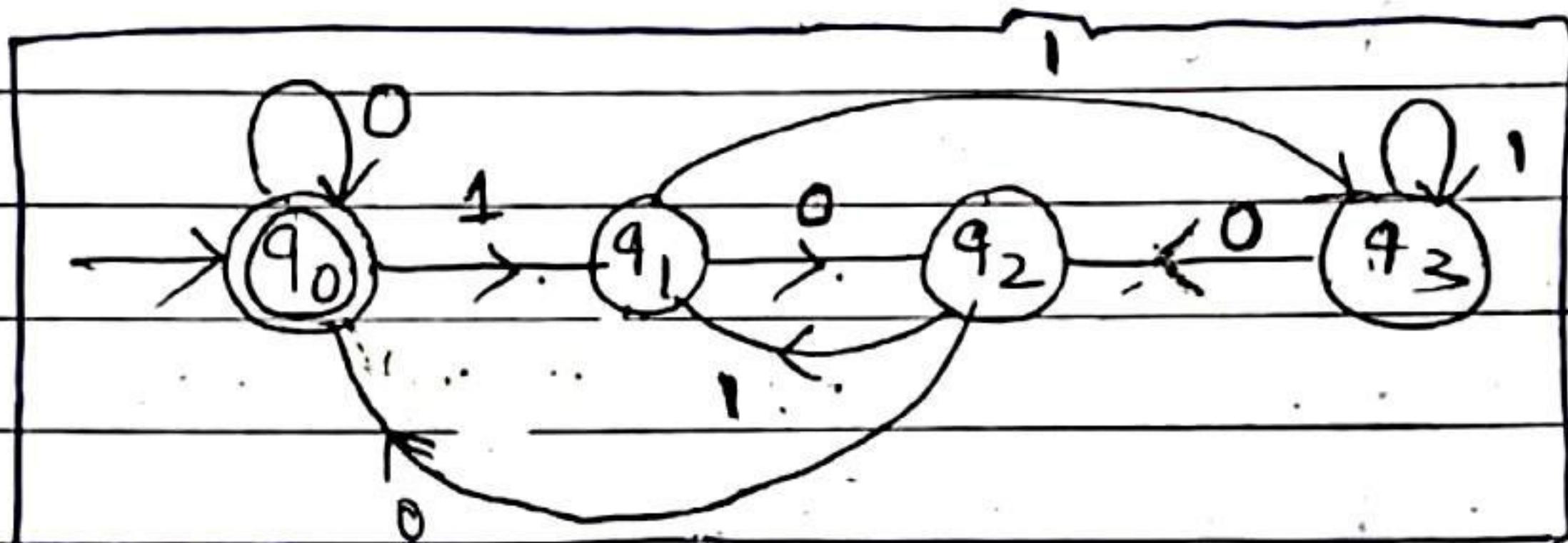
State transition table,

	0	1
$\rightarrow *A$	$A \rightarrow B$	
B.	C.	A
C.	$B \rightarrow e$	

③ Construct a minimal DFA, which accept all set of all strings over $\{0, 1\}$ which when interpreted as a binary number is divisible by 4.

$$\rightarrow \Sigma = \{0, 1\}, \text{ where } w \in \{0, 1\}^*$$

$$L = \{0, 100, 000, \dots, 100, 1000, 1100, 10000, 10100, \dots\}$$



State transition table -

	0	1
* q_0	$q_0 \xrightarrow{0} q_1$	$q_0 \xrightarrow{1} q_3$
q_1	$q_2 \xrightarrow{0} q_3$	
q_2	$q_0 \xrightarrow{0} q_1$	
q_3	$q_2 \xrightarrow{0} q_3$	

\rightarrow If the remainder is = 0
then FS $\rightarrow q_0$

remainder is = 1
 $FS \rightarrow q_1$

remainder is = 2
 $FS \rightarrow q_2$

remainder is = 3
 $FS \rightarrow q_3$

Example - 10

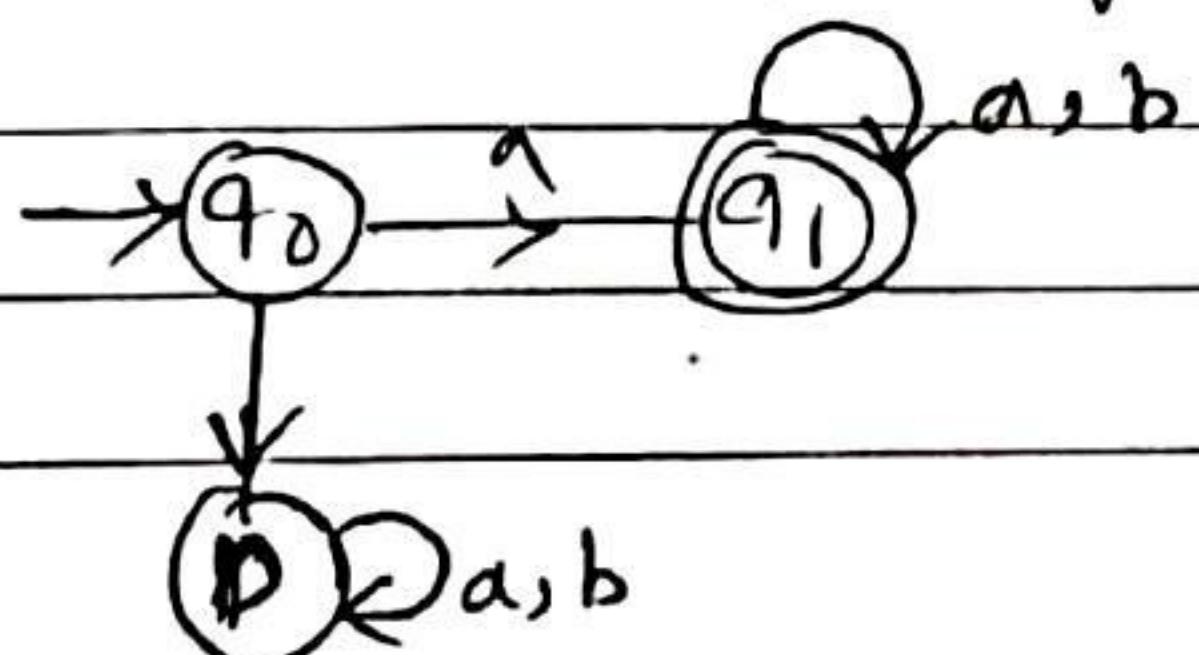
Construct a minimal DFA, which accepts set of all strings over $\{a, b\}$ where each string starts with any 'a'.

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

$$w \in \{a, b\}^*$$

language, $L_1 = \{a, aa, ab, aaa, \dots\}$

state transition Diagram,



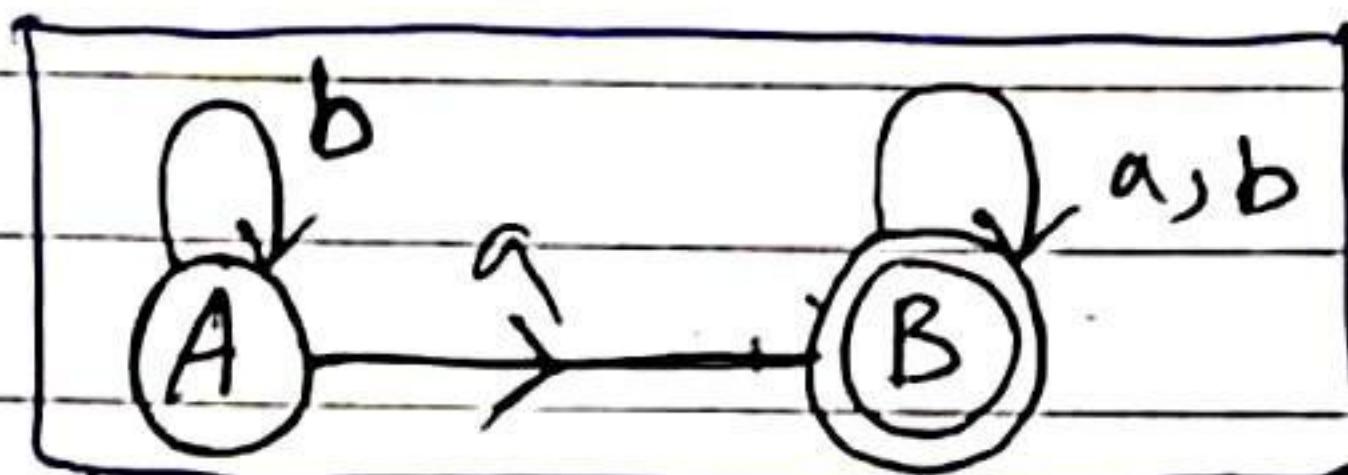
\rightarrow If the language is infinite
then take the smallest string and
make its skeleton.

Example-11

Construct a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$

"Whence each string contains 'a'."

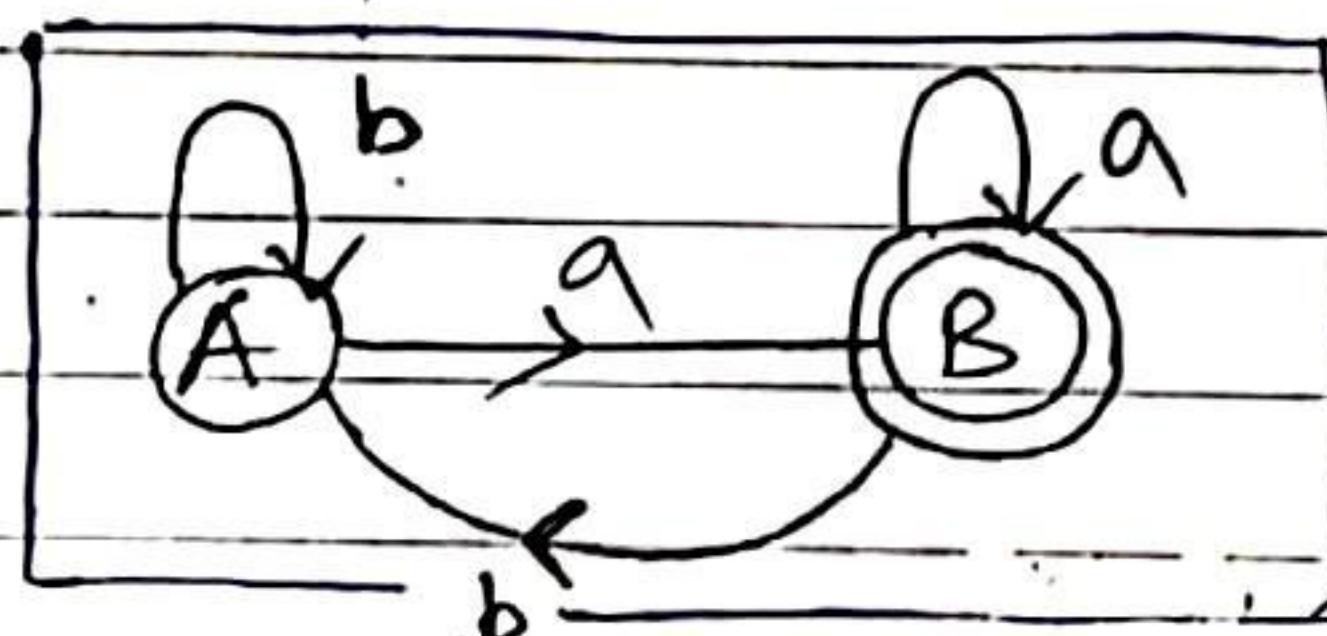
$$\rightarrow L = \{a, aa, ab, ba, aaa, \dots\}$$

**Example-12**

Construct a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$

String ends with an 'a'.

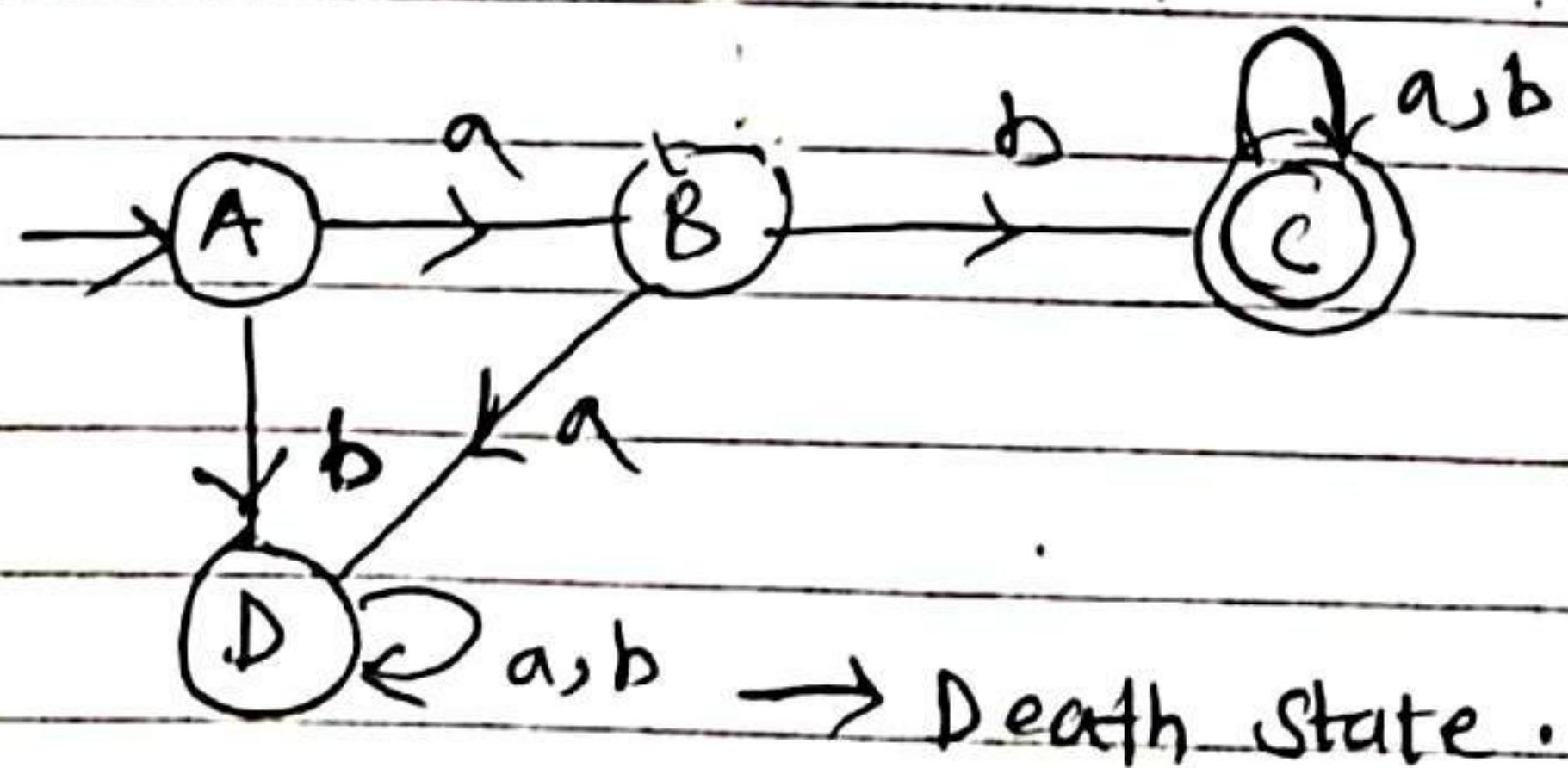
$$\rightarrow L = \{a, aa, ba, aaa, bba, baa, bbb, \dots\}$$

**Example-13**

Construct a minimal DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$

each string start with 'ab'.

$$\rightarrow L = \{ab, aab, abb, \dots\}$$

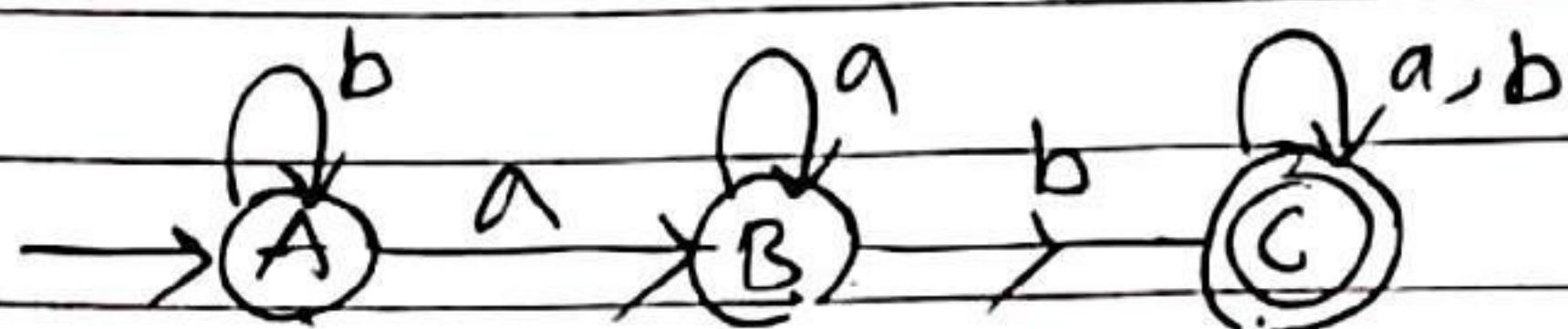


Ex-14

DFA, containing ab.

$$\rightarrow \Sigma = \{a, b\}, w \in \{a, b\}^*$$

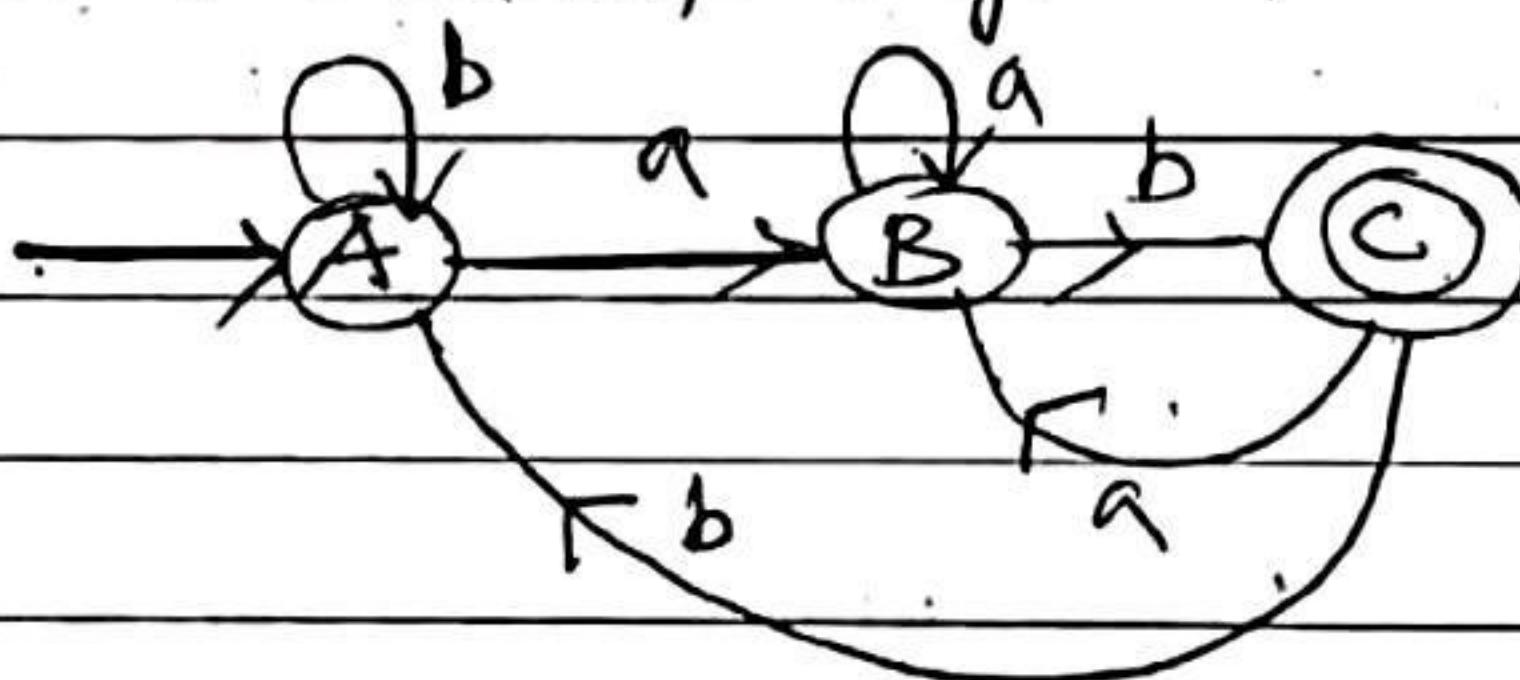
$$L = \{ab, abb, bab, \dots\}$$

**Ex-15**

construct a minimal DFA, ends with "ab".

$$L = \{ab, bbab, aabb, \dots\}$$

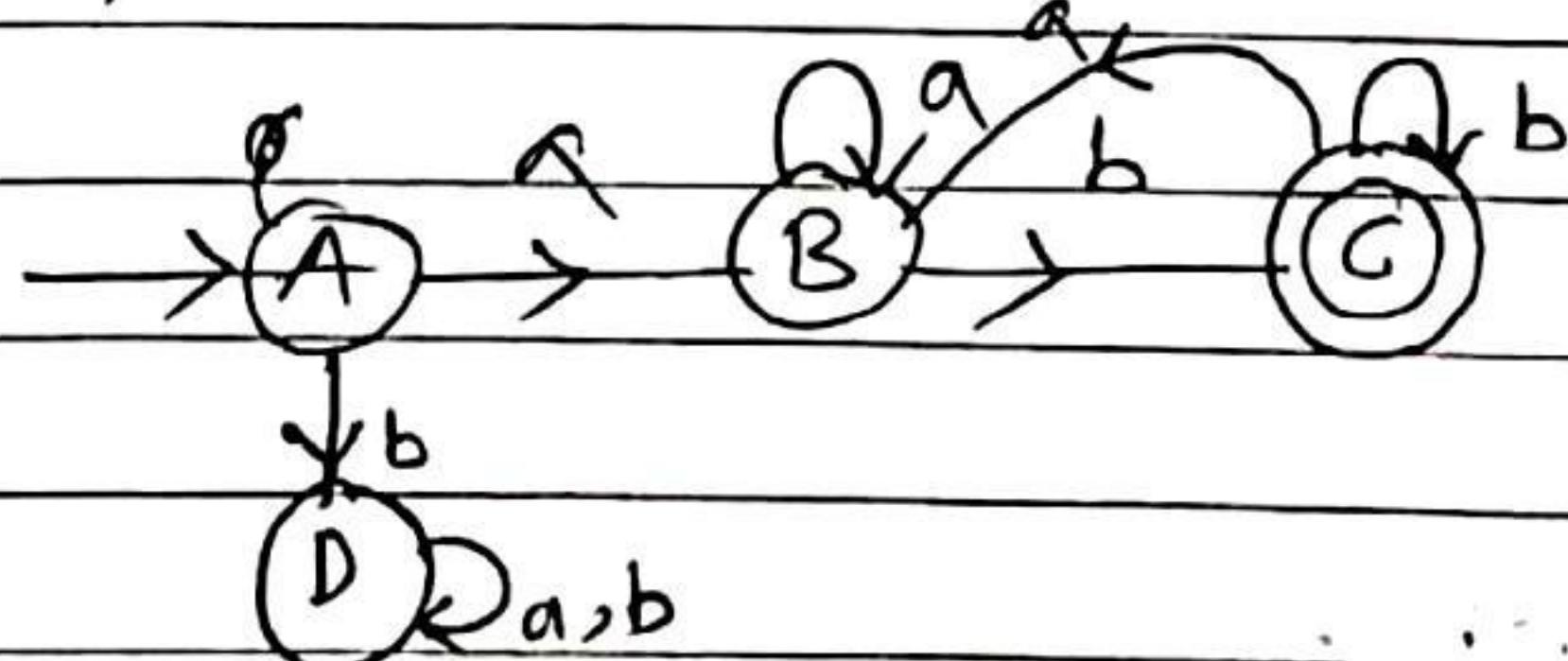
State transition diagram-

**Ex-16**

construct a minimal DFA, starts with 'a' and ends with 'b'.

$$\Sigma = \{a, b\}^*, w \in \{a, b\}^*$$

$$L = \{ab, aabb, abab, abbbab, \dots\}$$



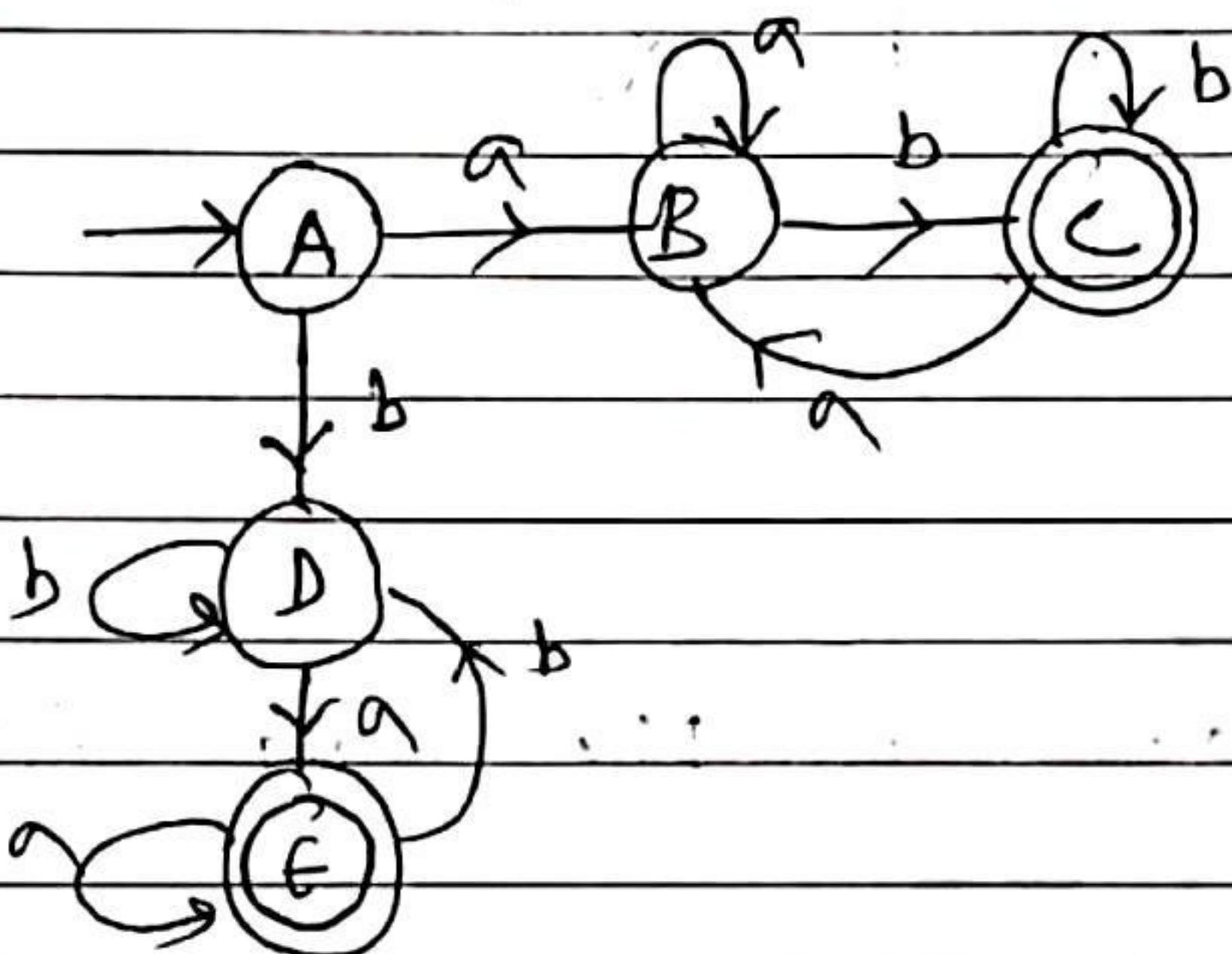
[Ex-17]

Construct a DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$

"start and ends with different symbol".

$\rightarrow \Sigma = \{a, b\}$, $w \in \{a, b\}^*$.

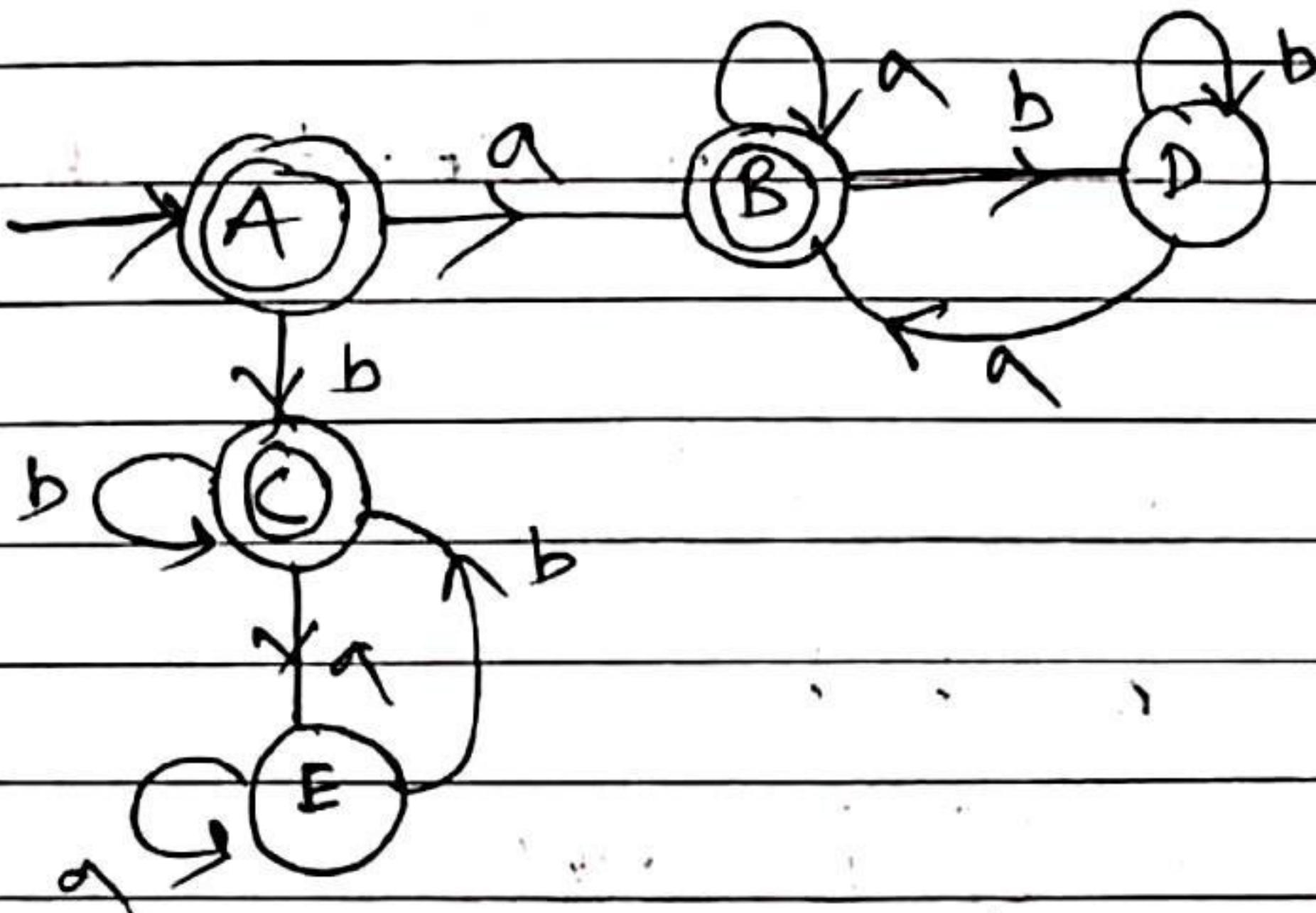
$$L = \{ab, ba, abb, bba, \dots\}$$

**[Ex-18]**

construct a DFA, $\Sigma = \{a, b\}$, $w \in \{a, b\}^*$.

"start and ends with same symbol".

$\rightarrow L = \{aa, \{e, a, b, aa, bb, \dots\}\}$



\rightarrow here, Example - 17 and 18 are complements of each other.

Ex-18

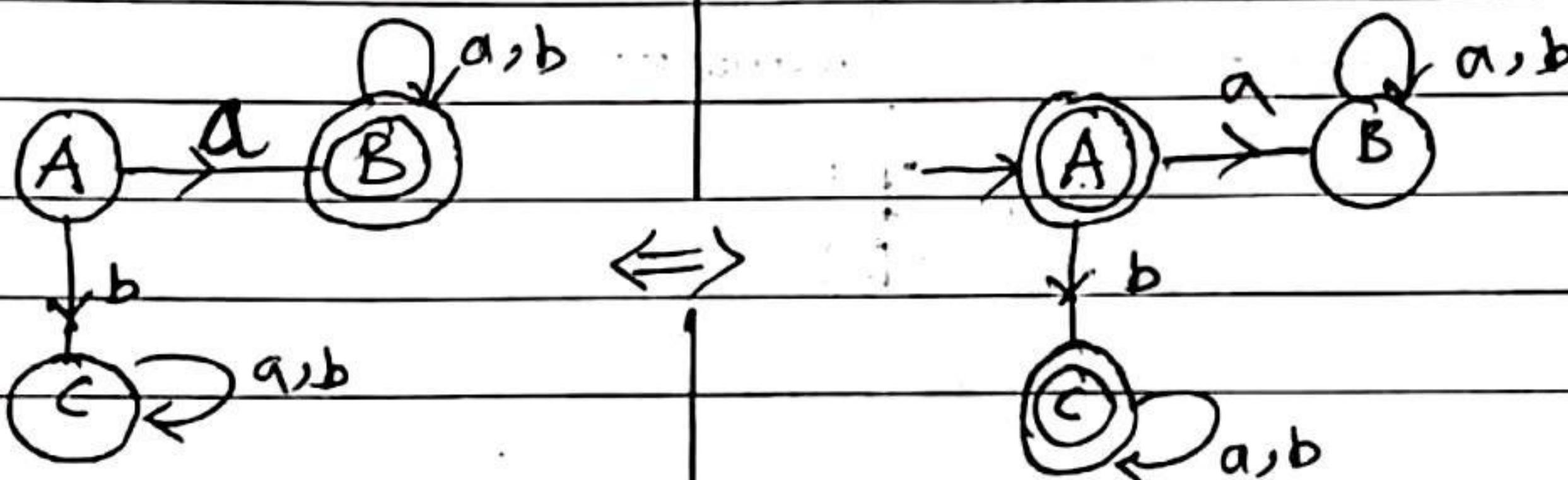
→ L_1 and L_2 are 2-language, then L_1 is said complement of L_2 ,

$$L_1 = \Sigma^* - L_2$$

• Complementation of DFA:

$L_1 = \{ \text{Starting with } 'a' \}$

$L_2 = \{ \text{Not starting with } 'a' \}$



$$L_1 = \overline{L_2}$$

→ Complementation method apply for only DFA not for NFA.
will be changed

→ Every thing has to be same only changing final state.

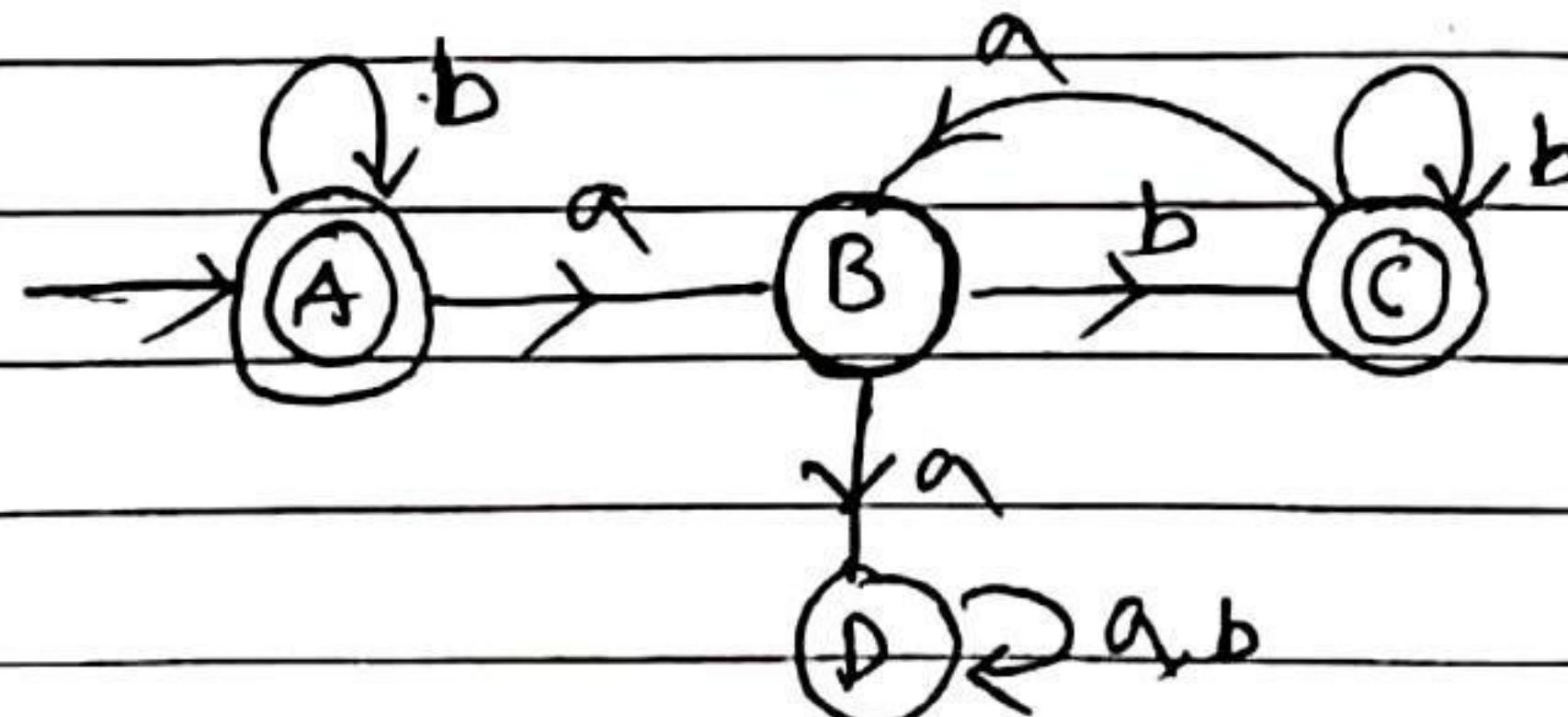
Ex-19

construct a DFA, $w \in \{a, b\}^*$, if

Every 'a' should be followed by a 'b'.

→ $L = \{ \epsilon, ab, abb, abab, babbb, \dots, b, bbb, \dots \}$

State transition Diagram -



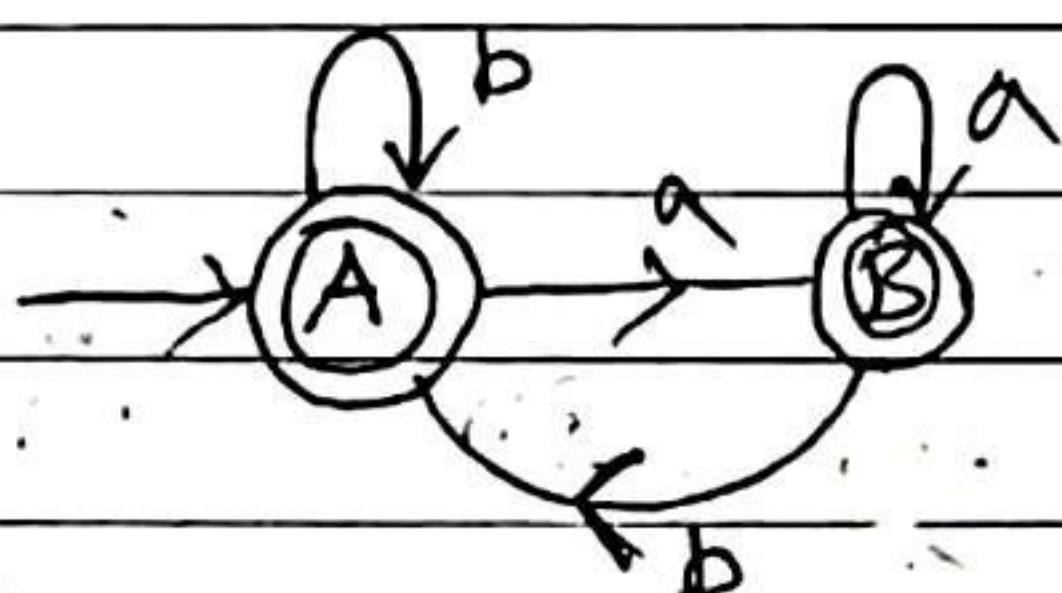
Ex-20

Construct a DFA, where, $w \in \{a, b\}^*$

Every 'a' should not never followed by a 'b'.

$$\rightarrow L = \{\epsilon, a, b, aa, ba, bb, aaa, bba, \dots\}$$

State transition diagram -



Ex-21

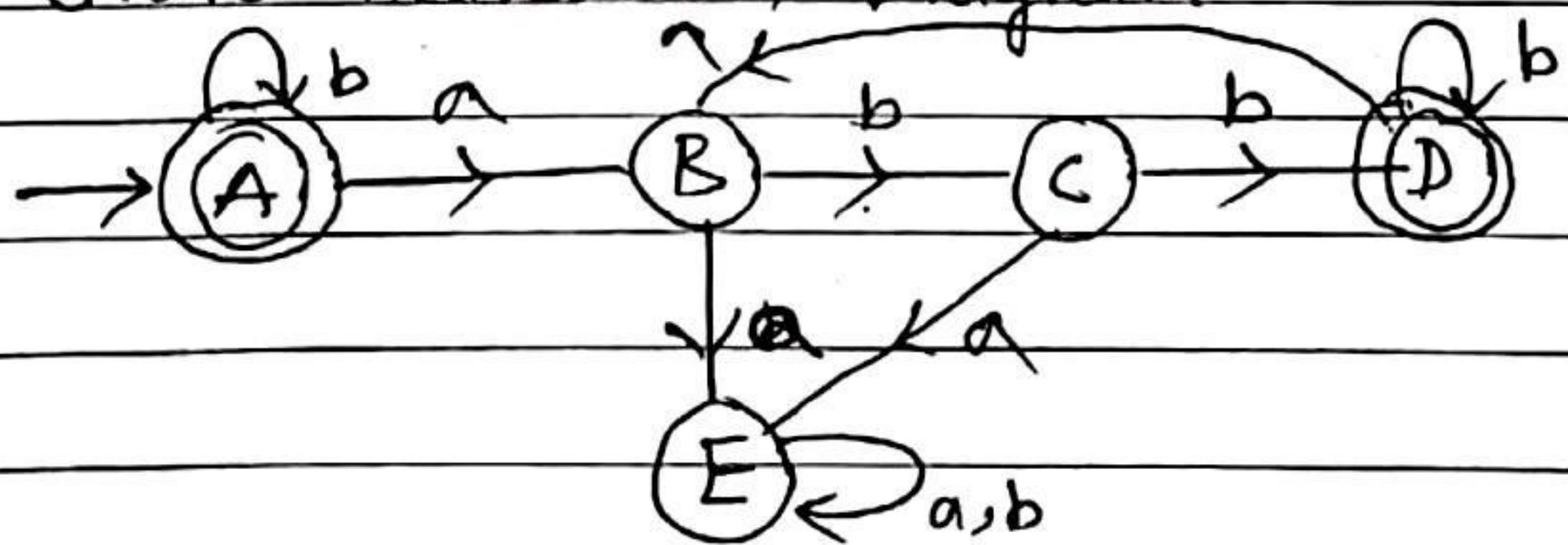
construct a minimal DFA, where

$$w \in \{a, b\}^*$$

every 'a' should be followed by 'bb'.

$$\rightarrow L = \{\epsilon, abb, \dots\}$$

State transition diagram -



$$\text{final state} = \{A, D\}$$

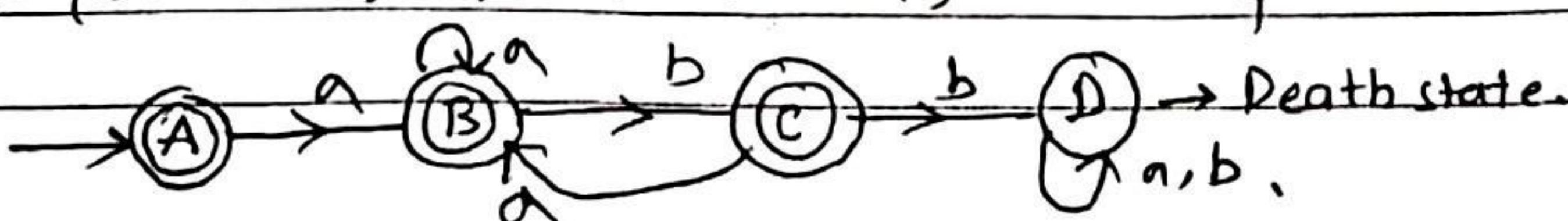
Ex-22

Construct a DFA, where

$$w \in \{a, b\}^*$$

every 'a' should never be followed by a 'bb'.

$$\rightarrow L = \{\epsilon, a, aa, ab, ba, bb, aaa, \dots\}$$

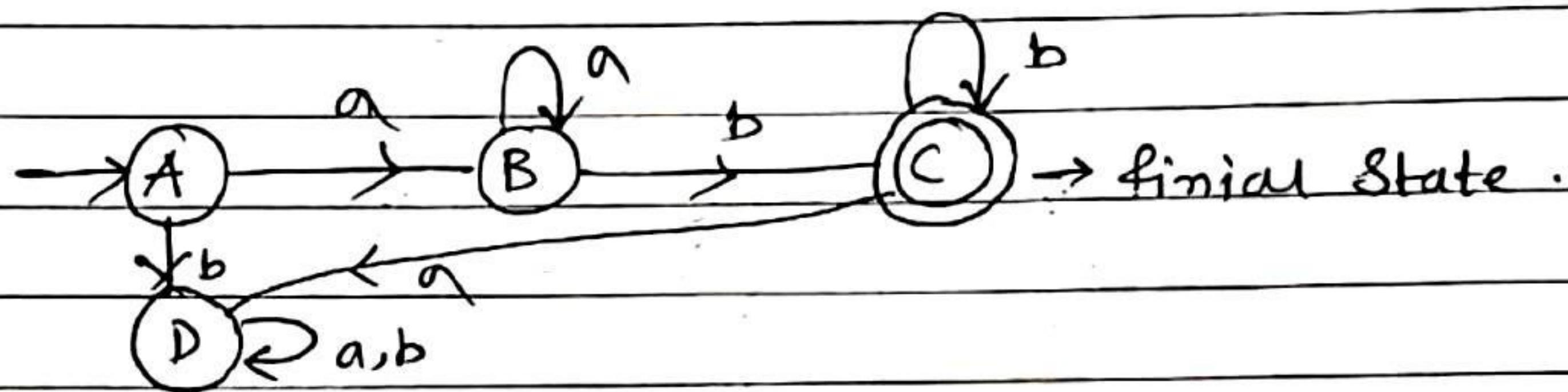


Ex-23

Construct a minimal DFA which accepts, $L = \{a^n b^m / n, m \geq 1\}$

$$\rightarrow L = \{ab, aab, aabb, aabb\dots\}$$

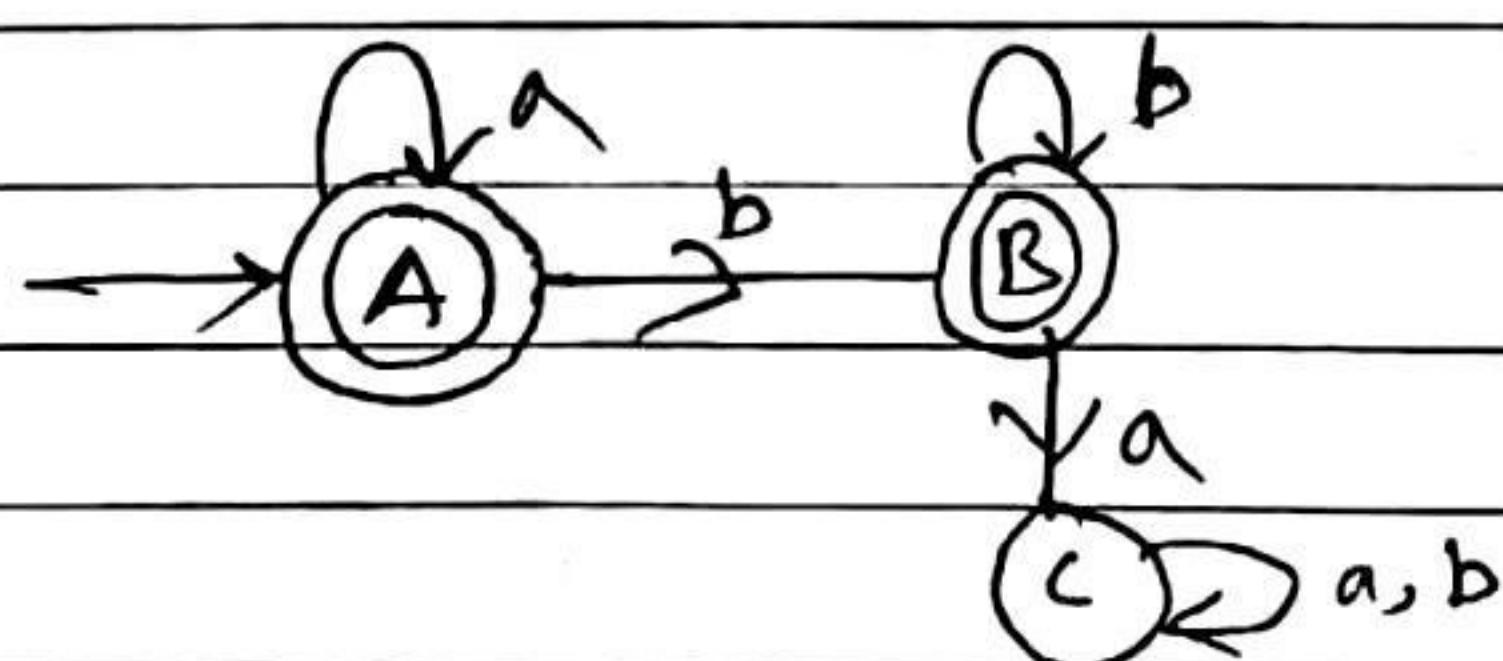
State transition diagram-

**Ex-24**

Construct a minimal DFA, which accepts $L = \{a^n, b^m / n, m \geq 0\}$

$$\rightarrow L = \{\epsilon, a, aa, aaa, \dots, b, bb, bbb\dots, ab, aabb\dots\}$$

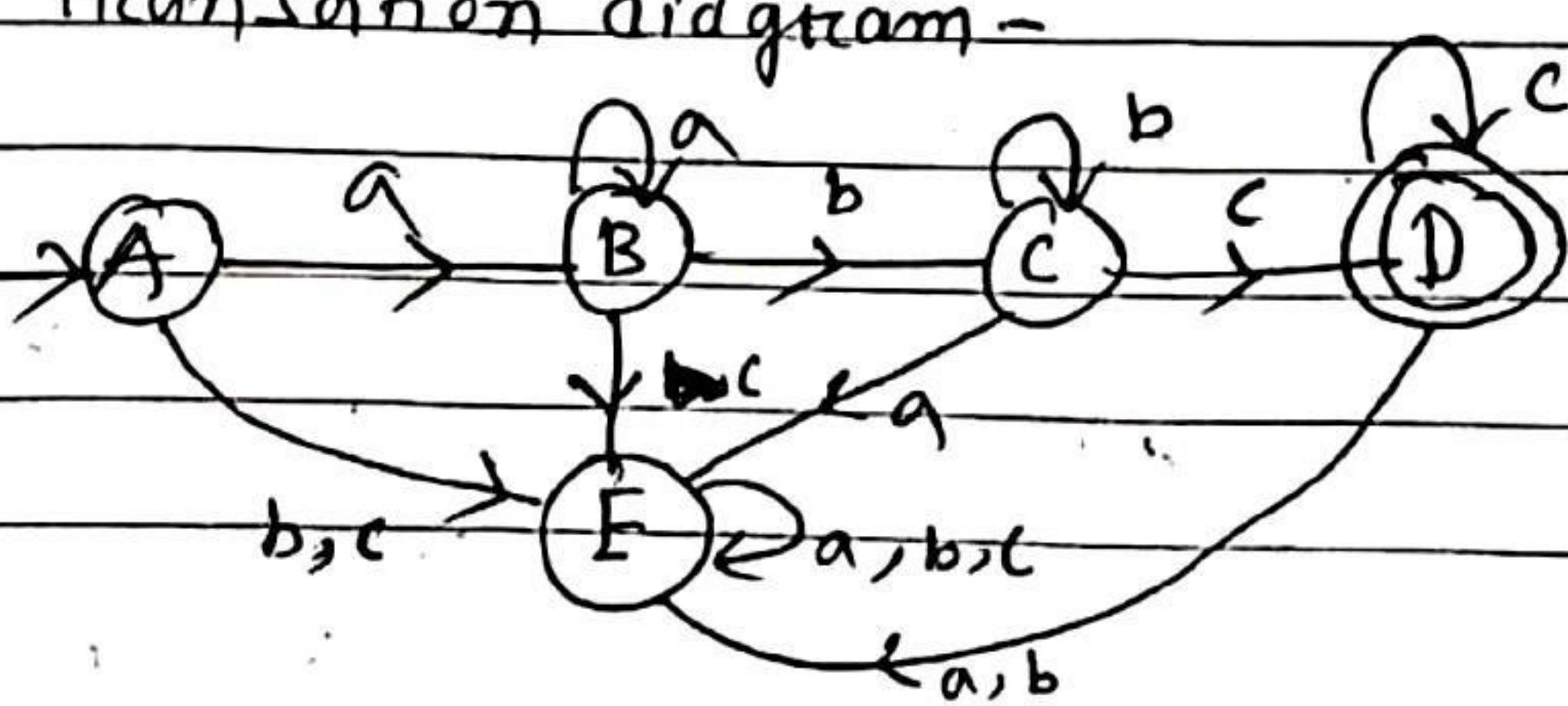
State transition Diagram-

**Ex-25**

Constructs a minimal DFA, which accepts $L = \{a^n b^m c^l / n, m, l \geq 0\}$

$$\rightarrow L = \{abc, aabbcc\dots\}$$

State transition diagram-



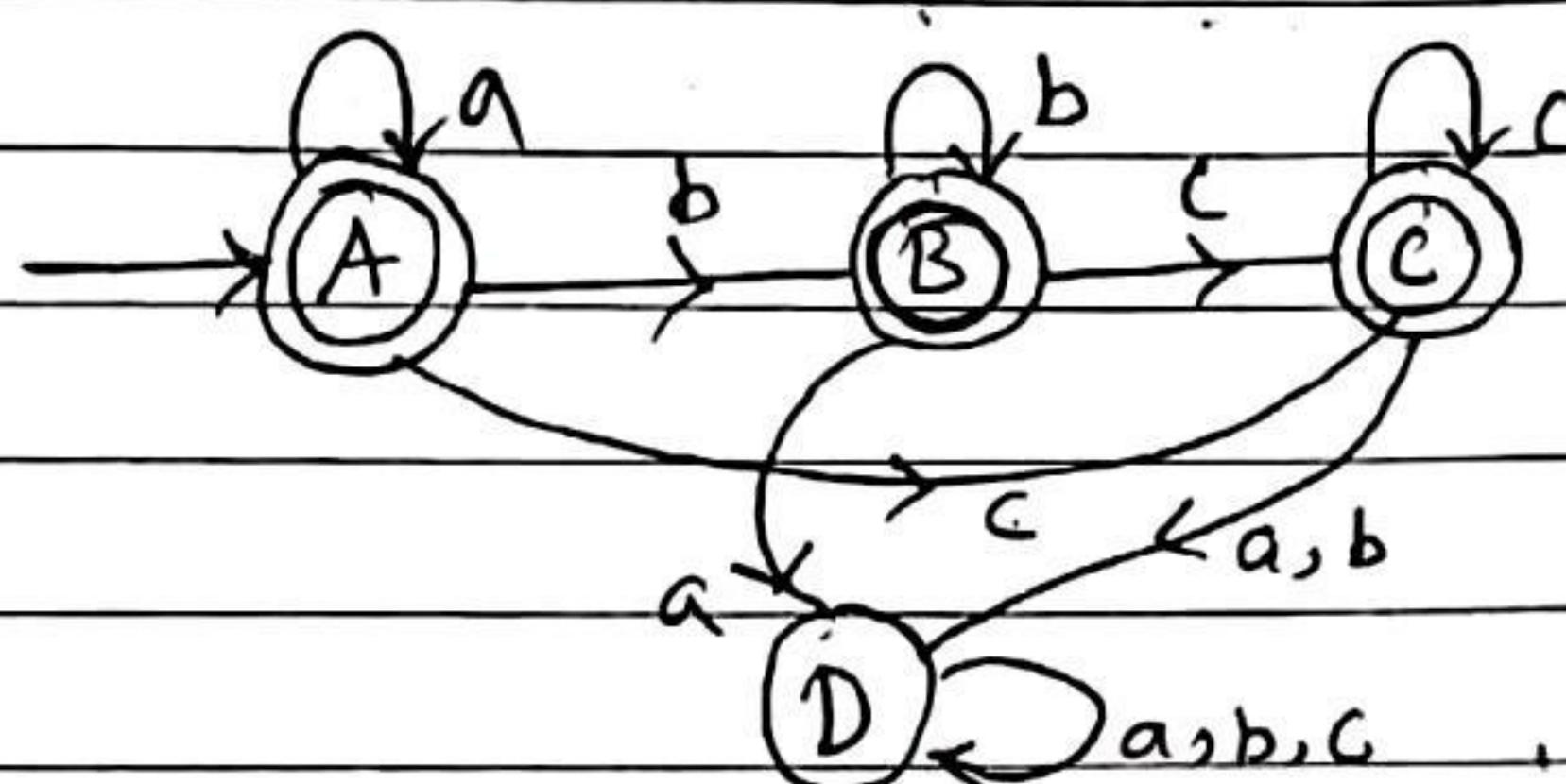
Ex-26

Construct a minimal DFA, which accepts,

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

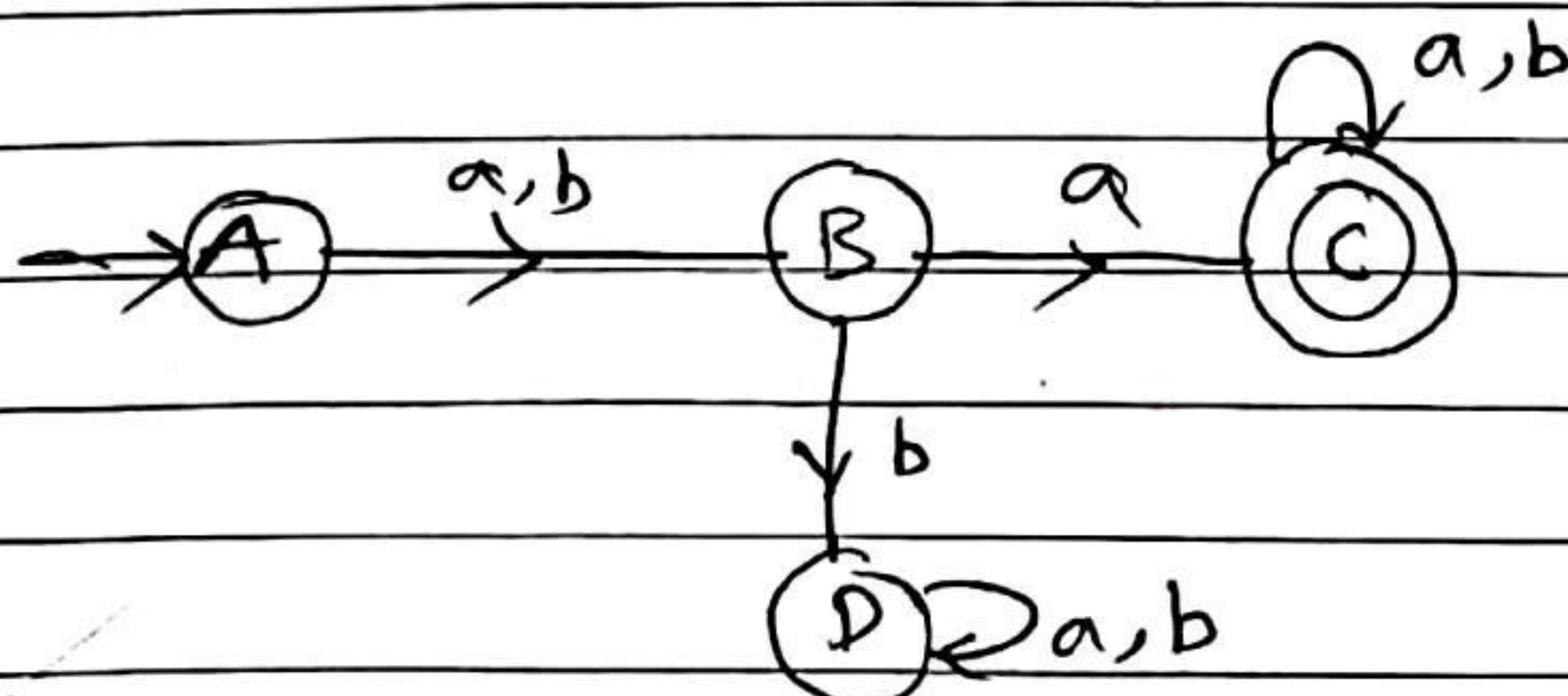
$$\rightarrow L = \{\epsilon, a, aa, \dots, b, bb, \dots, c, cc, \dots, abc, aabbcc\}$$

State transition Diagram -

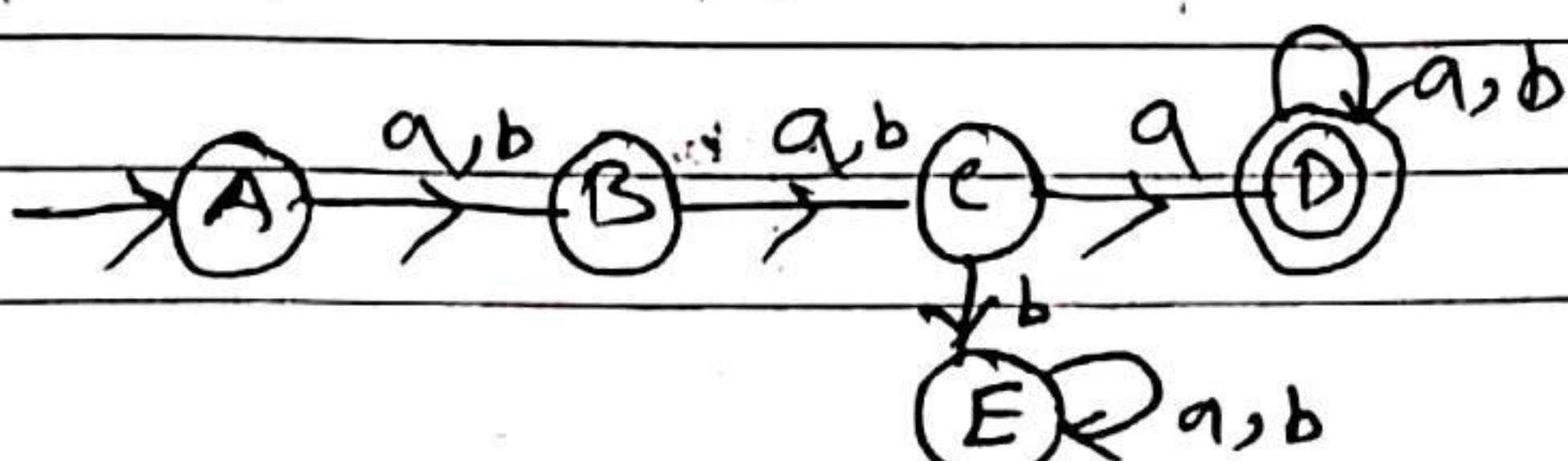
**Ex-27**Construct a minimal DFA which accepts set of all strings over $\{a, b\}$ such that second symbol from L.H.S is 'a'.

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

$$L = \{aa, ba, aaa, baa, bba, \dots\}$$

**Ex-28** 3rd symbol from L.H.S is 'a'.

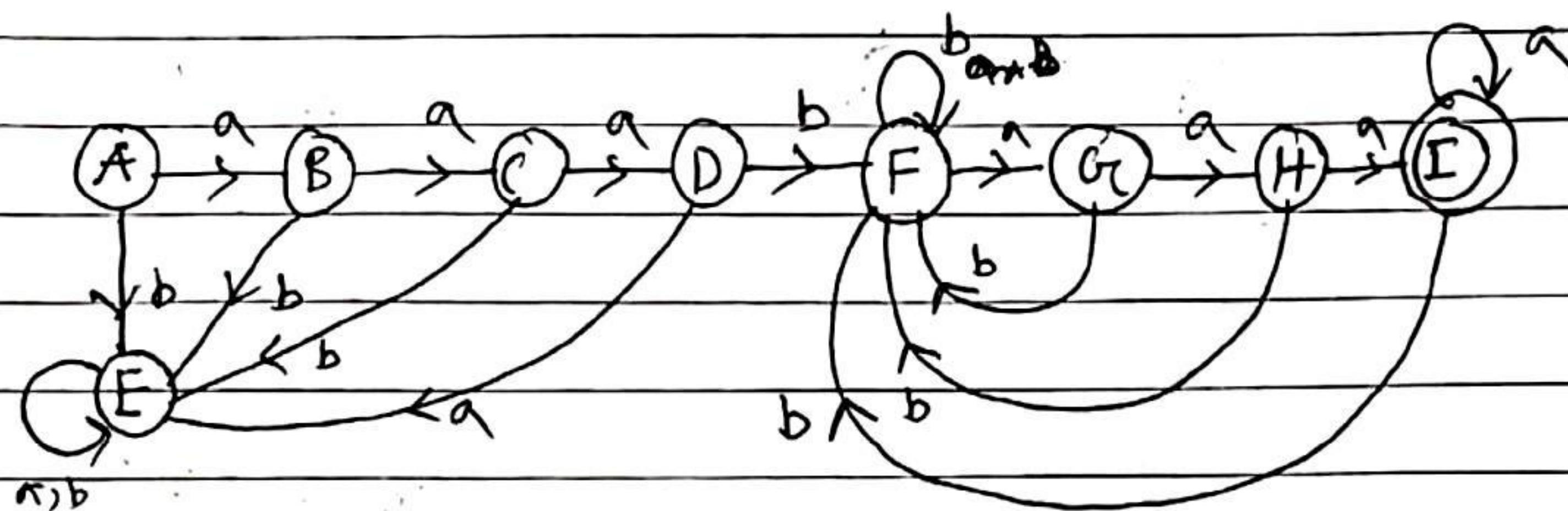
$$\rightarrow L = \{aaa, aba, baa, bba, aaa, abaa, \dots\}$$



Ex-29 construct a DFA which accepts set of all string over $\{a, b\}$ where strings are of the form ' a^3bwa^3 '.

where ' w ' is any string over $\{a, b\}$.

$$\rightarrow L = \{a^3b \in a^3, a^3baa^3, a^3bba^3, \dots\}$$



• OPERATION ON DFA like -

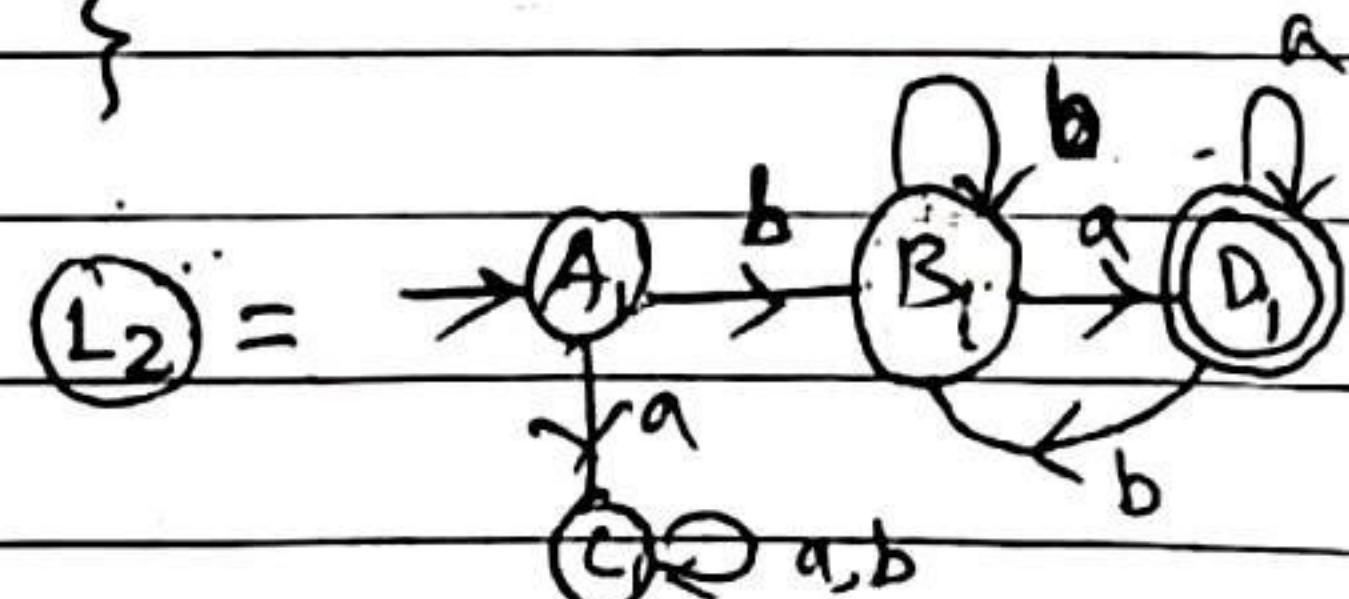
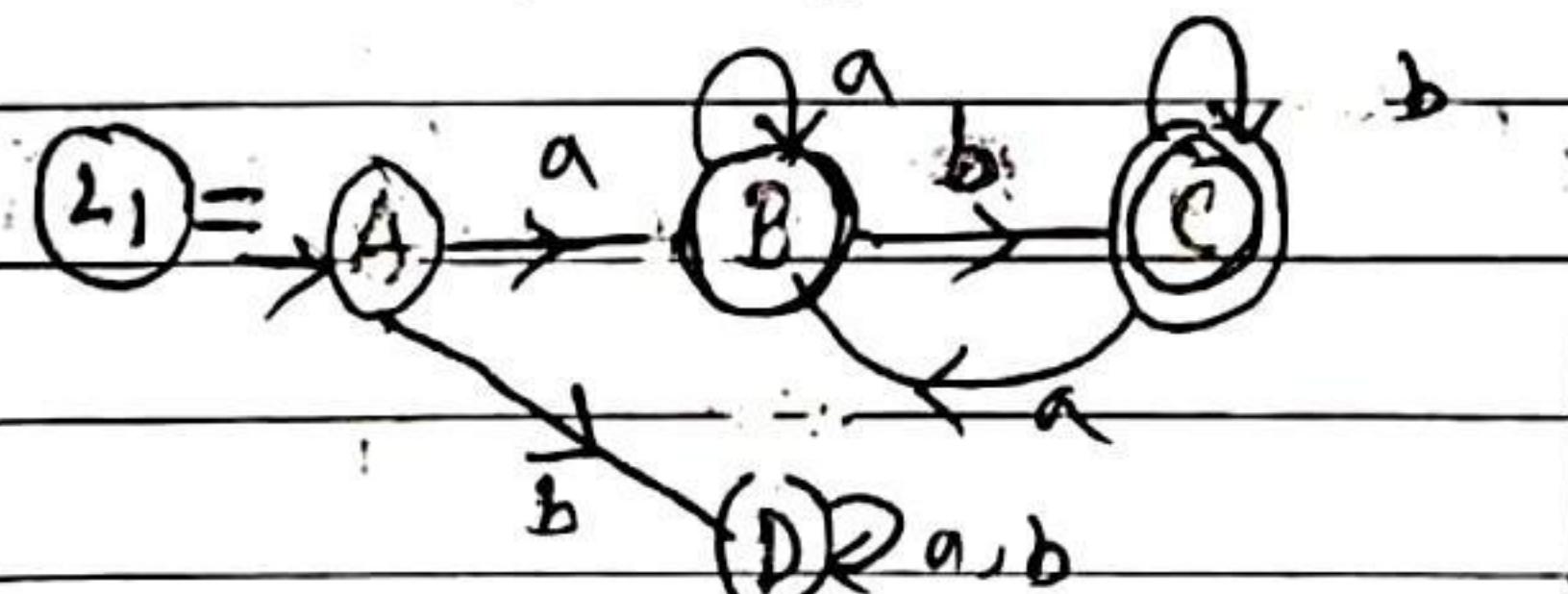
- (i) Union (ii) Concatenation (iii) cross product (iv) complementation.
- (v) reversal.

① Union :-

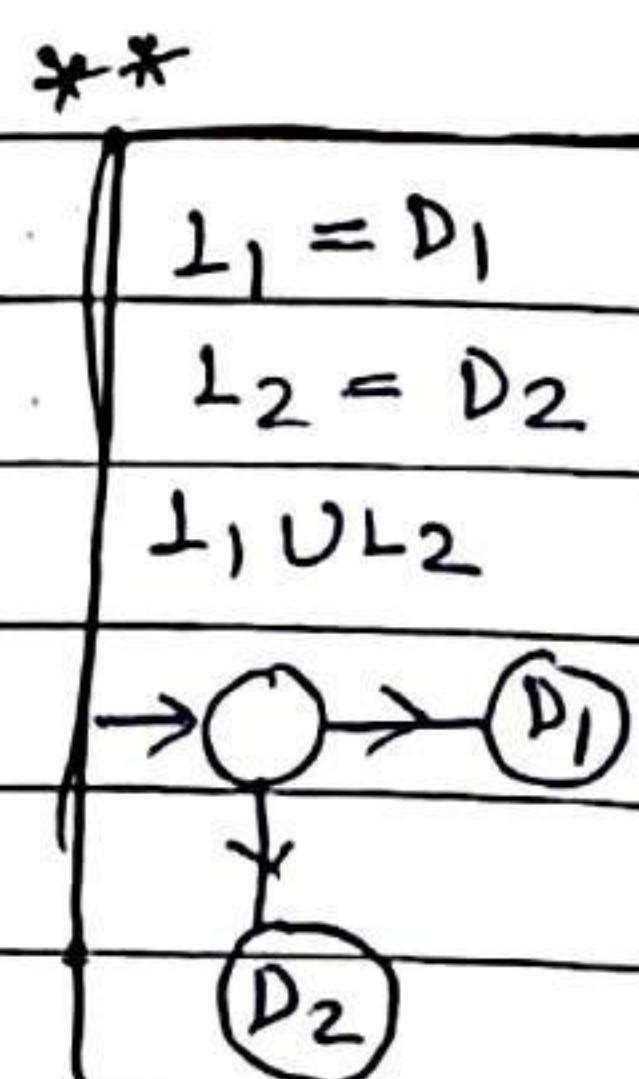
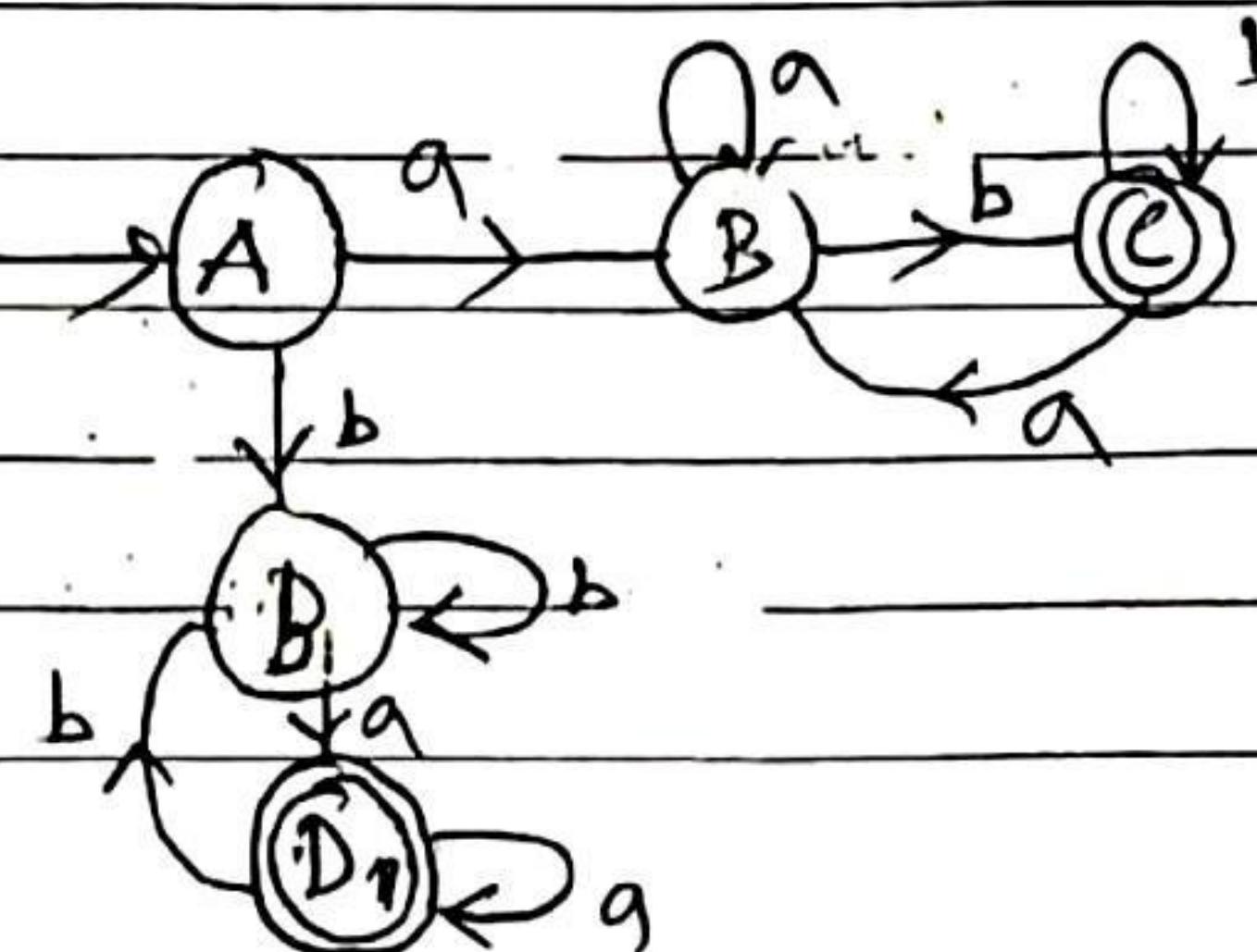
Ex - starts and ends with different symbol.

$$\rightarrow L_1 = \{ab, aab, aba, abb, \dots\}$$

$$L_2 = \{ba, baa, bba, \dots\}$$



$$[L_1 \cup L_2] =$$



② Concatenation — $L_1 = D_1, L_2 = D_2 / L_1 \cdot L_2 = D_1 \cdot D_2$.

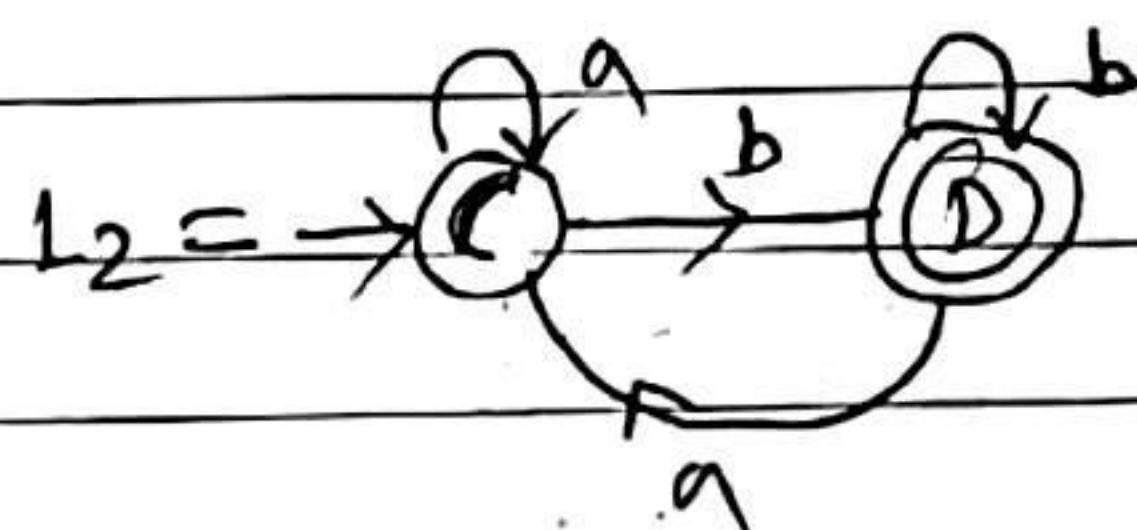
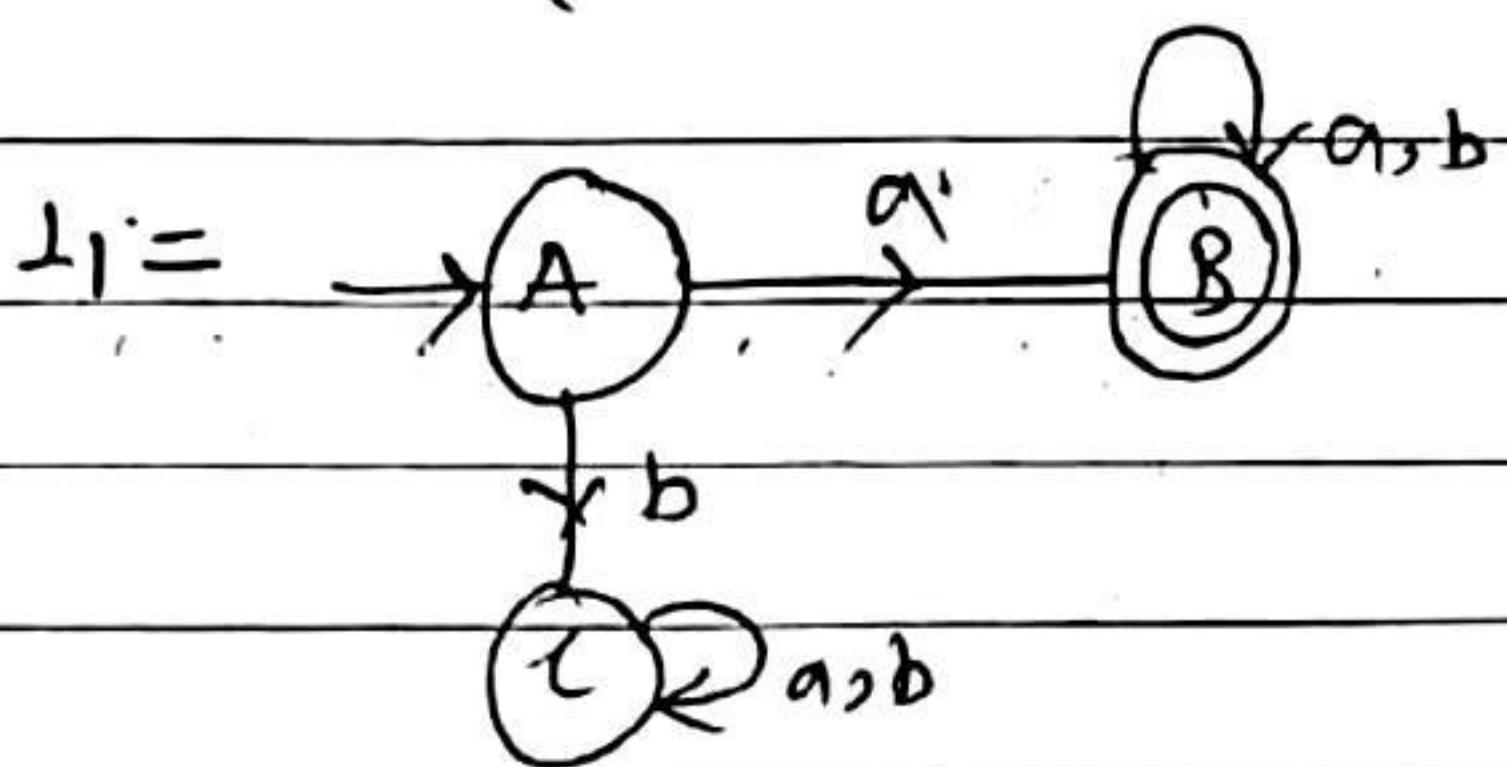
→ starting with 'a' and ending with 'b'.

$$L_1 = \{a, aa, ab, aua \dots\}$$

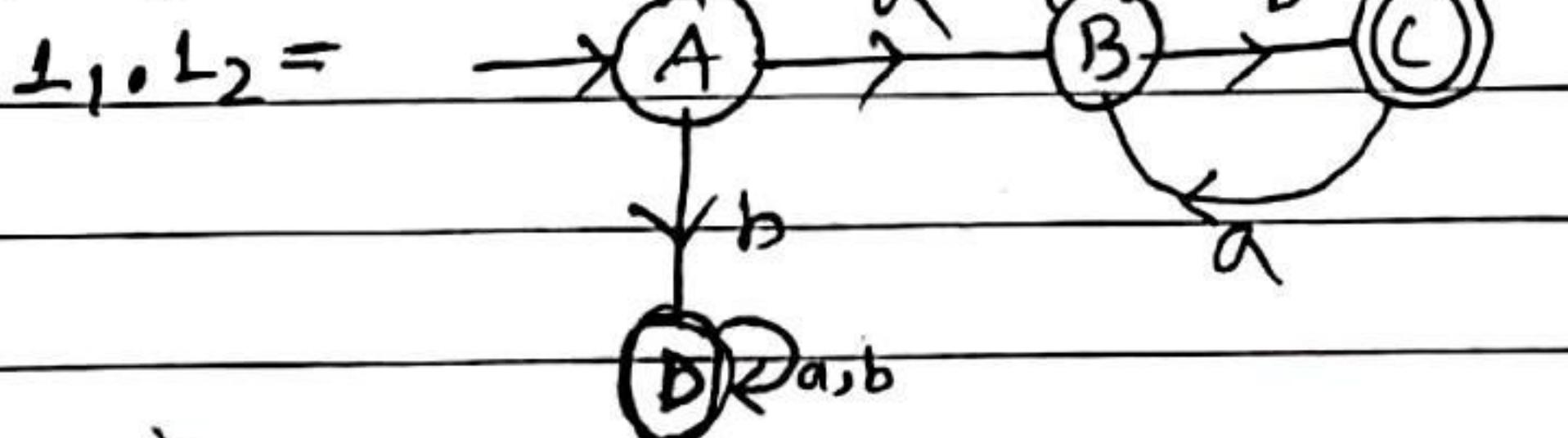
(starting with a)

$$L_2 = \{b, ab, bb, aab, bab, bbb, \dots\}$$

(ending with b)



Concatenation,



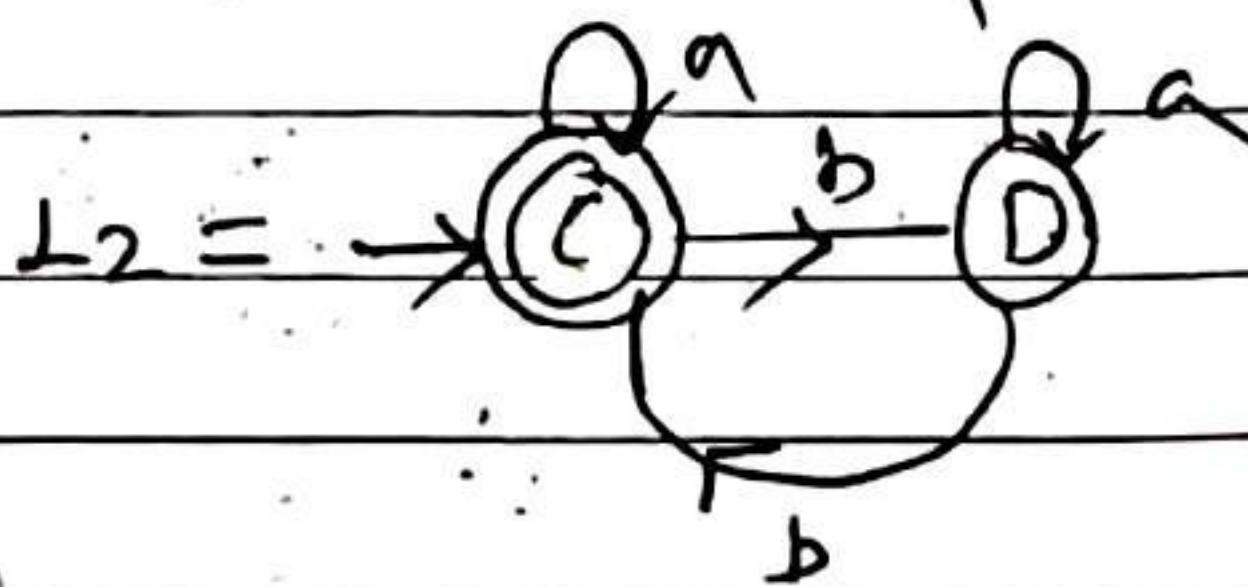
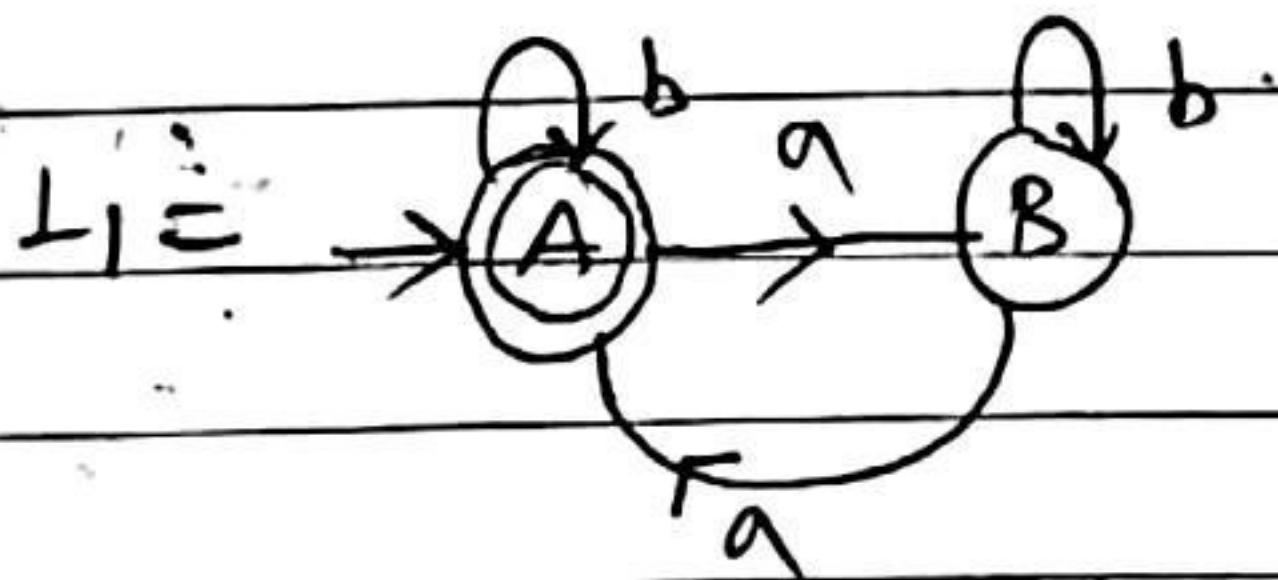
③ Cross product method —

→ even no of 'a's and even num of b's.

H

$$L_1 = \{aa, baa, aab, aaaa, aaaaaa \dots\}$$

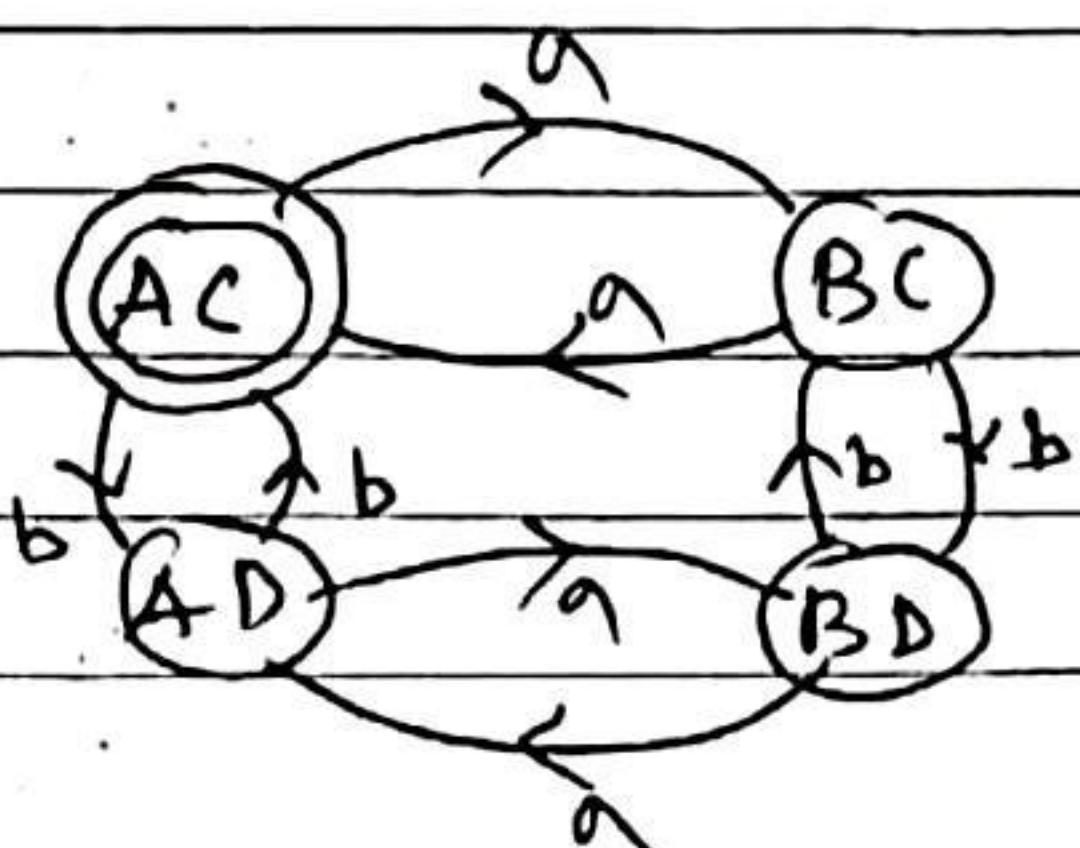
$$L_2 = \{b, bb, bba, bbb, bbb.bbb \dots\}$$



$L_1 \times L_2$

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

$$= \{AC, AD, BC, BD\}$$



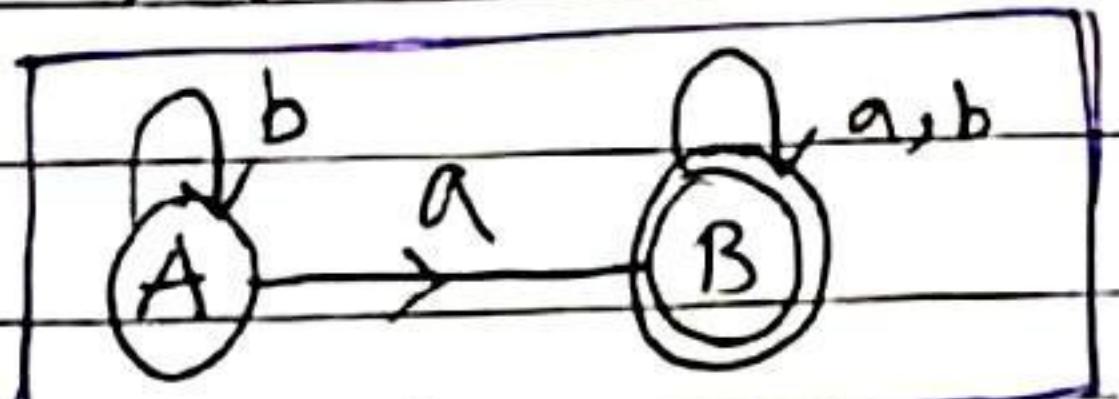
(4) Complement :-

\rightarrow Does not contain 'a'.

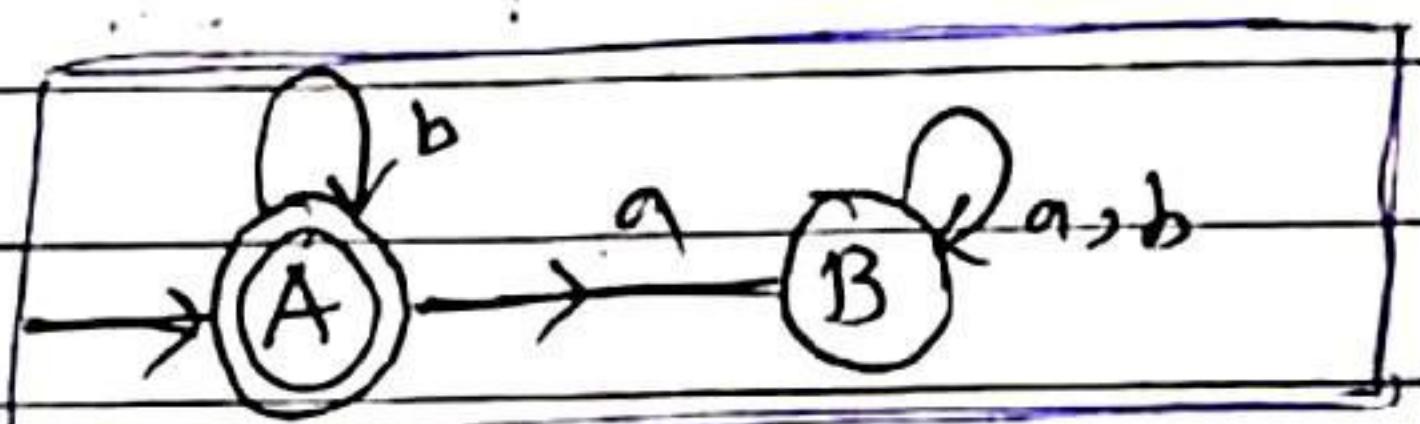
$$L_1 = \{ \text{containing 'a'} \}$$

$$= \{a, aa, ab, ba, aaa \dots\}$$

$$\bar{L}_1 = \{b, bb, bbb, bbbb \dots\}$$



↓ complement

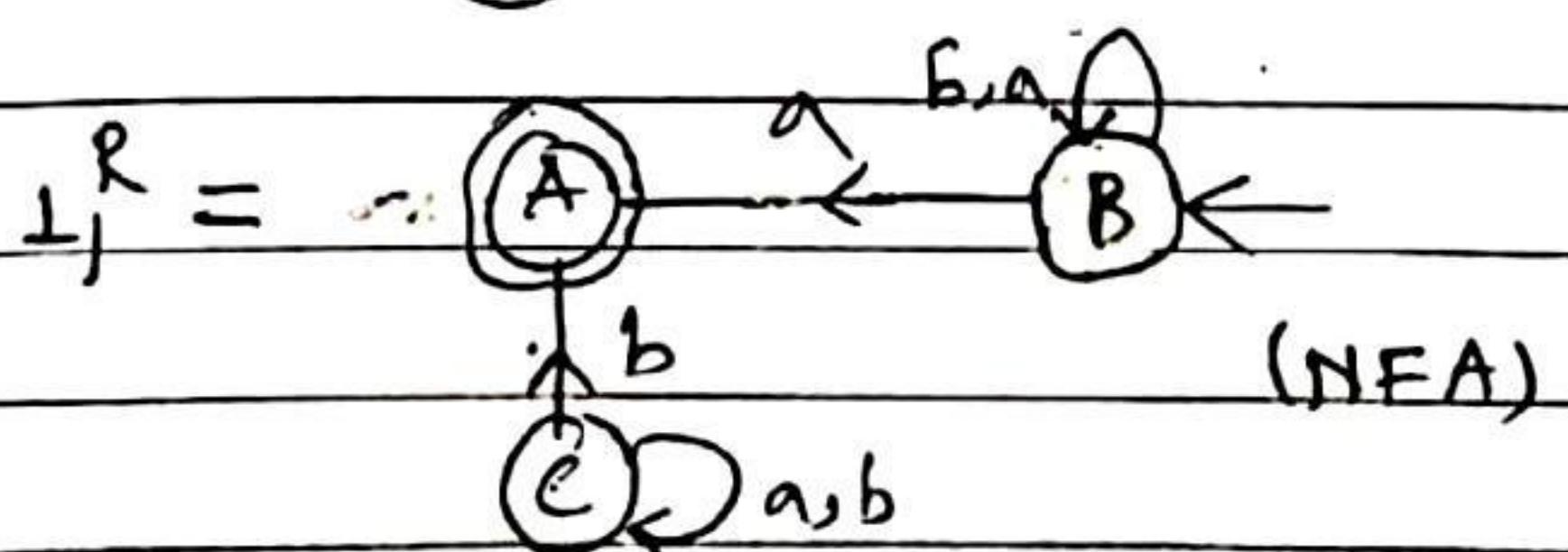
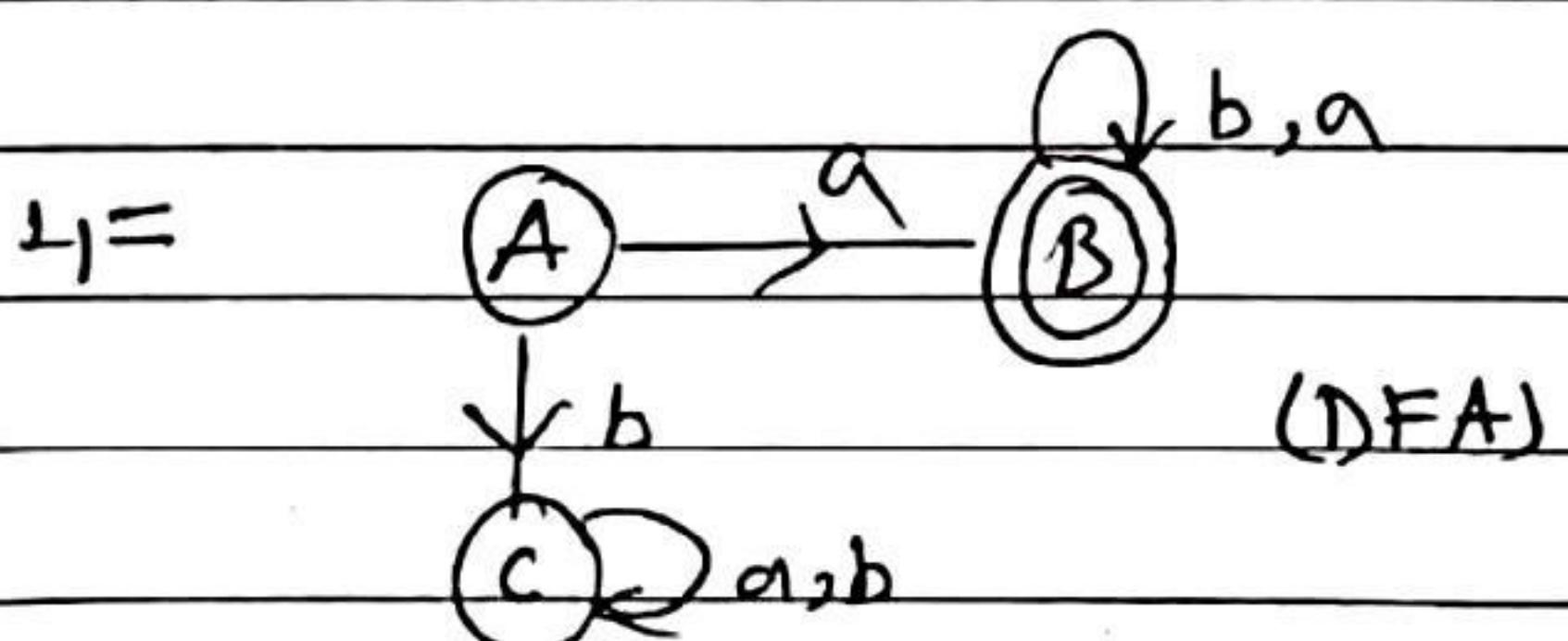
(5) Reversal :-

$\rightarrow awb \Leftrightarrow bwa$.

$$L_1 = \{ \text{start with } a \}$$

$$= \{a, aa, ab, aaa, aba, aaaa \dots\}$$

$$L_1^R = \{ a, aa, ba, aaa, aba, aaaa \dots\}$$



**

$$L_1 \xrightarrow{\exists} (DFA)^R \xrightarrow{\exists} L_1^R \xrightarrow{\exists} NFA/DFA$$

(30) Ex-30

Construct a minimal DFA over $\{a\}$.

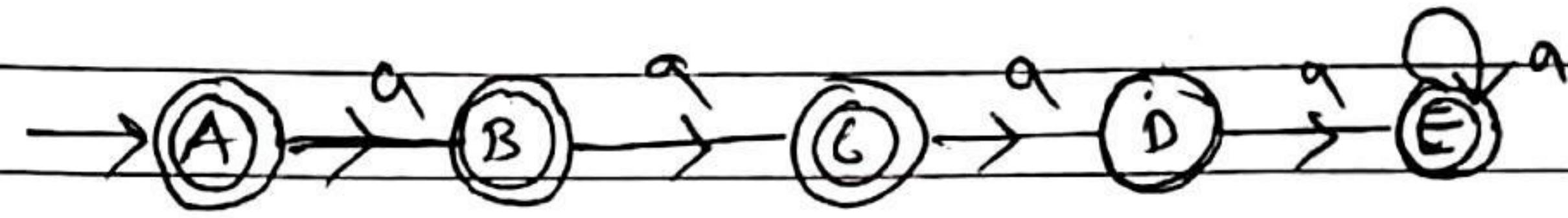
1. For $\{a^n / n \geq 0, n! = 3\}$

2. For $\{a^n / n \geq 0, n! = 2, n! = 4\}$.

$$\rightarrow \Sigma = \{a\}$$

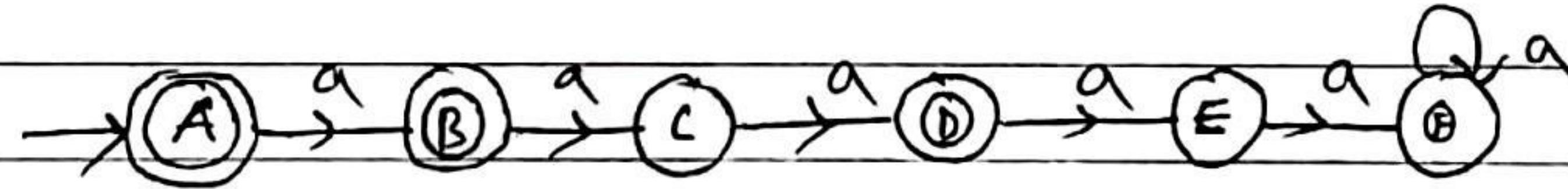
① $L = \{a^n / n \geq 0, n! = 3\}$

$$L = \{\epsilon, a, aa, aaaa, \dots\}$$



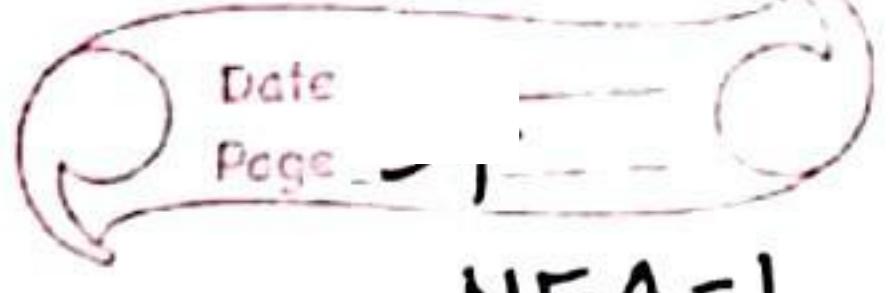
② For $L = \{a^n / n \geq 0, n! = 2, n! = 4\}$

$$= \{\epsilon, a, aaa, aaaaa, \dots\}$$



→ In NFA, not need to show dead state.

classmate



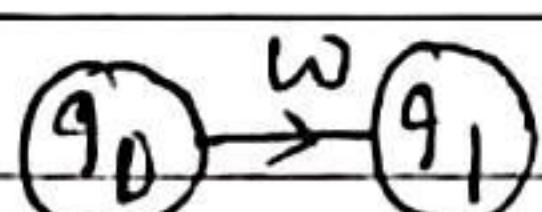
NFA-1

- NFA (Non-Deterministic Finite Automata):

→ Every DFA is NFA.

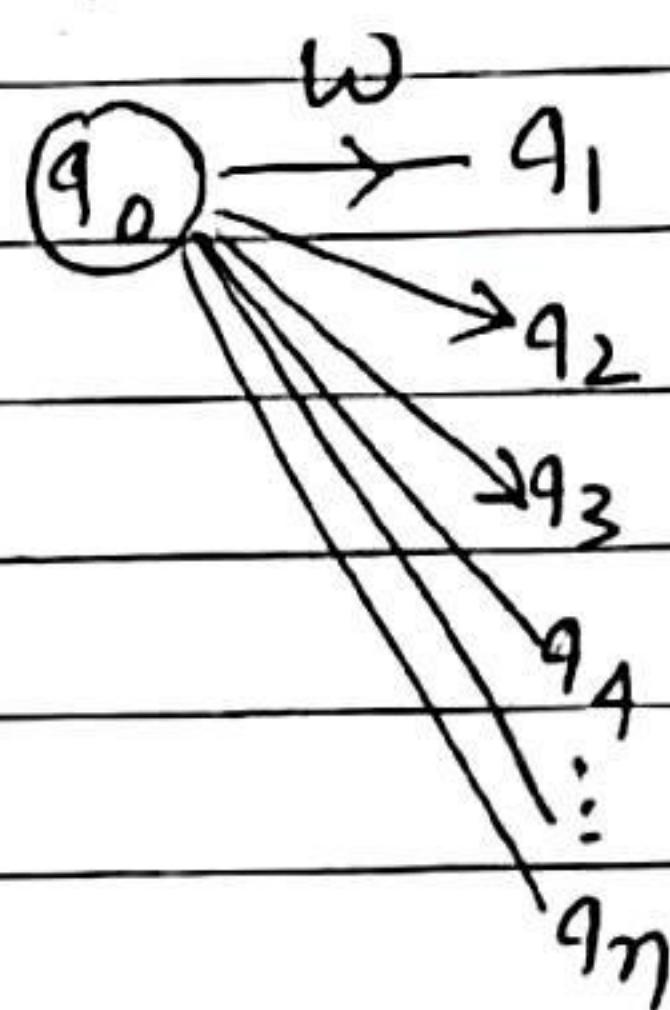
$$S: Q \times \Sigma \rightarrow Q$$

DFA



$$S: Q \times \Sigma^* \rightarrow 2^Q$$

NFA



Ex-1 Construct an NFA, which accepts set of all strings over $\{a, b\}$ such that $L = \{ \text{ends with } 'a' \}$

$$\rightarrow L = \{ a, aa, ba, aaa, baa, \dots \}$$

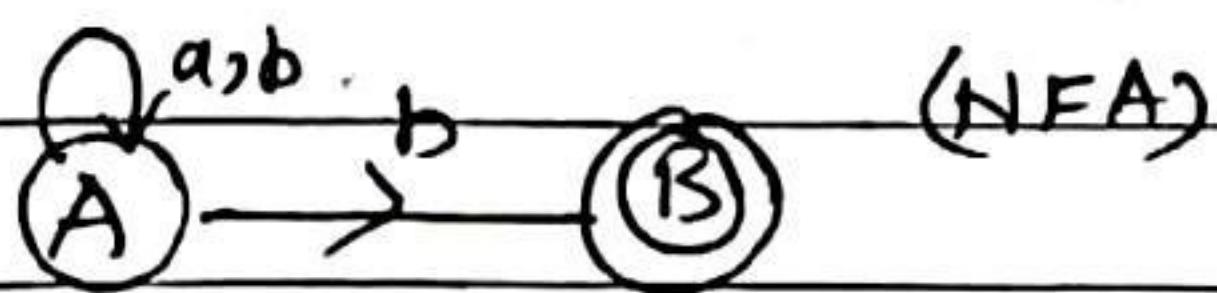
State transition Diagram,



→ [The string 'a' is accepted by NFA, if start with initial state and end if reach at least 1 state is final]

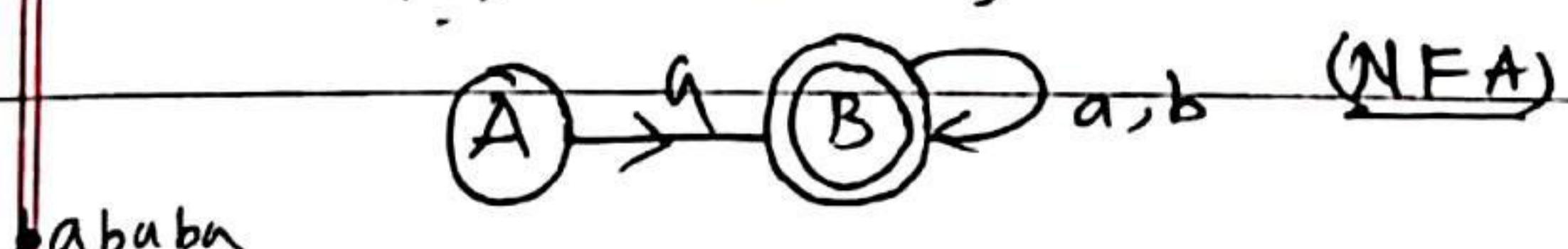
Ex-2 $L = \{ \text{ending with } b \}$

$$\rightarrow L = \{ b, ab, abb, \dots \}$$

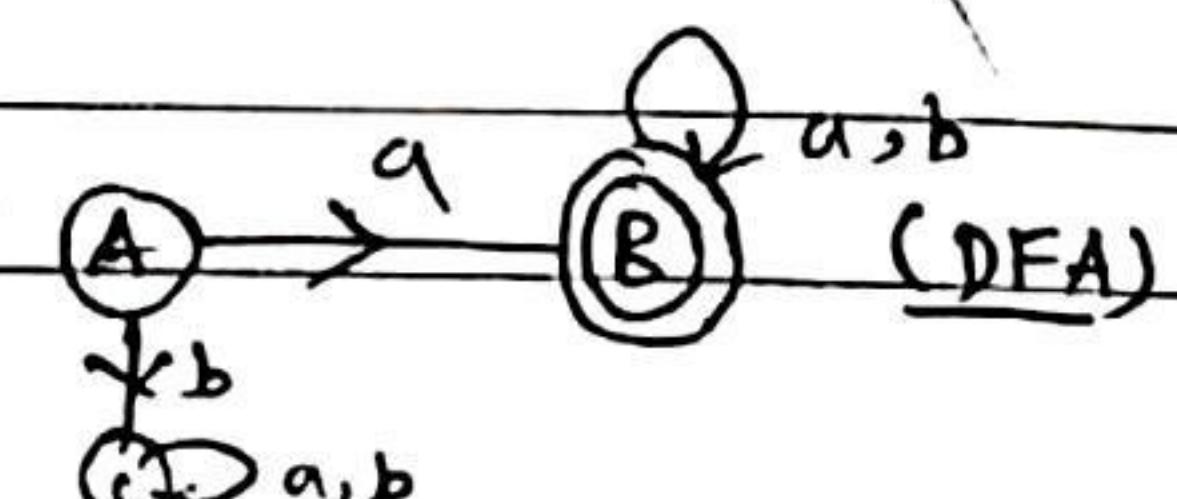


Ex-3 $L = \{ \text{starting string starts with } 'a' \}$

$$\rightarrow L = \{ a, aa, ab, \dots \}$$



bababa



Ex-3

$L = \{ \text{set of all strings containing 'a'} \}$

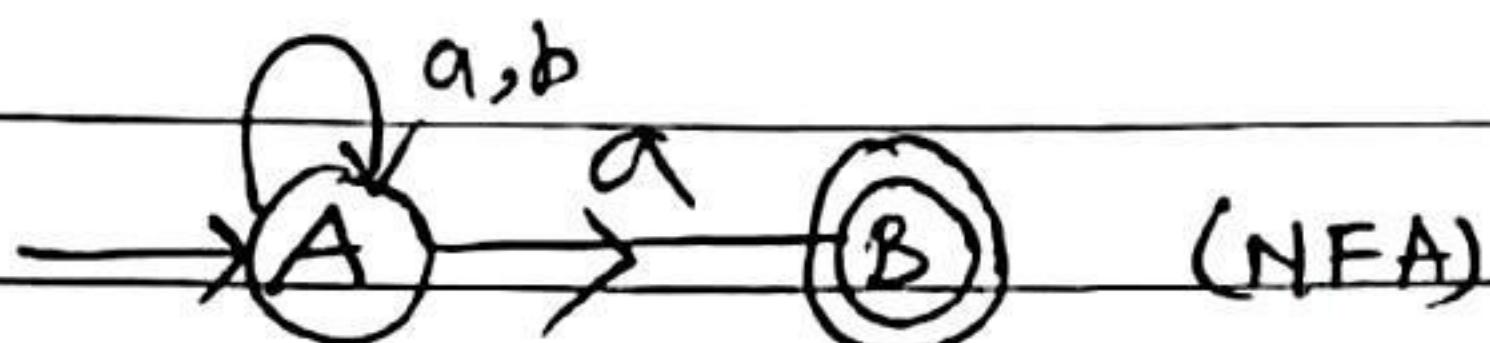
$\rightarrow L = \{ a, aa, ab, ba, aaa, aab, \dots \}$



Ex-4

$L = \{ \text{set of all strings ends with 'a'} \}$

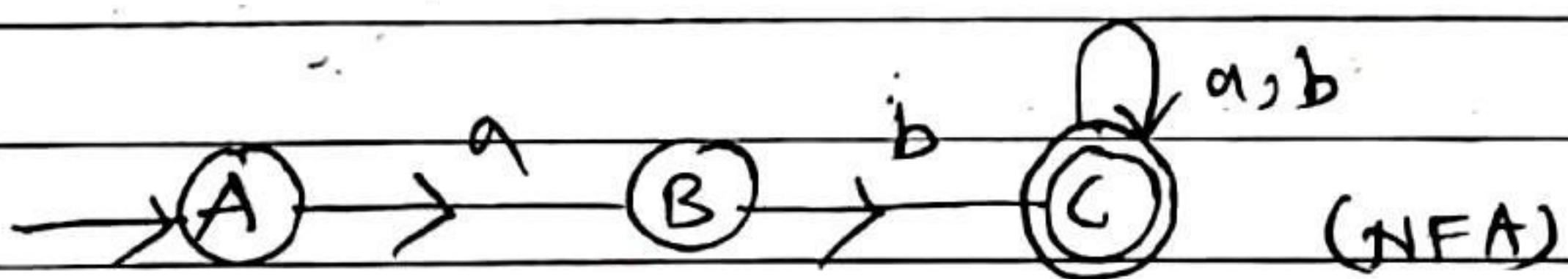
$\rightarrow L = \{ a, aa, ba, \dots \}$



Ex-5

$L = \{ \text{all string start with 'ab'} \}$

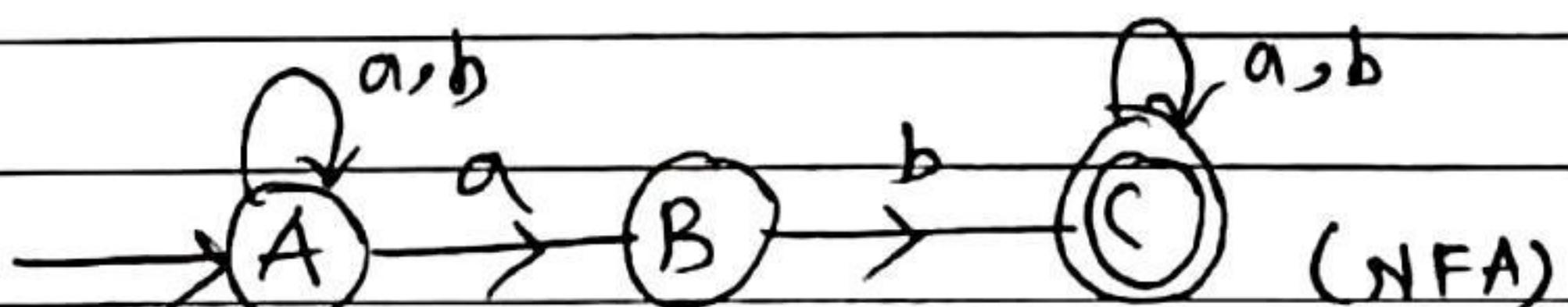
$\rightarrow L = \{ ab, abb, abab, \dots \}$



Ex-6

$L = \{ \text{set of all strings contains 'ab'} \}$

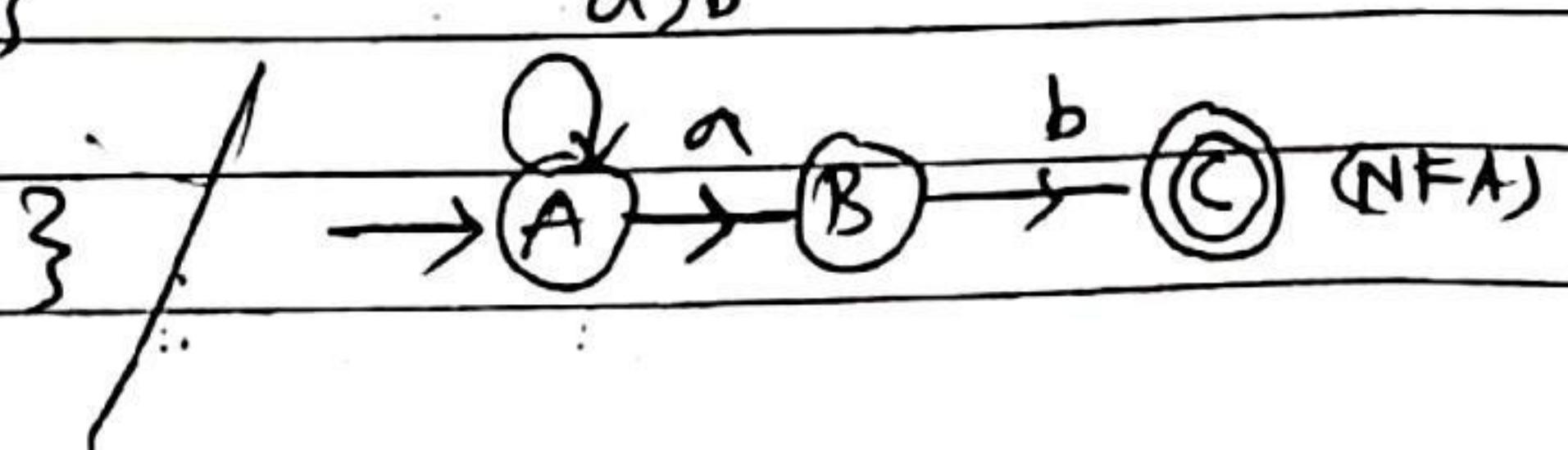
$\rightarrow L = \{ ab, abb, aabb, \dots \}$



Ex-7

$L = \{ \text{ending with 'ab'} \}$

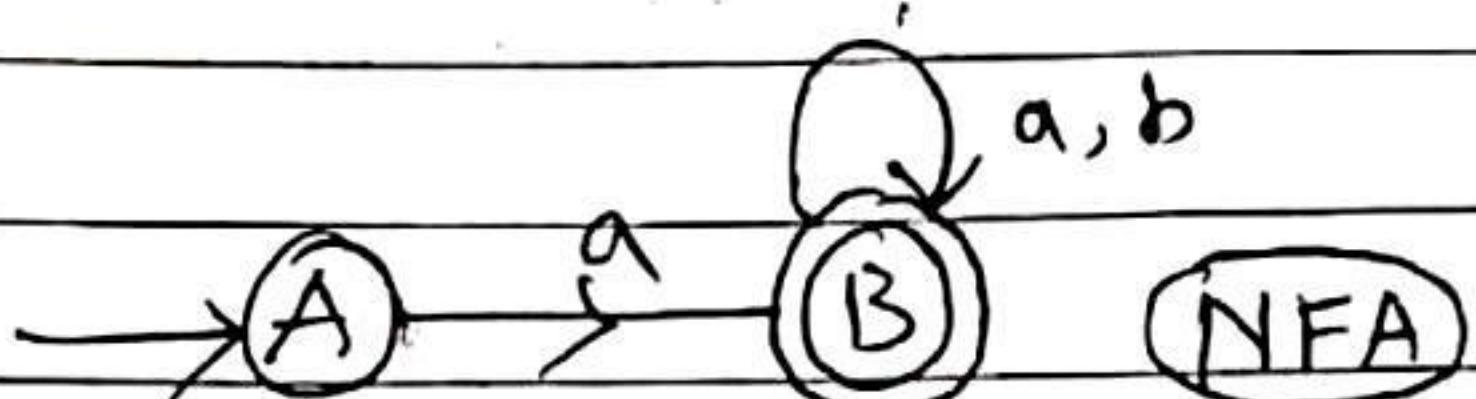
$\rightarrow L = \{ ab, aab, bab, \dots \}$



- CONVERSION of NFA to DFA for the **Example (1)** " all strings start with 'a'".

$$\rightarrow L = \{ a, aa, ab, aab, aba, \dots \}$$

State transition Diagram of NFA .



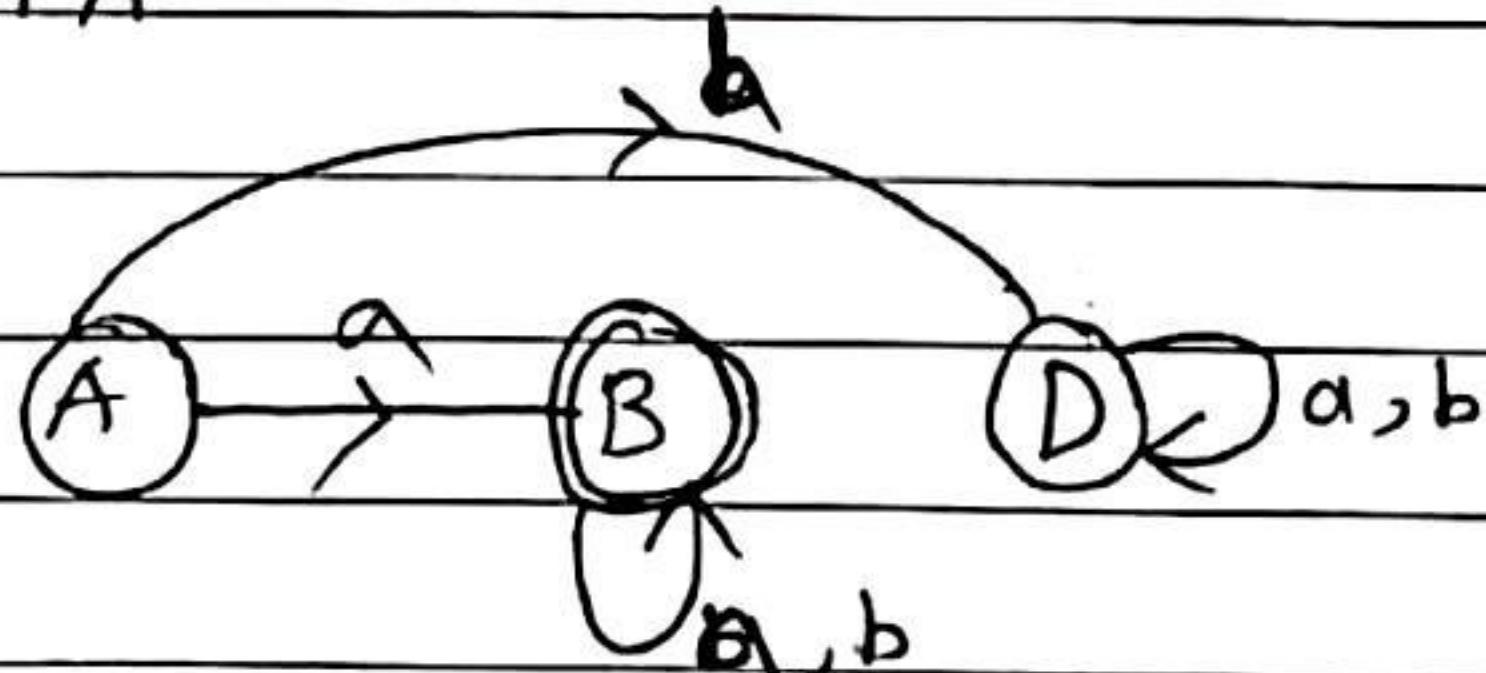
STT of NFA -

	a	b
A	B	\emptyset
*B	B	B

STT of DFA -

	a	b
A	B	D
*B	B	B

STD of DFA -

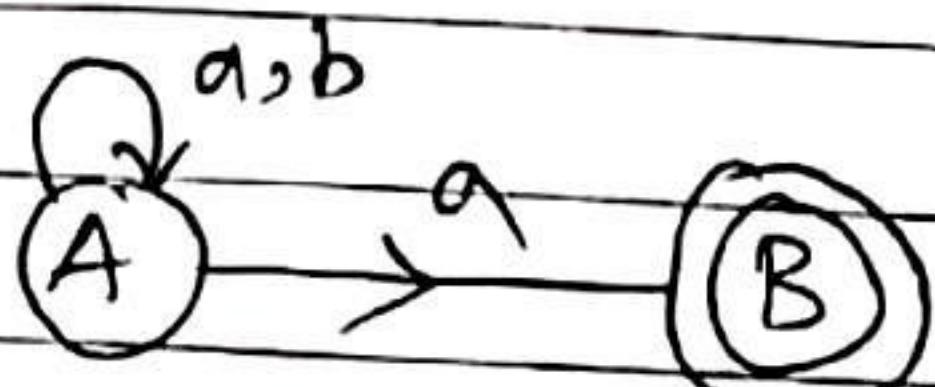


Example-2

Conversion of NFA to DFA :

$L =$ " all strings ends with 'a'" .

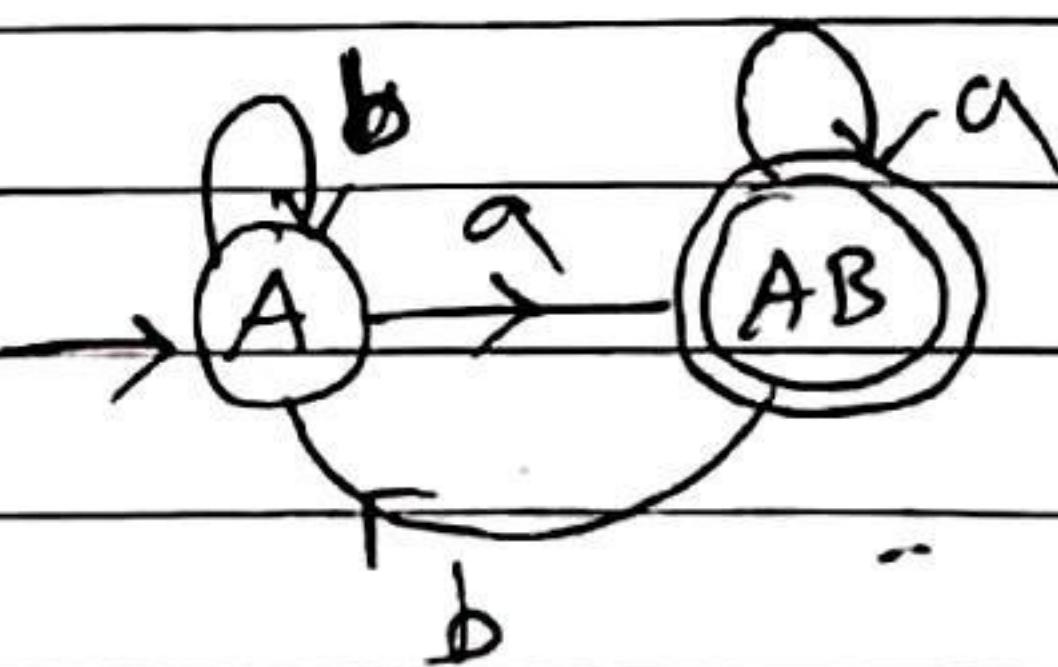
$$\rightarrow \{ a, aa, abba, aad, aba, \dots \}$$

ST-Diagram of NFA -ST-table of NFA

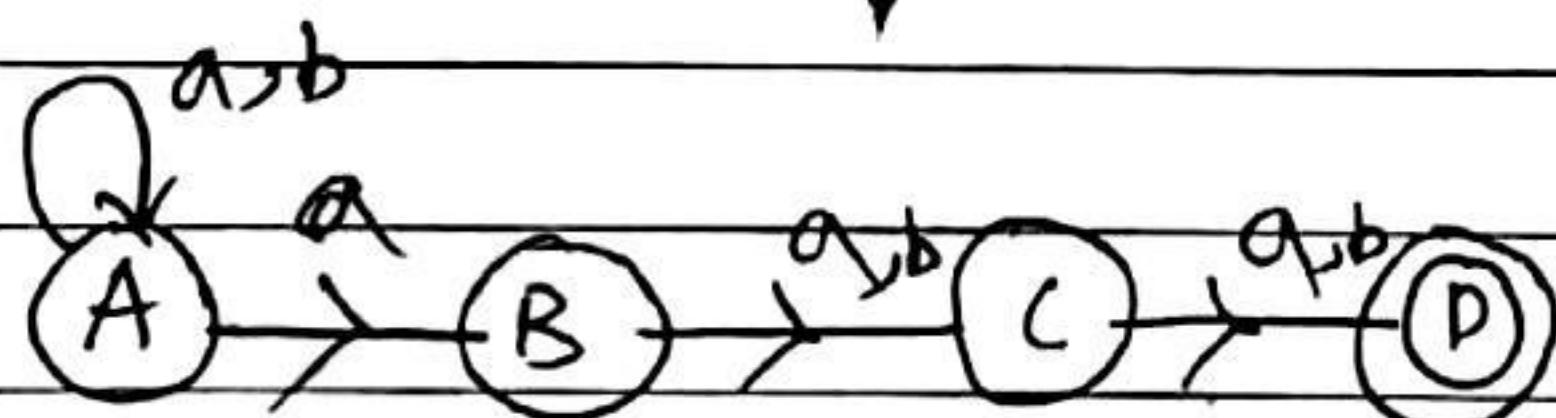
	a	b
A	{A,B}	{A}
B	{Ø}	{Ø}

ST-Table of DFA

	a	b
→ A	[AB]	[A]
* [AB]	[AB]	[A]

ST-Diagram of DFA**Ex-2)**

Conversion of NFA to DFA,

 $L = \{ \text{all strings in which third symbol from R.H.S is 'a'} \}$ $\rightarrow L = \{aaa, abb, aba, baa, \dots\}$ ST-D of NFAST-Table of NFA

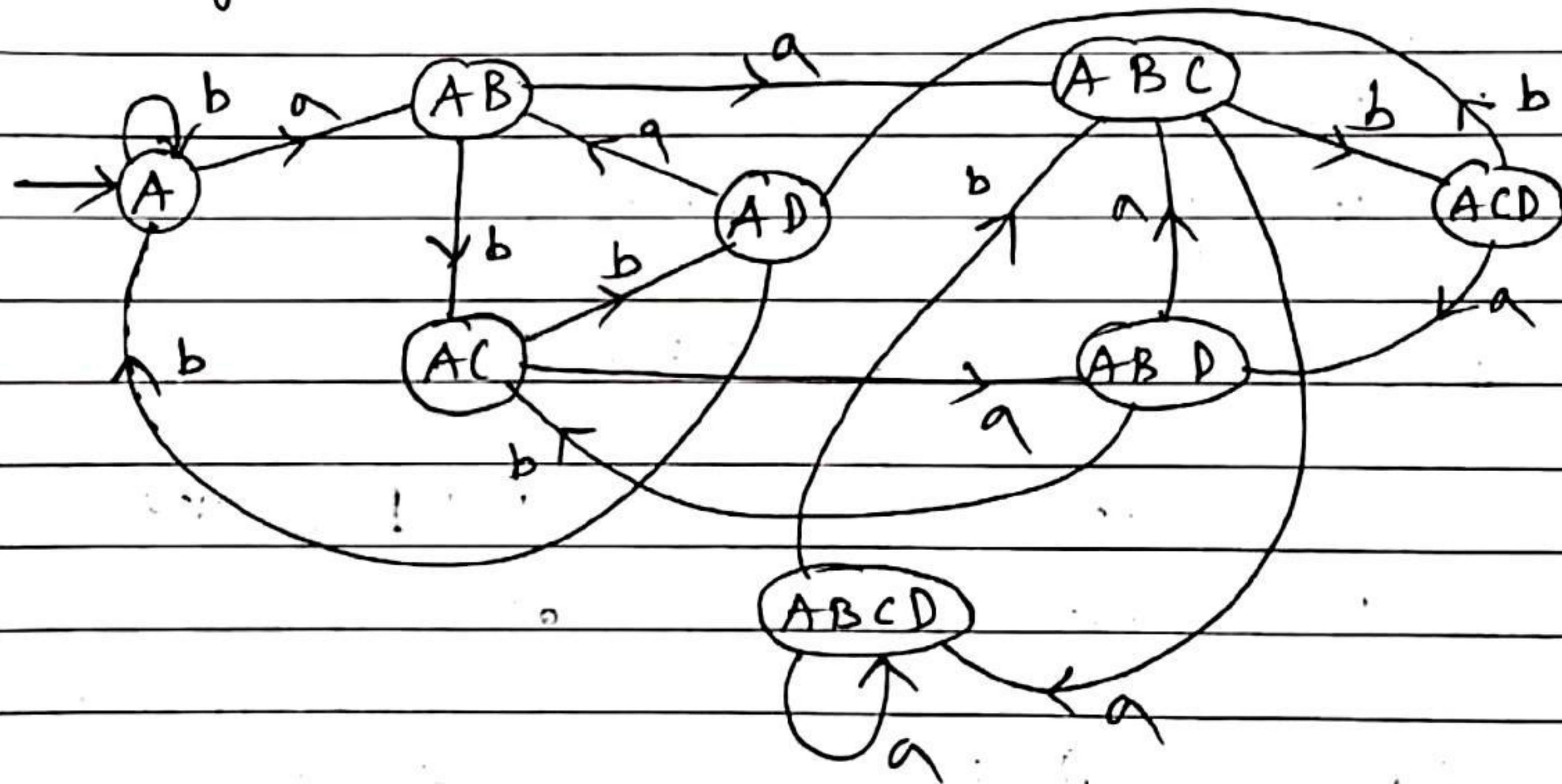
	a	b
→ A	{A,B}	{A}
B	{C}	{C}
C	{D}	{D}
* D	{Ø}	{Ø}

ST-Table of DFA from S-TT of NFA -

	a	b
$\rightarrow [A]$	$[A, B]'$	$[A]'$
$[A, B]$	$[ABC]'$	$[AC]'$
$[AC]$	$[ABD]'$	$[AD]'$
* $[AD]$	$[AB]'$	$[A]'$
$[ABC]$	$[ABCD]$	$[A(D)]'$
* $[ABD]$	$[ABC]'$	$[AC]'$
* $[ACD]$	$[ABD]'$	$[AD]'$
* $[ABCD]$	$[ABCD]'$	$[ACD]'$

any
two
contains

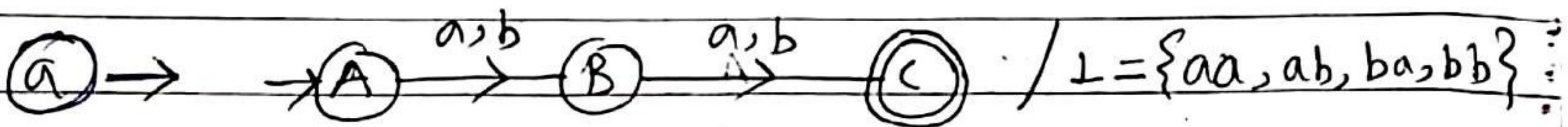
ST-Diagram of DFA from above S-TT of DFA -



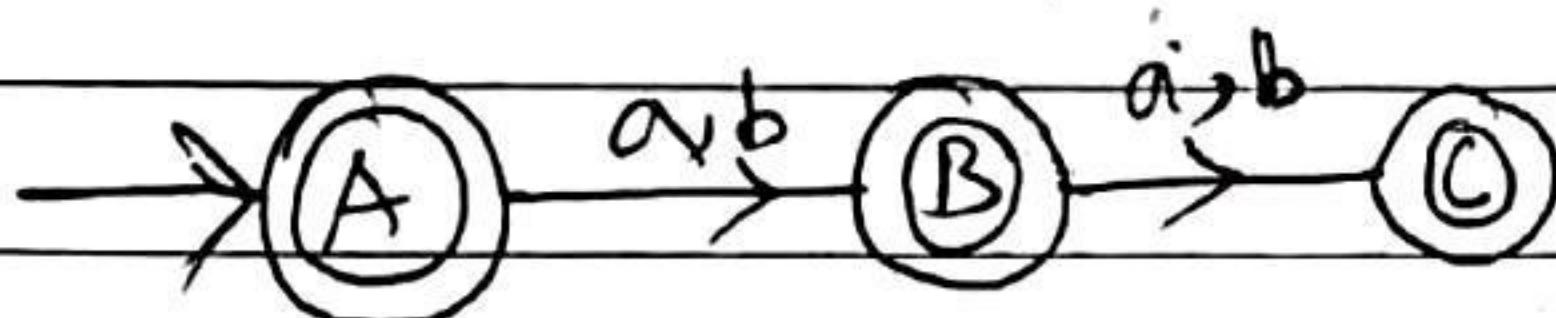
* If an minimal NFA Contain ' n ' states then an DFA Contain 2^n states in worst case.
 $n \rightarrow 2^n$

• Ex-] NFA for string of length-

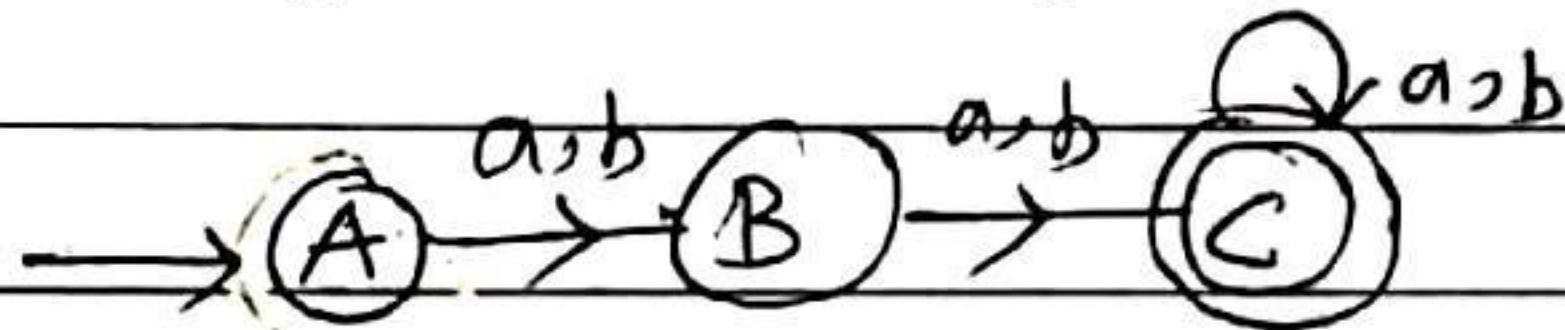
- a) exactly 2. b) at most 2 c) at least 2



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



(c) $\rightarrow L = \{aas, ab, ba, bb, aaa, \dots\}$



** → If we have a string of length 'n' then for NFA it is going to be 'n' states.

• COMPLEMENTATION of NFA -

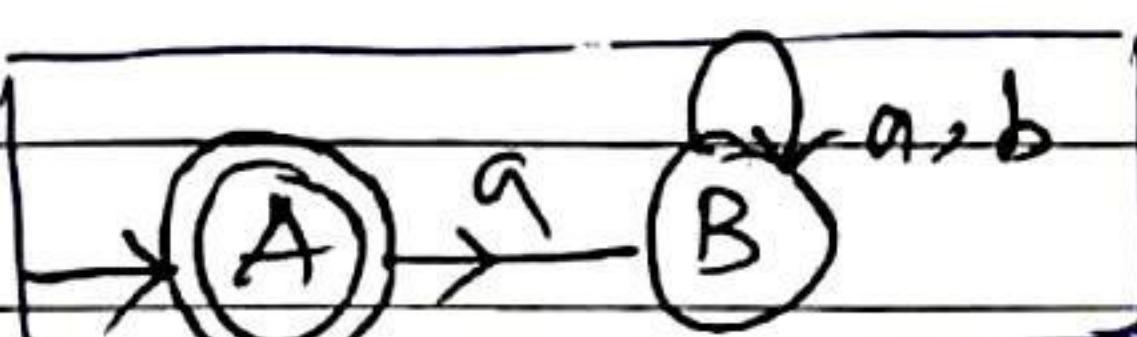
$$\Sigma = \{a, b\}$$

$$L = \{ \text{starts with } a \}$$

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



Then
I =

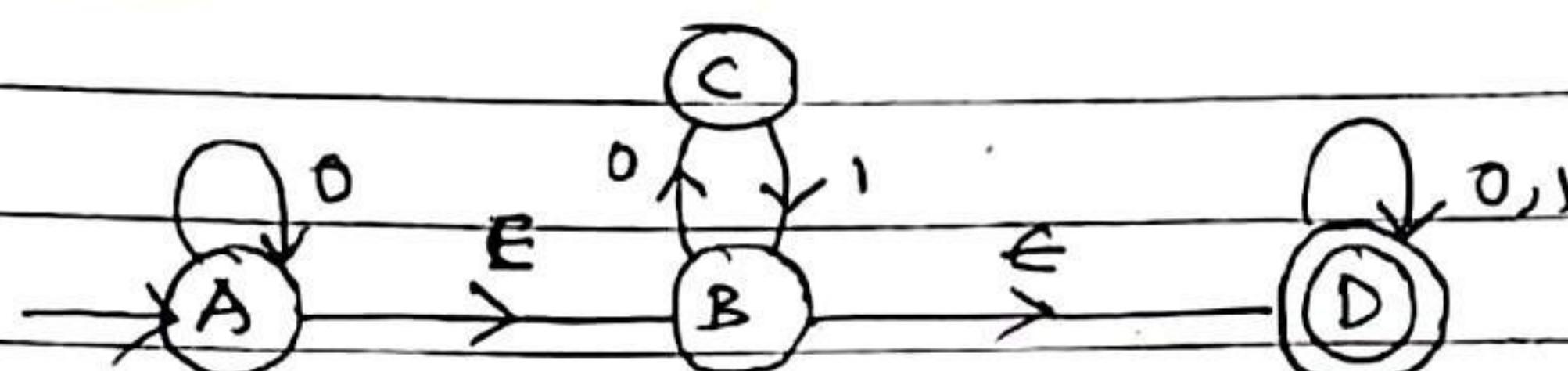


$$[L \neq \bar{L}] (\text{in NFA})$$

• Q-1 What is the language accepted by the complement of NFA.
 $\rightarrow \{\epsilon\}$.

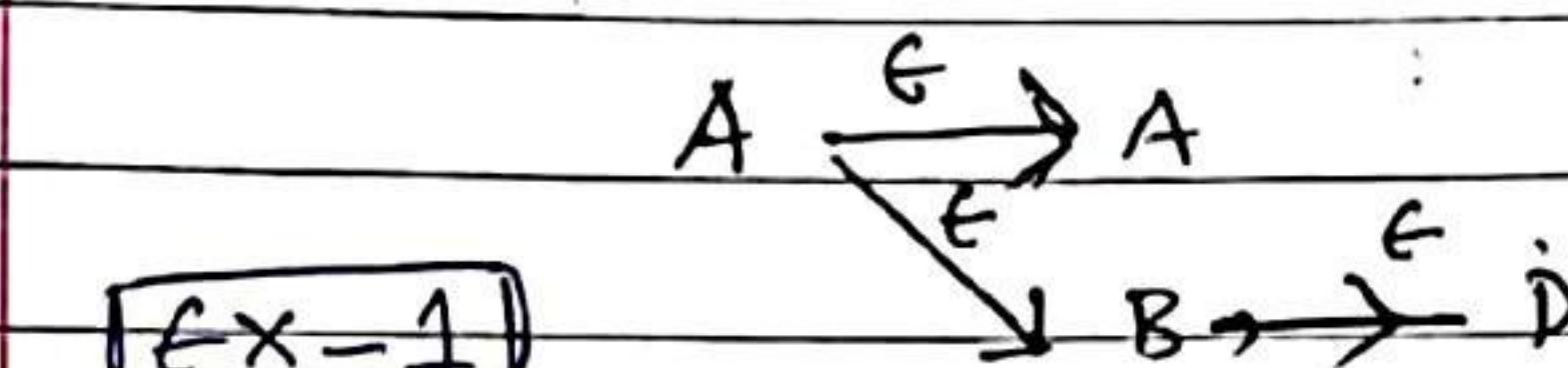
• Q-2 What is the complement of language accepted by NFA.
 $\rightarrow \{\epsilon, b, bb, bbb, \dots, ba\}$.

- Epsilon NFA :- (ε-NFA)

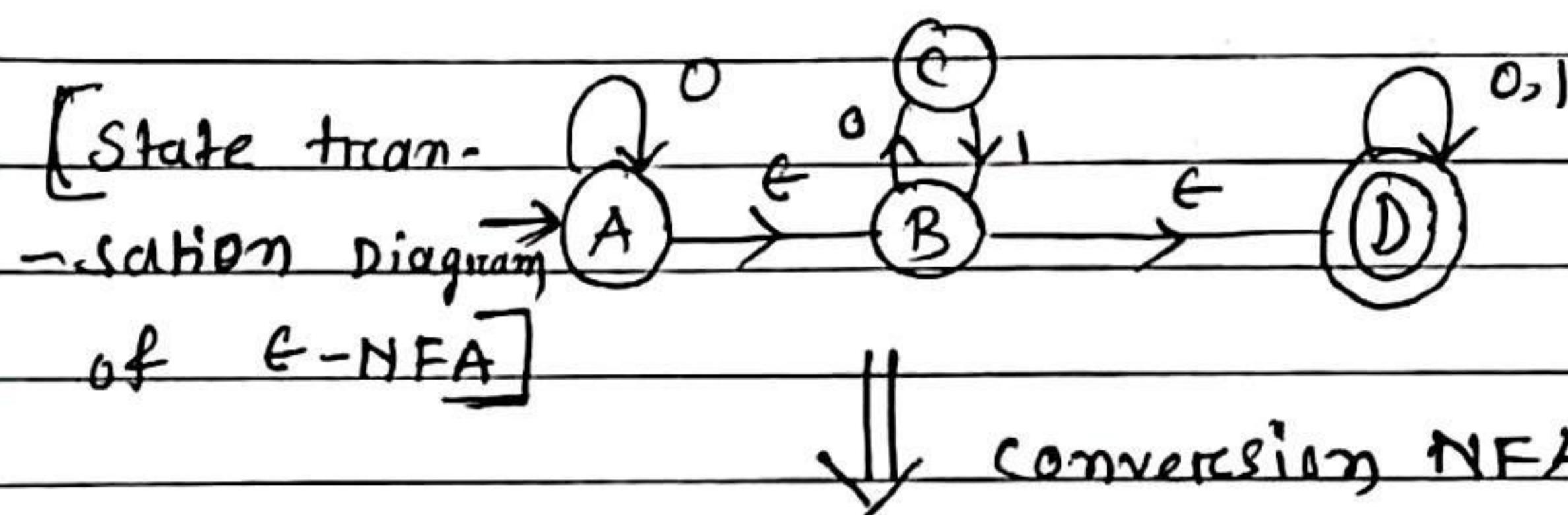


→ For ε-NFA : $Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

→ ε-closure (A) = {A, B, D}

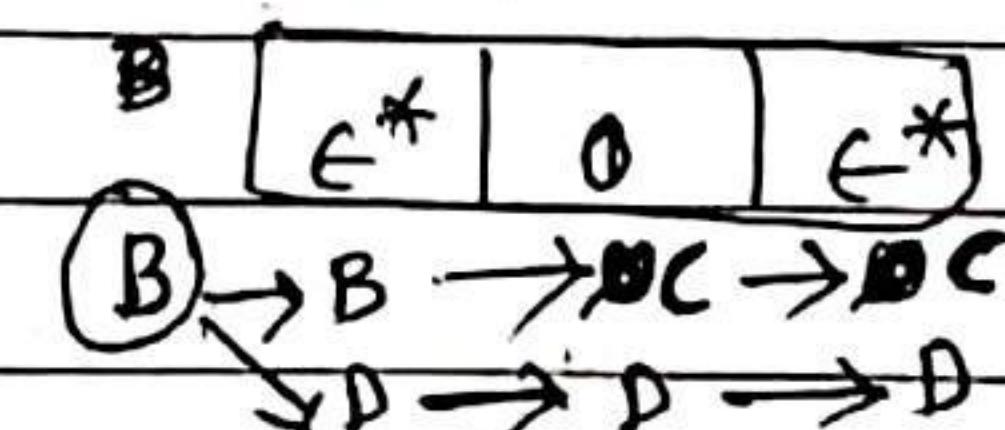
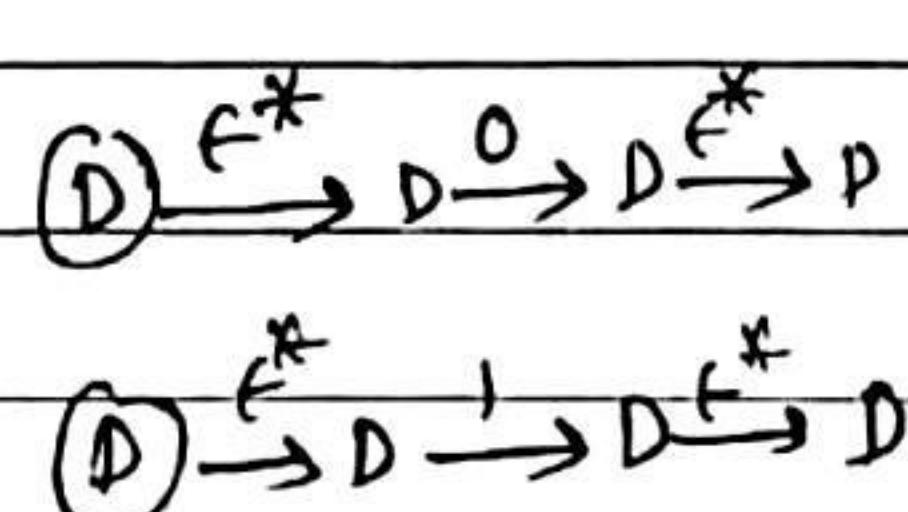
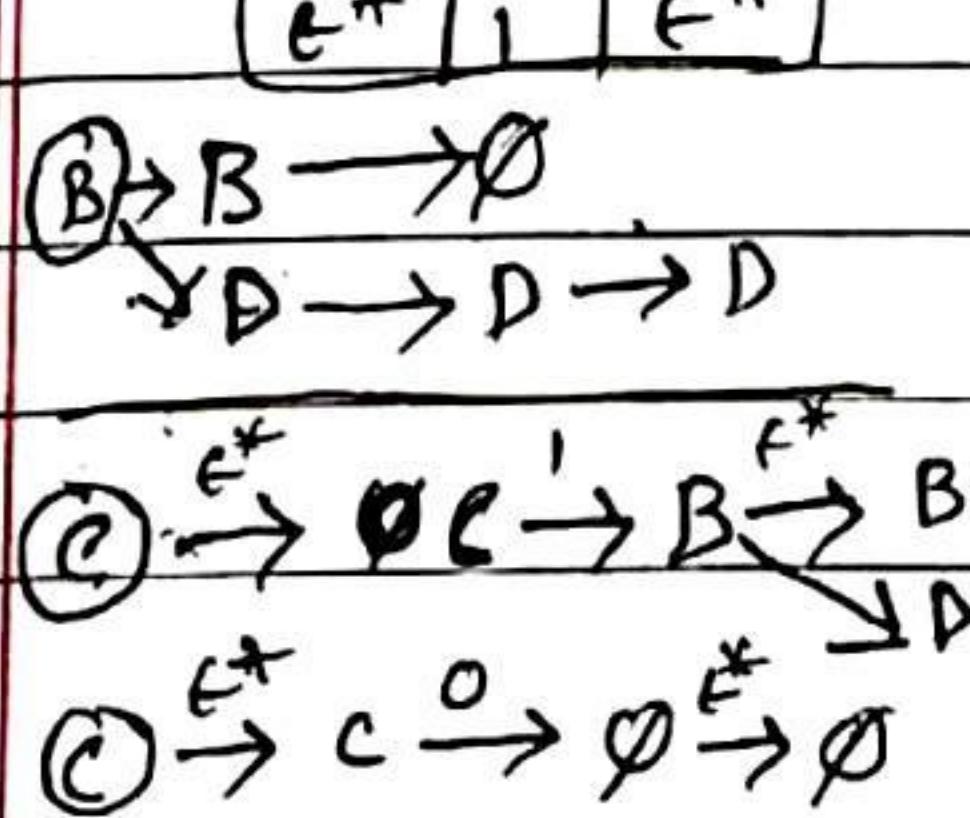
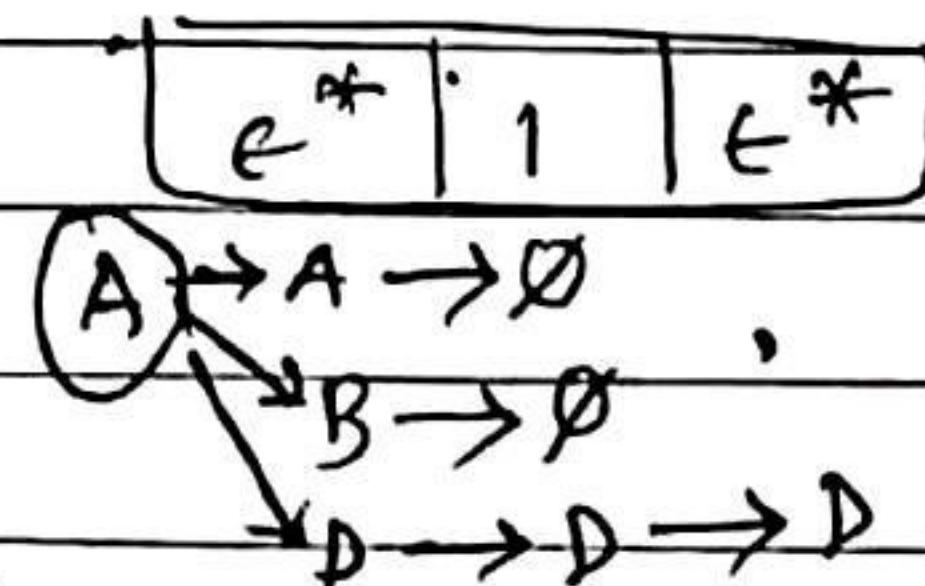
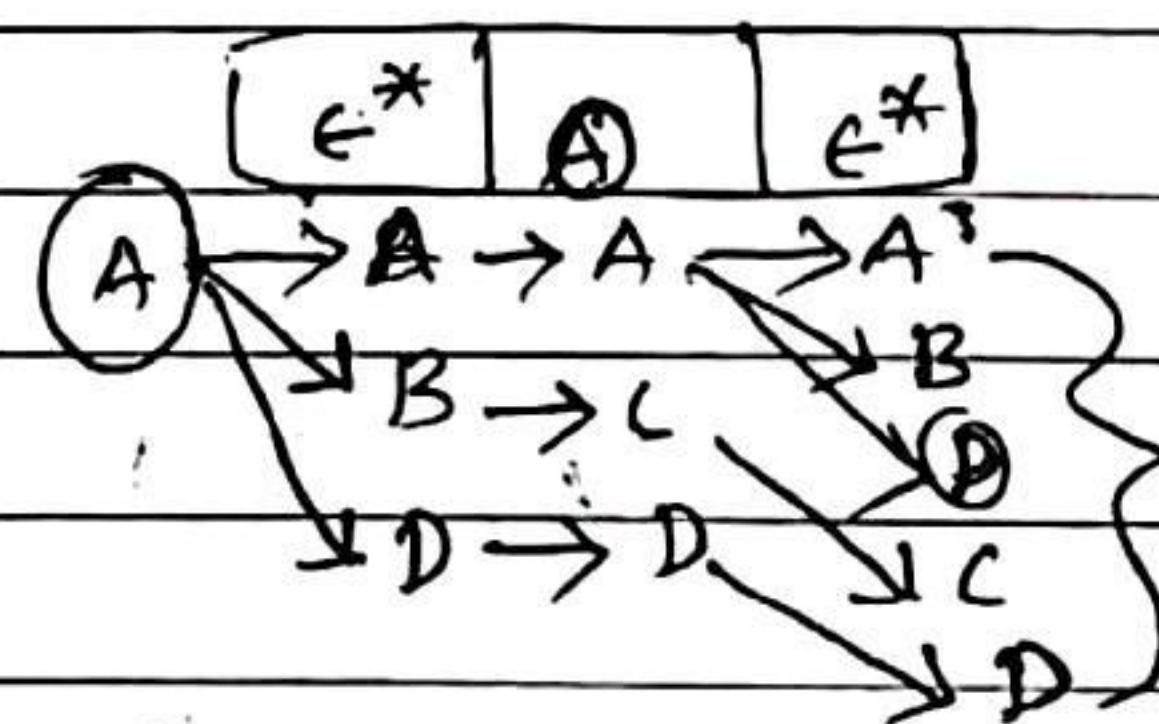


- Conversion ε-NFA to NFA :

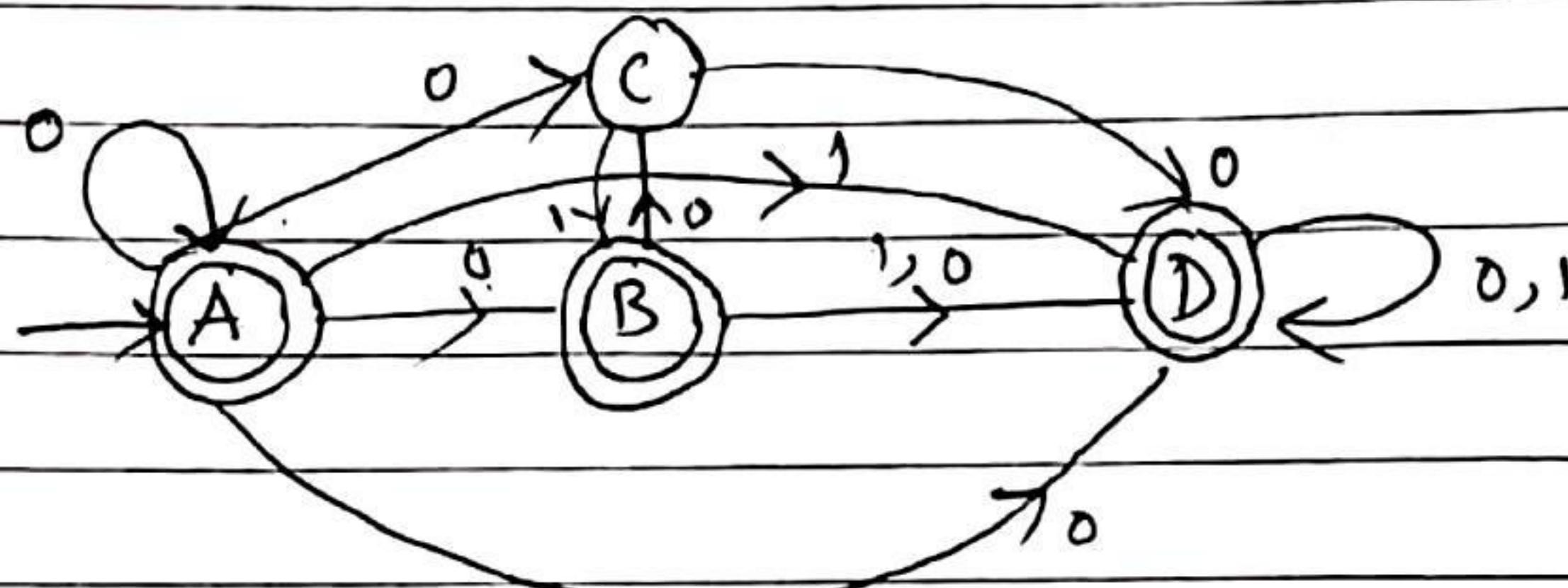


State transition table of NFA :

	0	1
A	{A, B, C, D}	{D}
B	{C, D}	{D}
C	{∅}	{B, D}
D	{D}	{D}



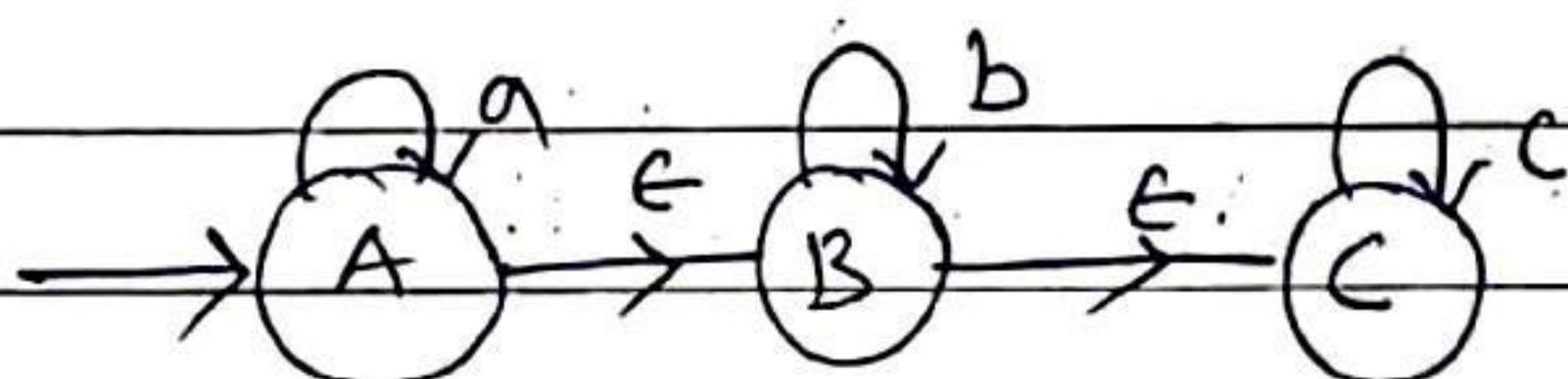
from the state transition diagram :-



- 'A' is going to 'D' by seeing ' ϵ ' so 'A' will be the final state.
- 'B' is going to 'D' by seeing ' ϵ ' so 'B' will be the final state.

Ex-2

conversion \in NFA \rightarrow NFA.

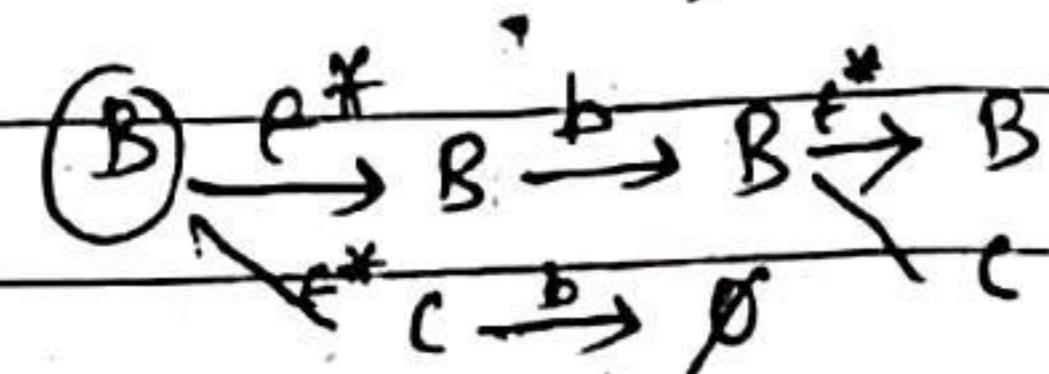
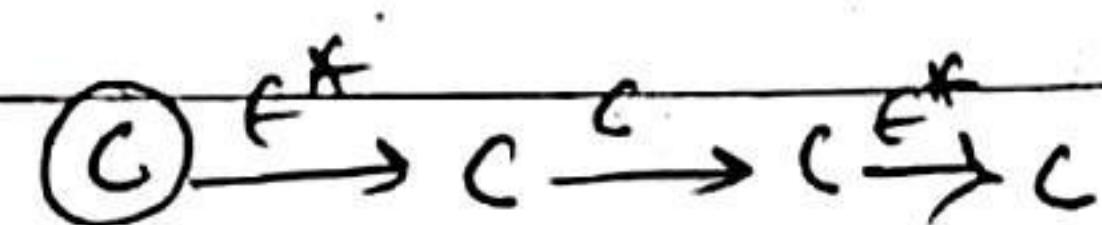
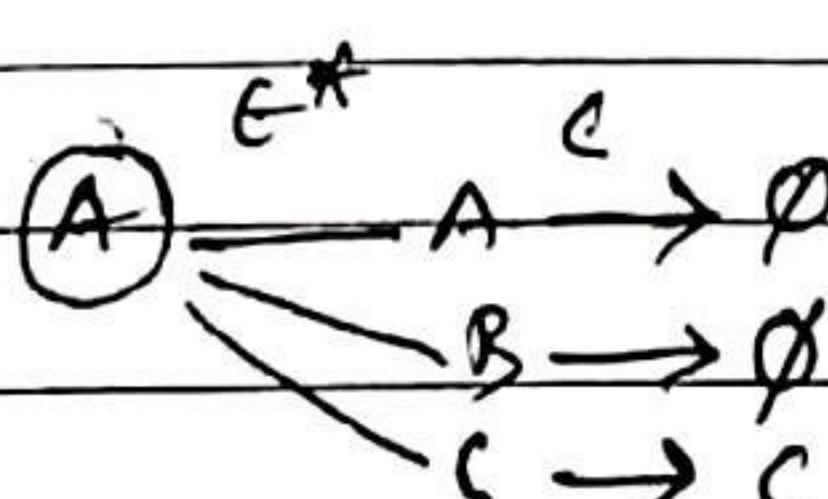
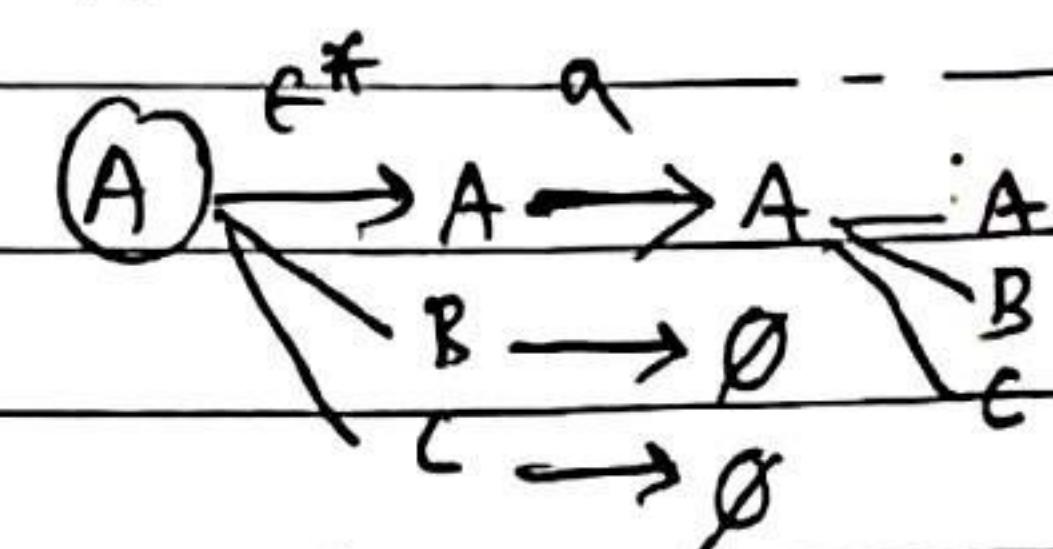
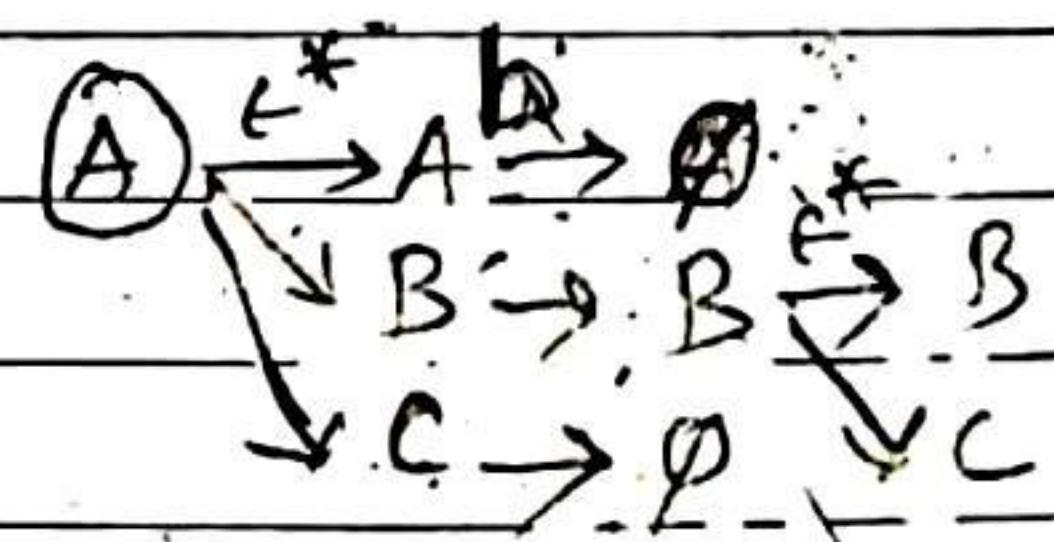


(\in NFA)

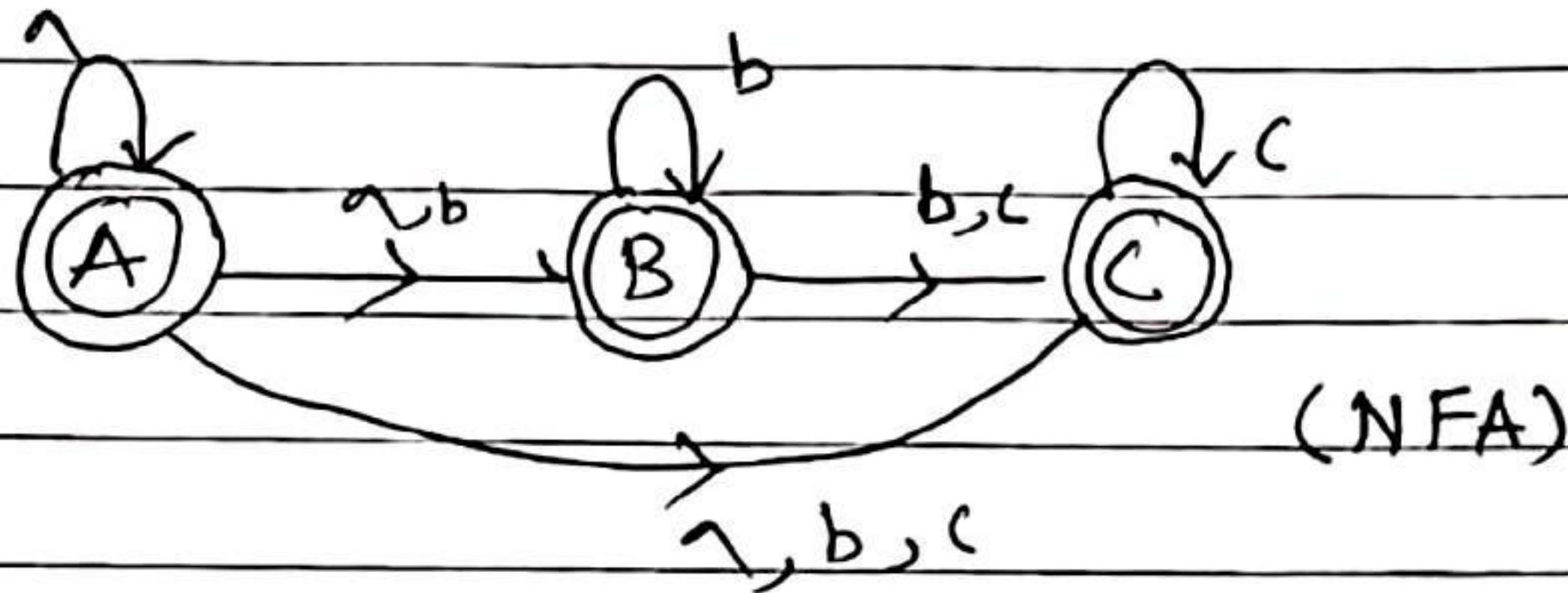
↓ conversion.

transcation table :-

	a	b	c
A	$\{\emptyset, B, C\}$	$\{B, C\}$	$\{C\}$
B	$\{\emptyset\}$	$\{B, C\}$	$\{C\}$
C	$\{\emptyset\}$	$\{\emptyset\}$	$\{C\}$



State transition Diagram for from state transition table:



→ [all the DFA, NFA and E-NFA are equal in power.]

Minimisation of DFA:

→ ** Two state called equivalent,
 $S(p, w) \in F$
 $\Rightarrow S(q, w) \in F$

Or
 $S(p, w) \notin F$
 $\Rightarrow S(q, w) \notin F$

When,

length of string $|w|=0$, 0 equivalent.

$|w|=1$, 1 equi.

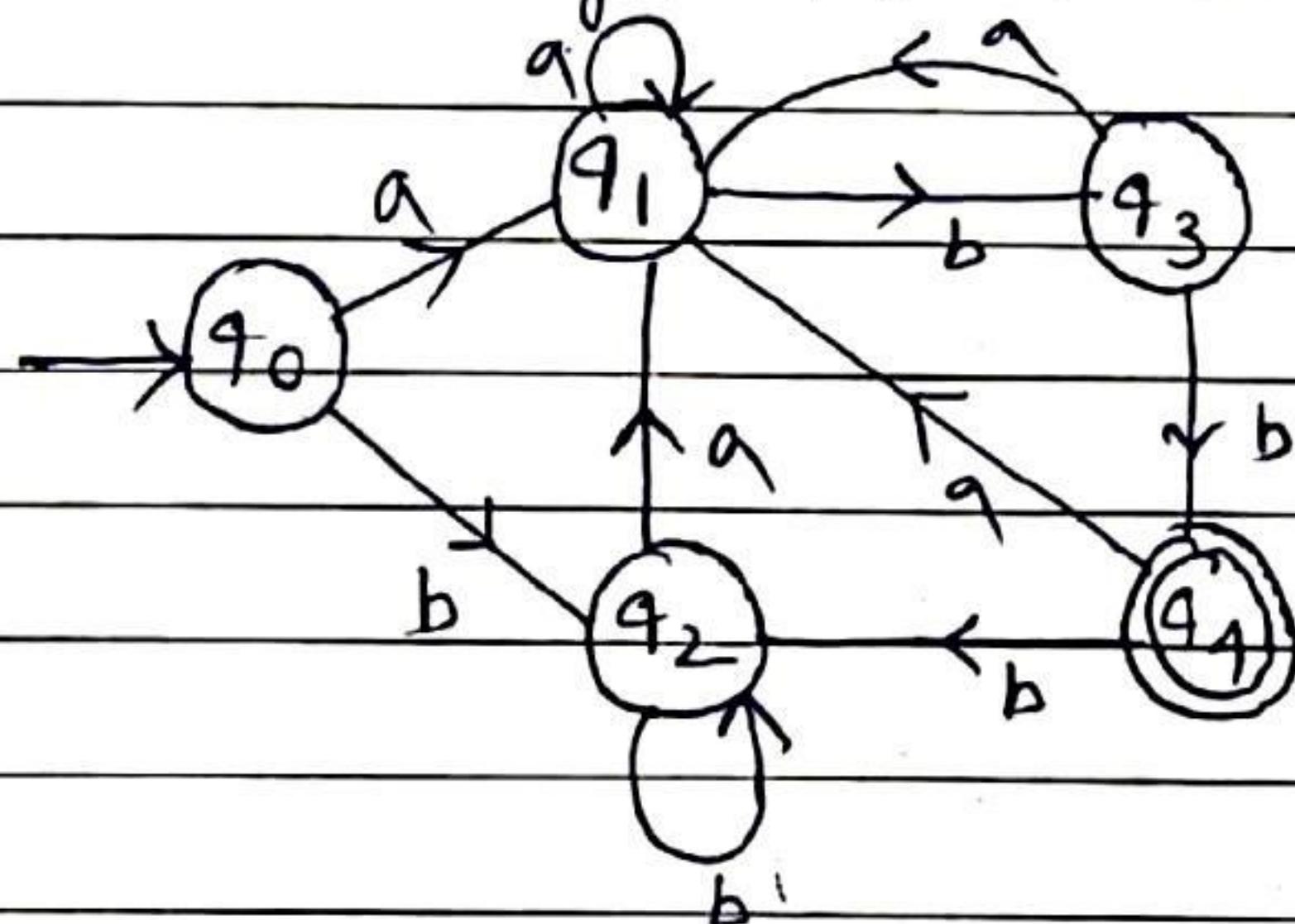
$|w|=2$, 2 equi.

}

$|w|=n$, n equivalent.

Example-1

Minimise the given DFA -



→ Step-1 : Identify the Initial state and final state.

Step-2 : Delete all the state that not reachable ^{in final stat} from initial state.

Step-3 : Draw State transition table.

	a	b	
$\rightarrow q_0$	q_1	q_2	
q_1	q_1	q_3	
q_2	q_1	q_2	
q_3	q_1	* q_4	
* q_4	q_1	q_2	

(1) Find out '0' equivalent State,

$$[q_0, q_1, q_2, q_3] \quad [q_4]$$

(2) Find out '1' equivalent State,

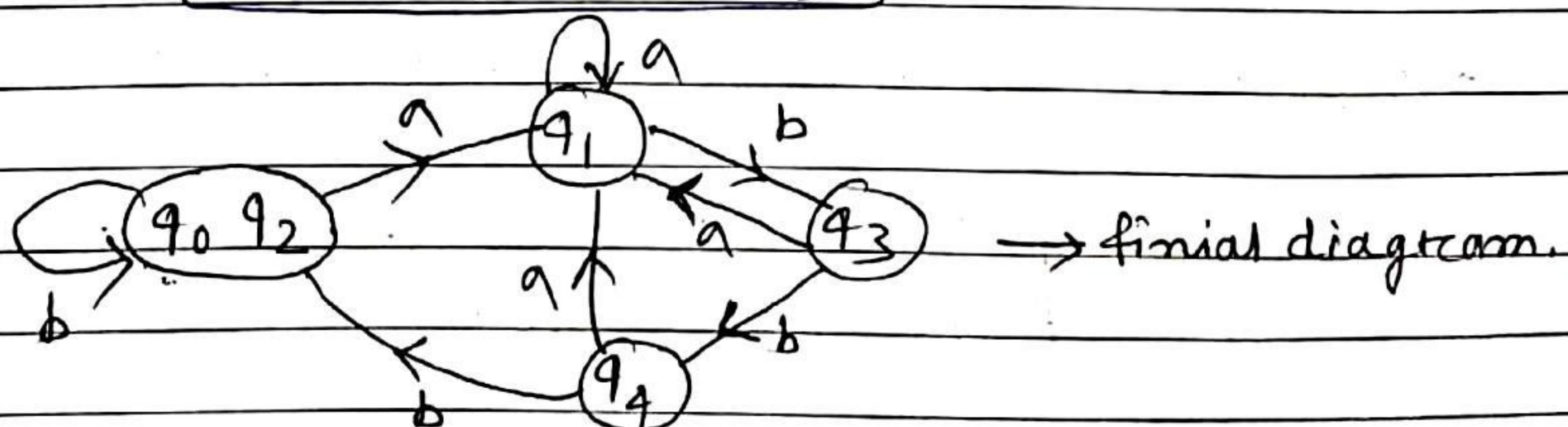
$$[q_0, q_1, q_2] \quad [q_3] \quad [q_4]$$

(3) Find out '2' equi state,

$$[q_0, q_2] \quad [q_1] \quad [q_3] \quad [q_4]$$

(4) Find out '3' equi state,

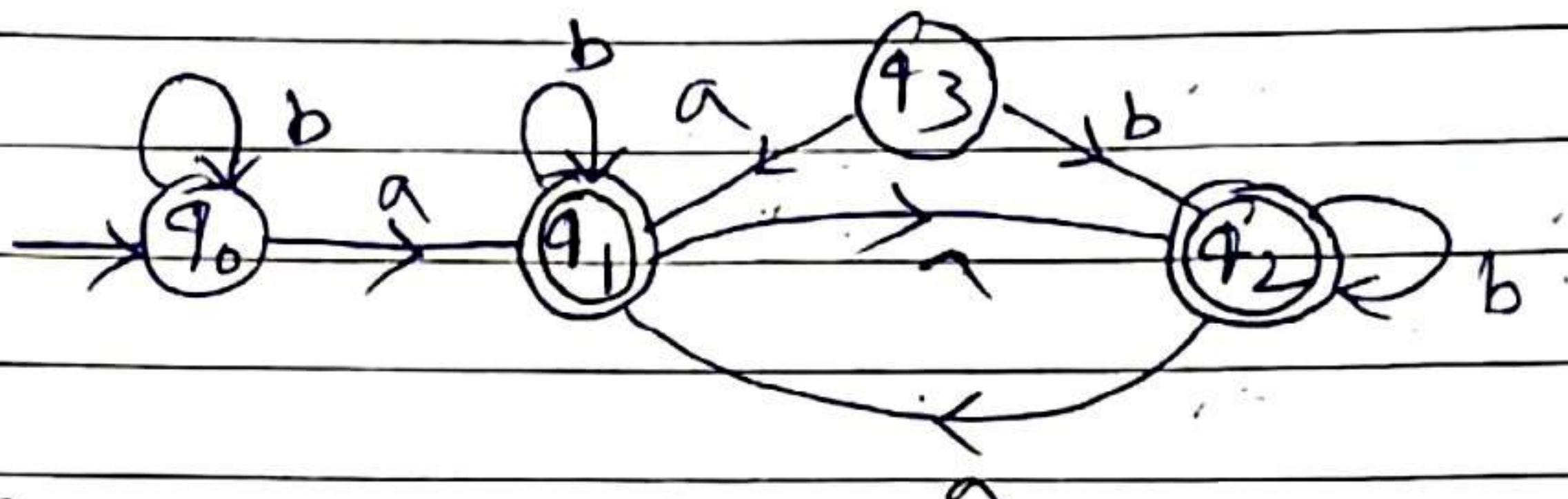
$$[q_0, q_2] \quad [q_1] \quad [q_3] \quad [q_4]$$



* State can be decrease, If you find out some state that are equivalent.

• Example - 2 (gate)

Minimise the given DFA -



→ Remove state q_3 , because we can't reach in q_3 state from initial state q_0 .

→ State Transition Table -

	a	b
q_0	q_1	q_0
q_1	* q_2	q_1
* q_2	q_1	* q_2
q_3	q_1	q_2 X

(1) Findout '0' equivalent state, $[q_0] \quad [q_1, q_2]$
 Non-F F

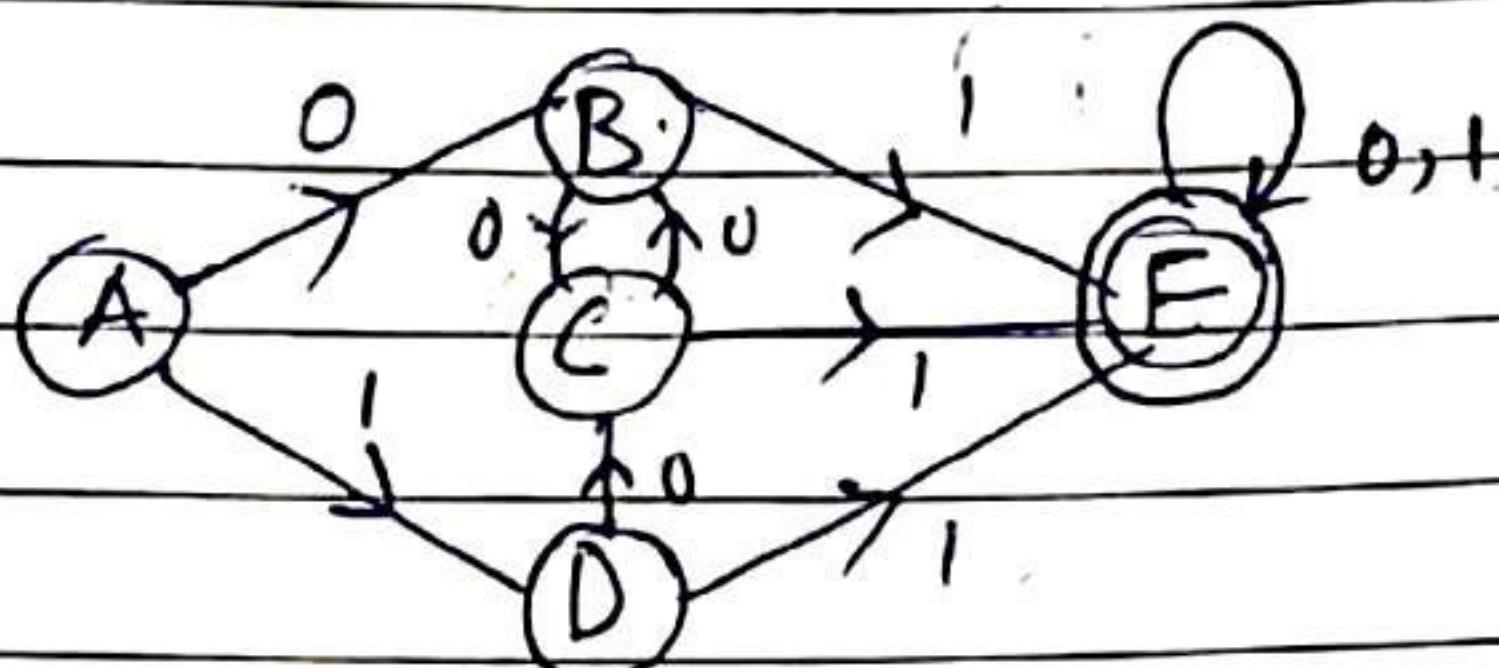
(2) " " " " " , $[q_0] \quad [q_1, q_2]$

final STD of DFA -



Example-3

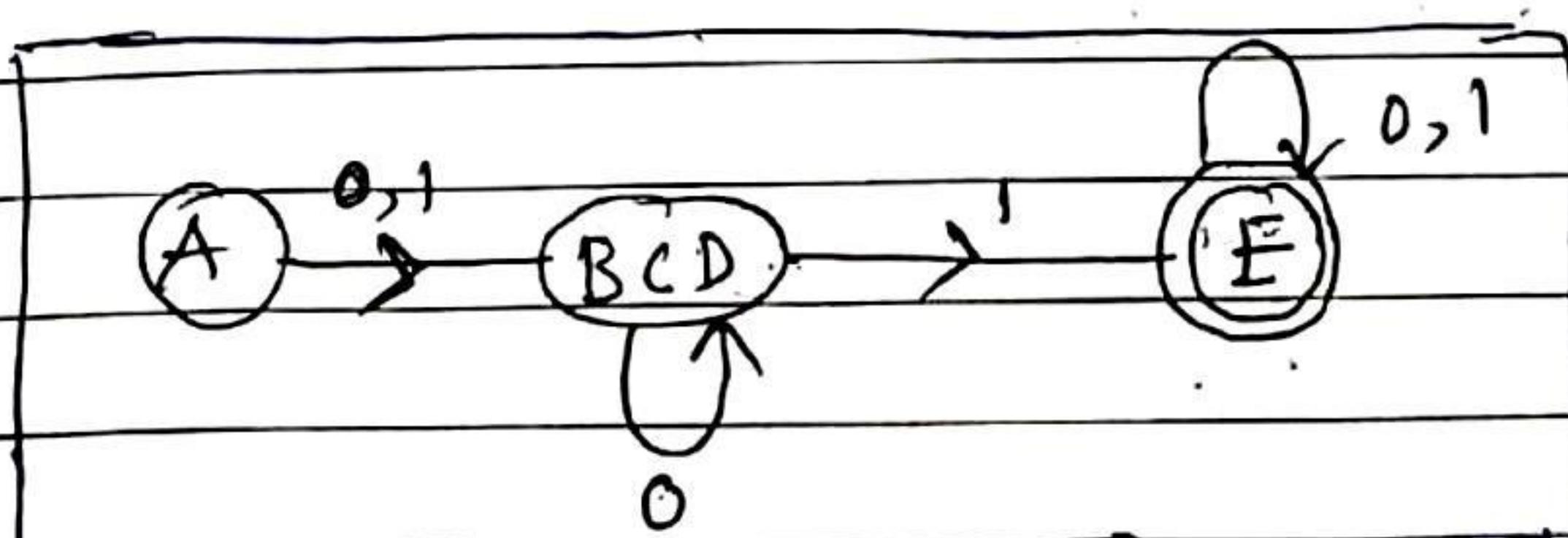
Minimise given DFA -



\rightarrow	0	1
A	B	D
B	C	*E
C	B	*E
D	C	*E
*E	*E	*E

'0' equivalent, $[A, B, C, D] [E]$.'1' equivalent, $[A] [B, C, D], [E]$ '2' equivalent, $[A] [B, C, D] [E]$

∴ Final State Transition Diagram,-



• Equivivalence of DFA and NFA:

- In contrast to NFA, the DFA has:
 - No ϵ -transition.
 - For every (q, a) with $q \in Q$ and $a \in \Sigma$ at most one successor state.
 - DFA can simulate the behaviour of NFA by increasing no of state.
 - In DFA for each state, there is at most one transition for each possible input.
- In NFA there can be more than one transition from a given state for a given possible input.
- All DFA are NFA, but not all NFA are DFA.
- All NDFA (NFA) and DFA have the same power.
- Processing of an input string is more time consuming when NFA is used and less time consuming when DFA is used.

• MOORE AND MEALY MACHINE:

• Introduction of moore and mealy machines-

→ FA with output capabilities -

There are two types of machines : moore machine and mealy machine.

• Moore machine → It is a FA in which output is associated with each state.

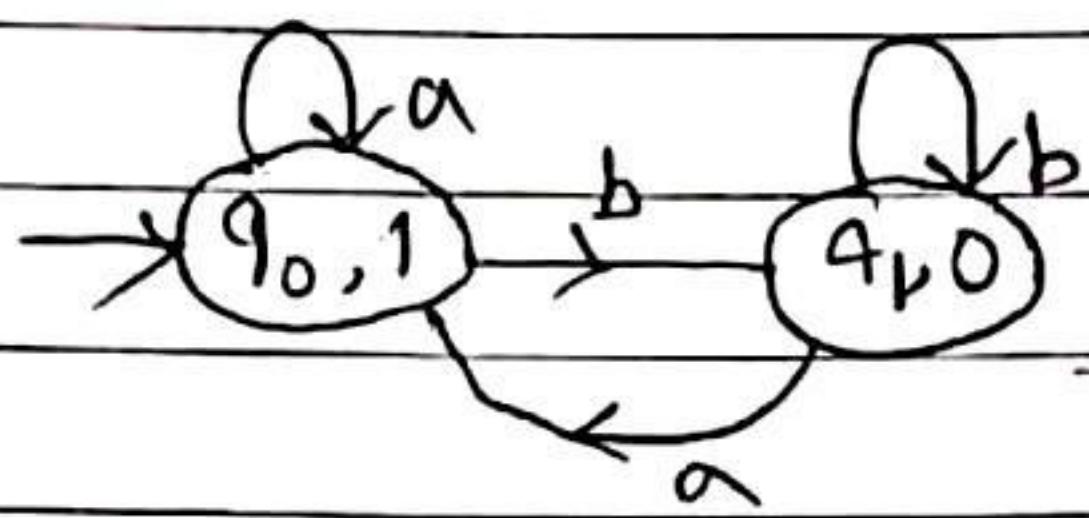
• Mealy machine → It is a FA in which output depends upon both the present input and present state

$$n \cdot i/p = (n+1) o/p \dots$$

DFA with output (O/P)

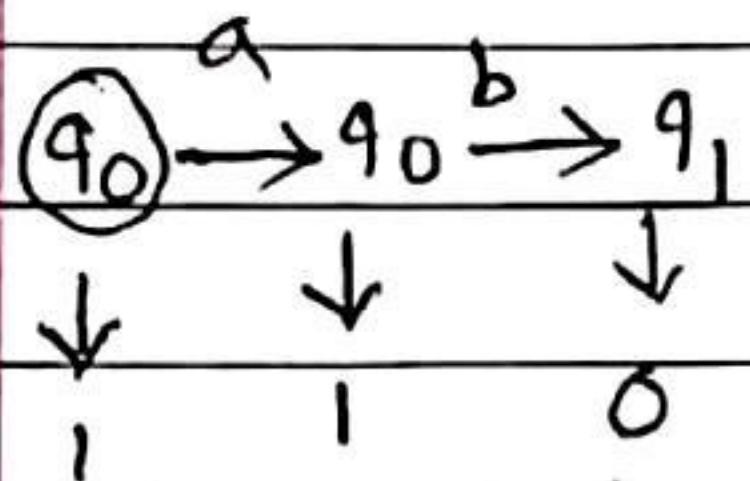
$$n i/p = n o/p$$

Moore
m/c



$$\lambda: Q \rightarrow \Delta$$

$$q_0 \rightarrow 1, q_1 \rightarrow 0$$



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

$Q \rightarrow$ finite set of states.

$\Sigma \rightarrow$ i/p alphabet.

$\delta \rightarrow$ Transition function

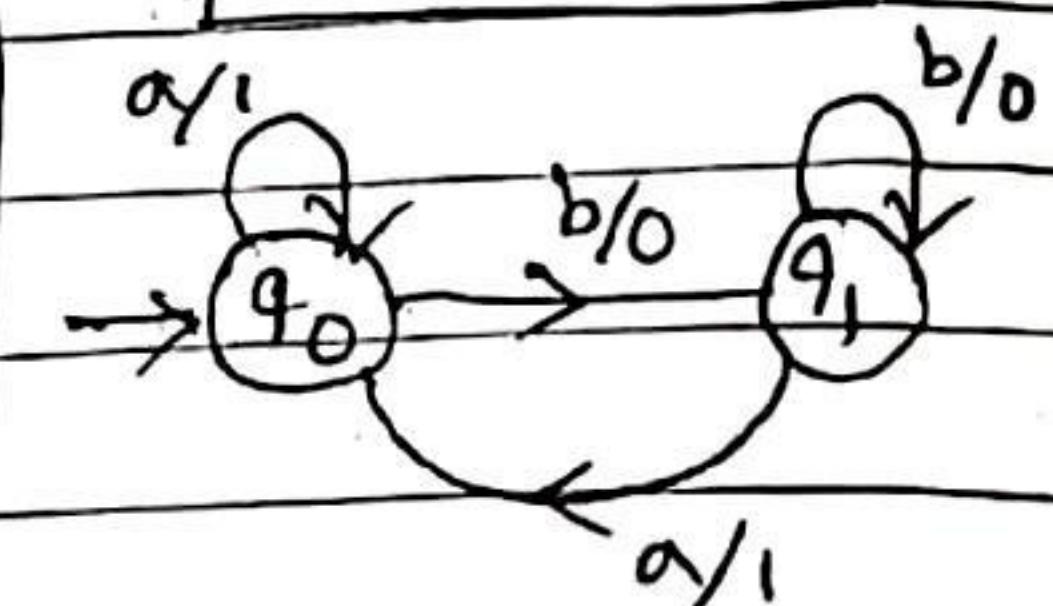
$$Q \times \Sigma \rightarrow Q$$

$q_0 \rightarrow$ initial state

$\Delta \rightarrow$ o/p alphabet.

$\lambda \rightarrow$ o/p function.

mealy m/c



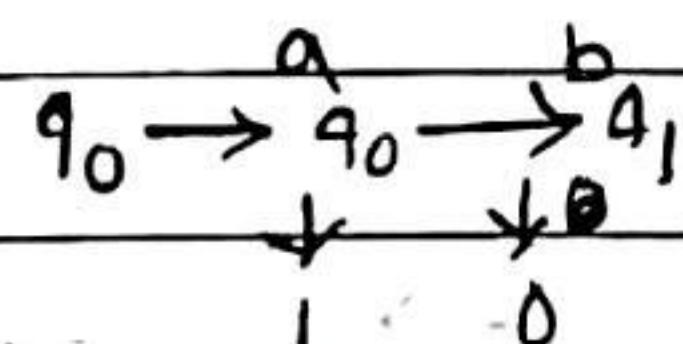
$$\lambda: Q \times \Sigma \rightarrow \Delta$$

$$(q_0, a) \rightarrow 1$$

$$(q_0, b) \rightarrow 0$$

$$(q_1, a) \rightarrow 1$$

$$(q_1, b) \rightarrow 0$$



• Example - 1

Moore Machine example -

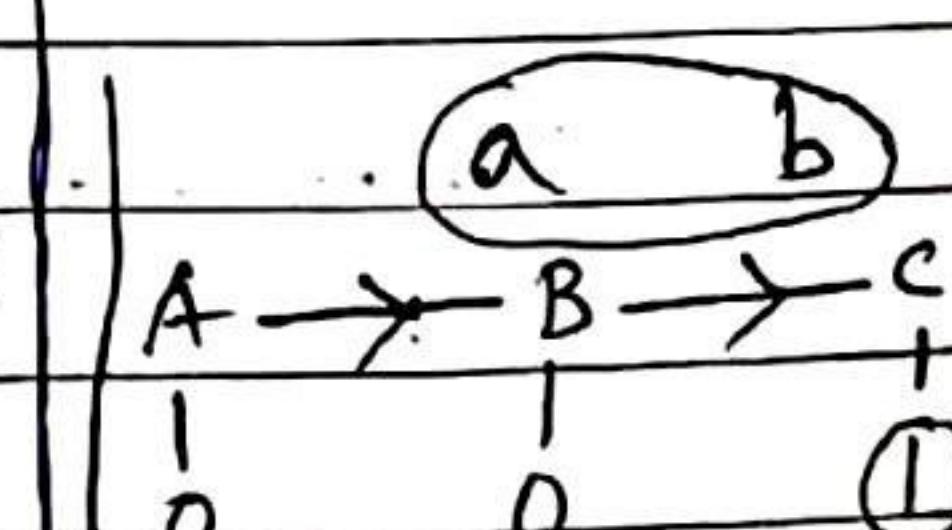
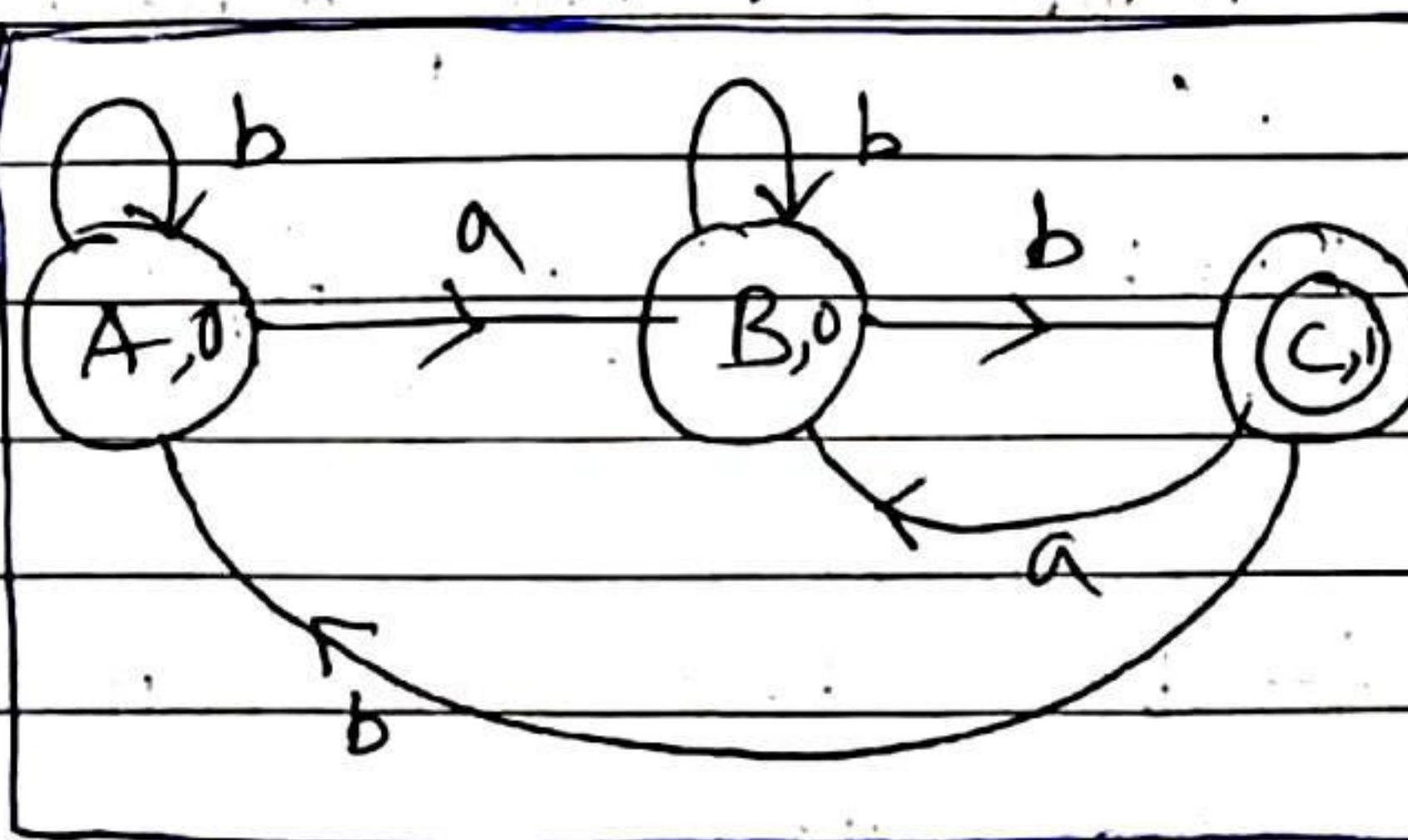
Construct a moore machine first take set of all strings over $\{a, b\}$ as i/p and prints '1' as o/p for every occurrence of 'ab' as a substring.

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

→ for every ab it is going to print '1'.

DFA

(ending
with ab)

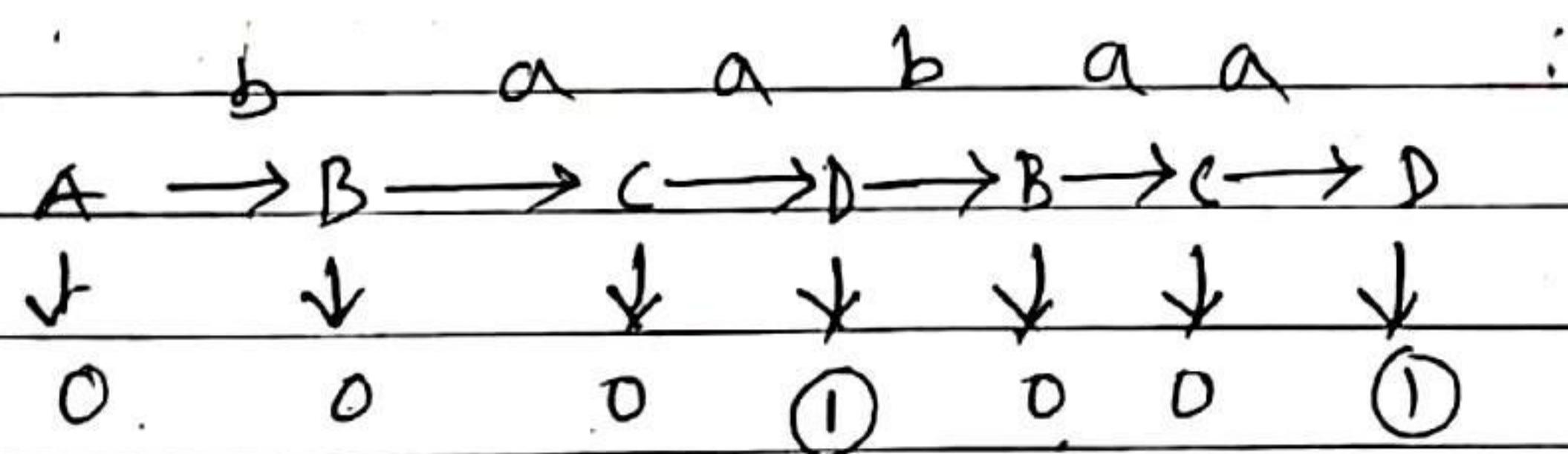
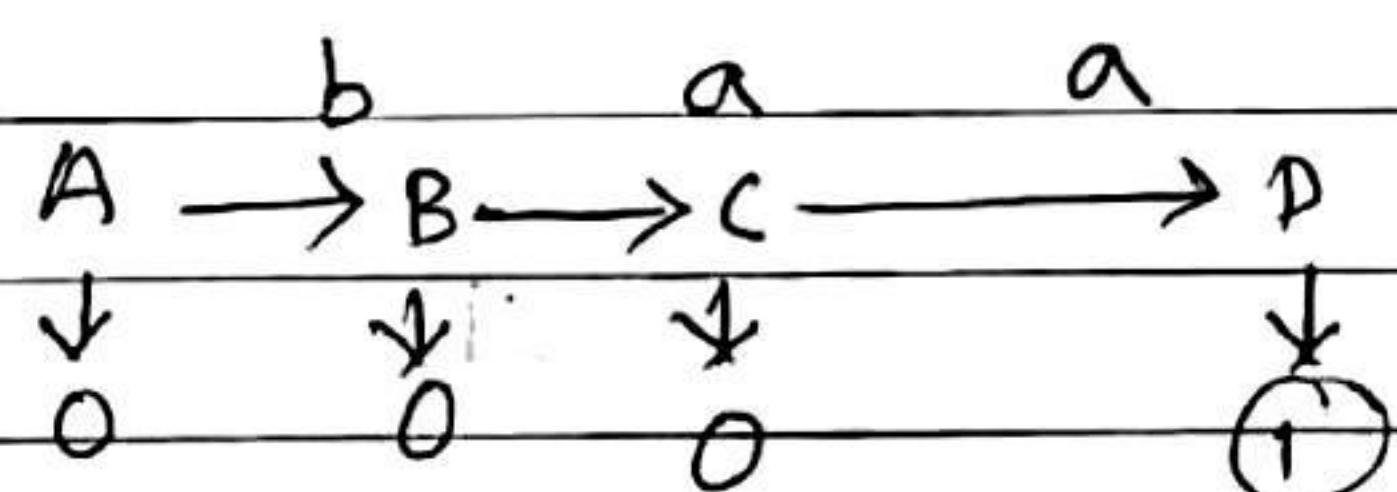
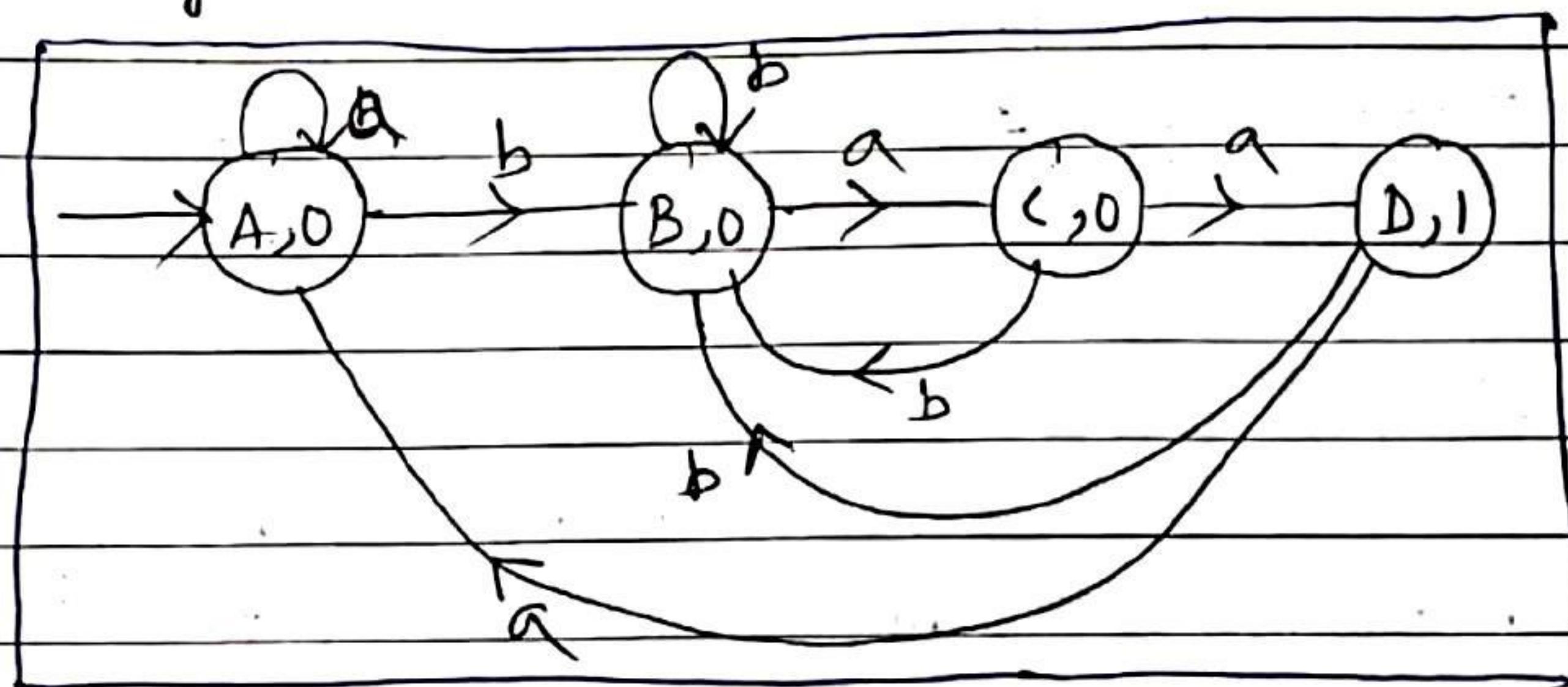


Ex-2

Construct a moore m/c that take set of all strings over $\{a, b\}$ and counts no of occurrences of string 'baa'.

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

ending with 'baa'.

**Ex-3**

Construct a moore machine that takes set of all strings over $\{0, 1\}$ and produces 'A' as o/p if i/p ends with '10' or produces 'B' as o/p if it ends with '11' otherwise produces 'C'.

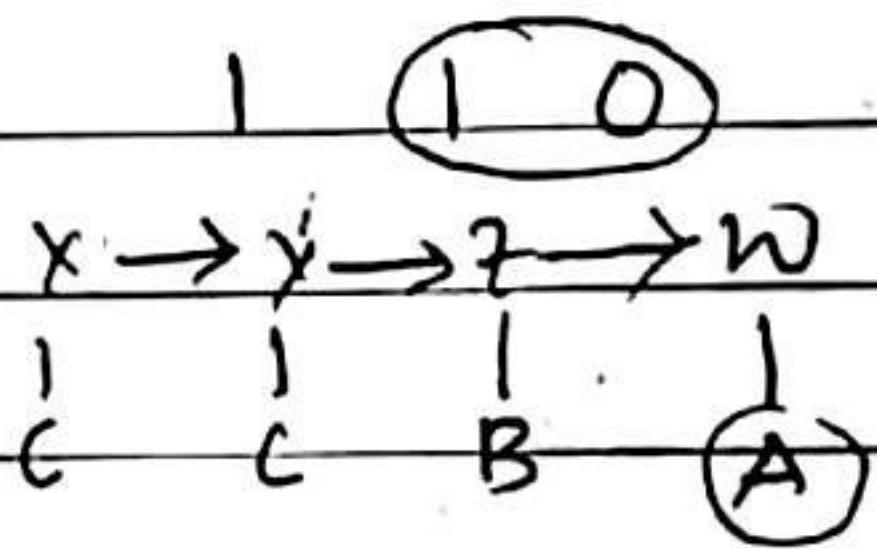
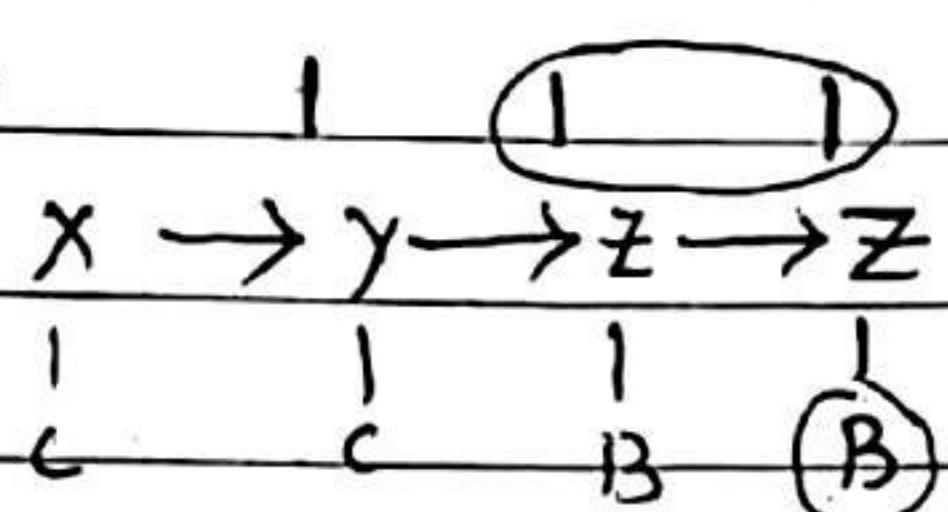
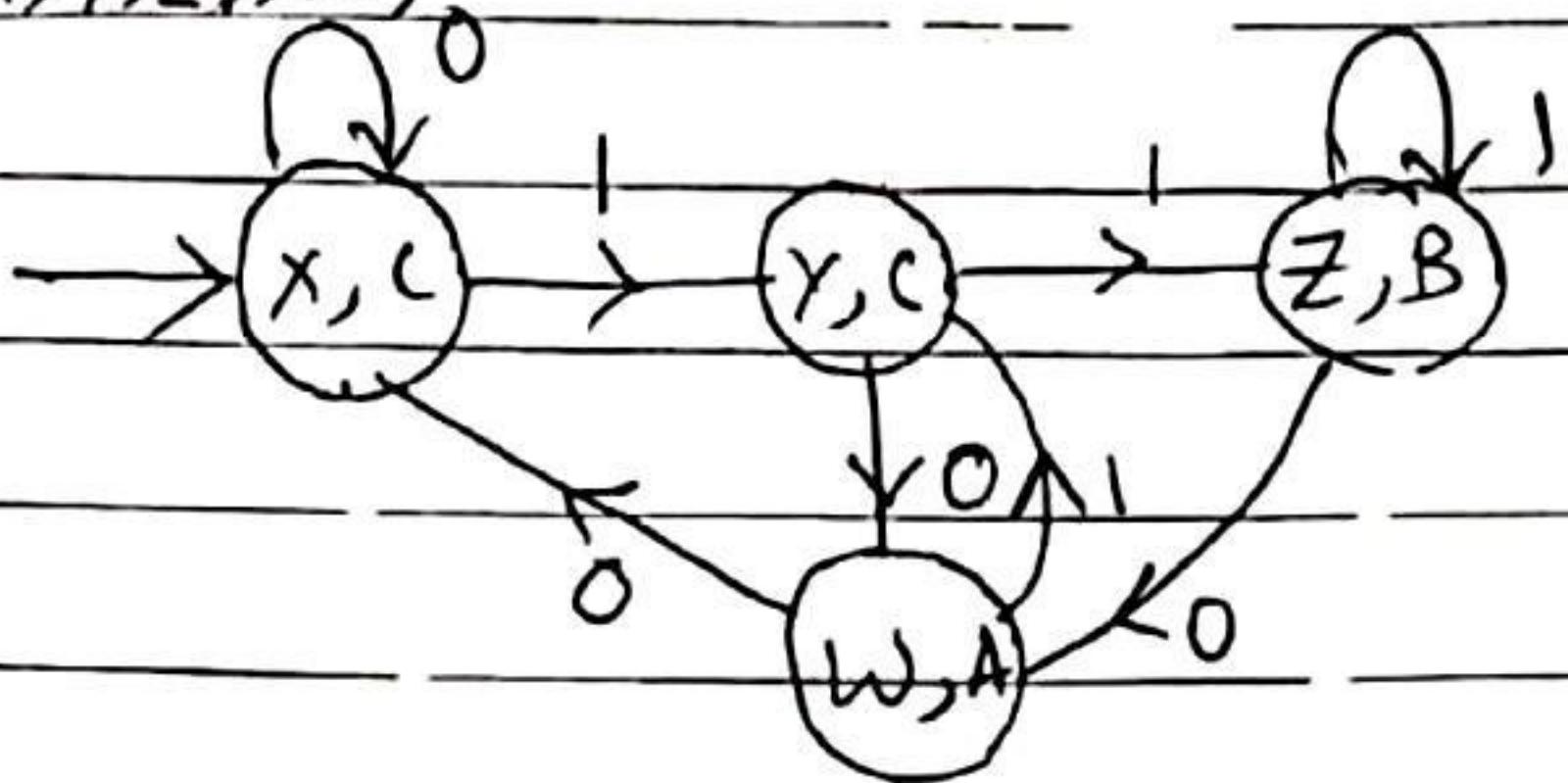
$$10 \rightarrow A$$

$$11 \rightarrow B$$

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

$$\Delta = \{A, B, C\}$$

$$Q = \{x, y, z\}, W$$



Ex-4

construct a moore machine that takes binary no's as i/p and produces residue modulo '3' as o/p.

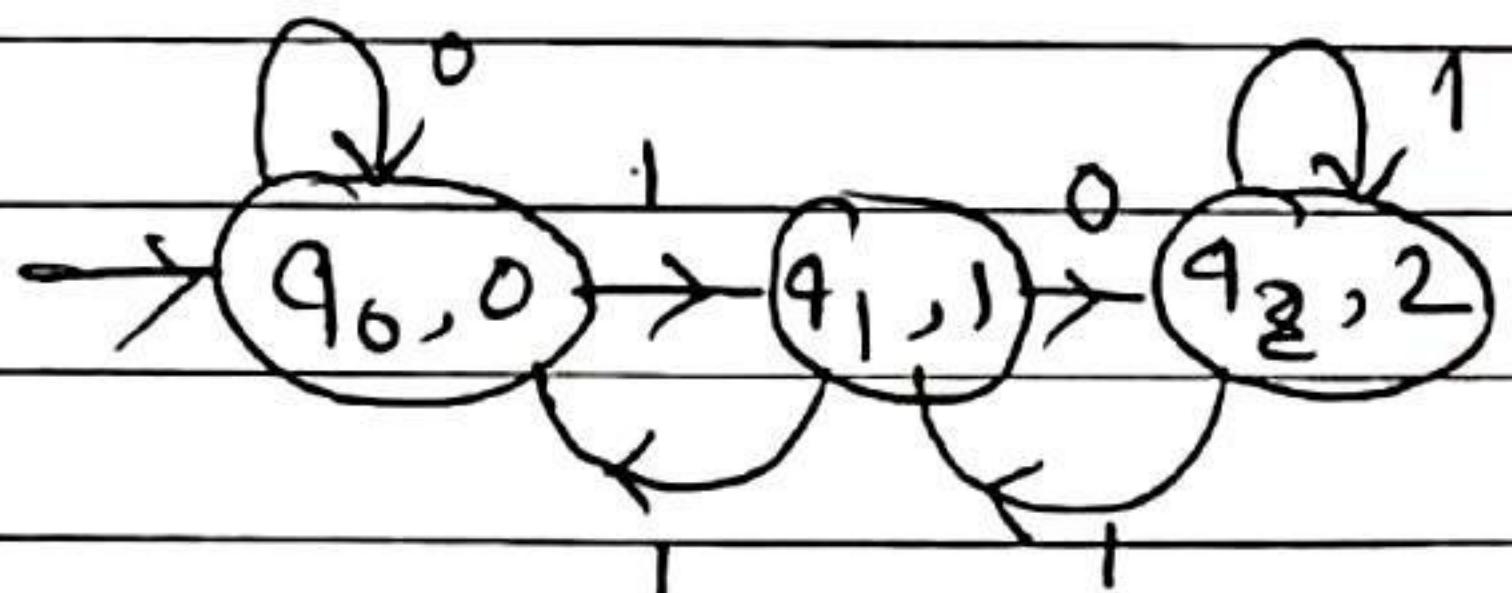
$$\rightarrow \Sigma = \{0, 1\}$$

$$\Delta = \{0, 1, 2\}$$

ST-Table

	0	1	Δ
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

ST-Diagram



construct a moore machine that takes base 4 no's as i/p and produces residue modulo '5' as o/p.

$$\rightarrow \Sigma = \{0, 1, 2, 3\}$$

$$\Delta = \{0, 1, 2, 3, 4\}$$

	0	1	2	3	Δ
q_0	q_0	q_1	q_2	q_3	0
q_1	q_4	q_0	q_1	q_2	1
q_2	q_3	q_4	q_0	q_1	2
q_3	q_2	q_3	q_4	q_0	3
q_4	q_1	q_2	q_3	q_4	4

Example - 5Mealy machine example.

- Construct a mealy m/c that takes binary numbers as input and produces 2's complement of that number as O/P Assume the string is read LSB to MSB and ends carry is discarded.

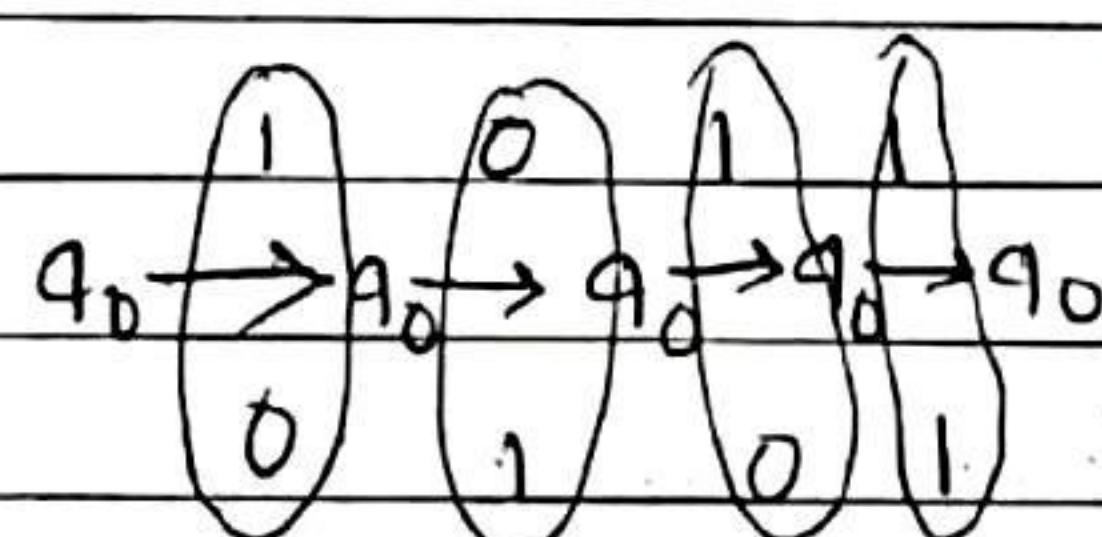
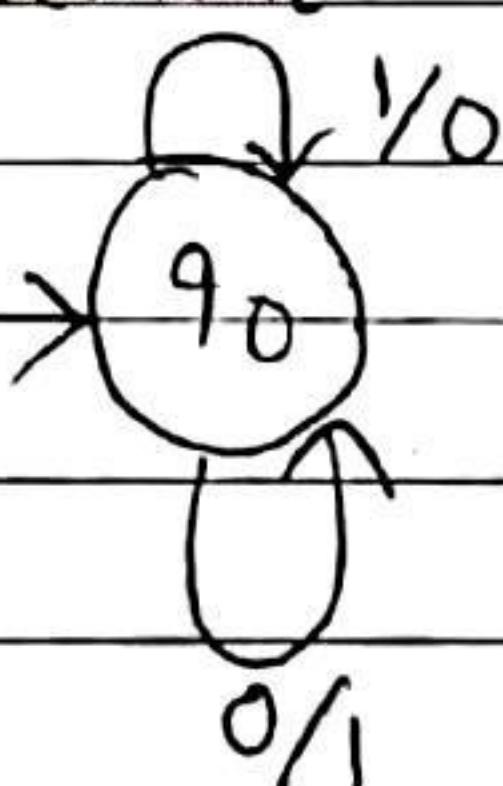
→ For every state, for every input there will be a output.

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

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

$$\begin{array}{r} 0 \ 1 \\ 1 \ 0 \\ + 1 \\ \hline 0 \ 1 \end{array}$$

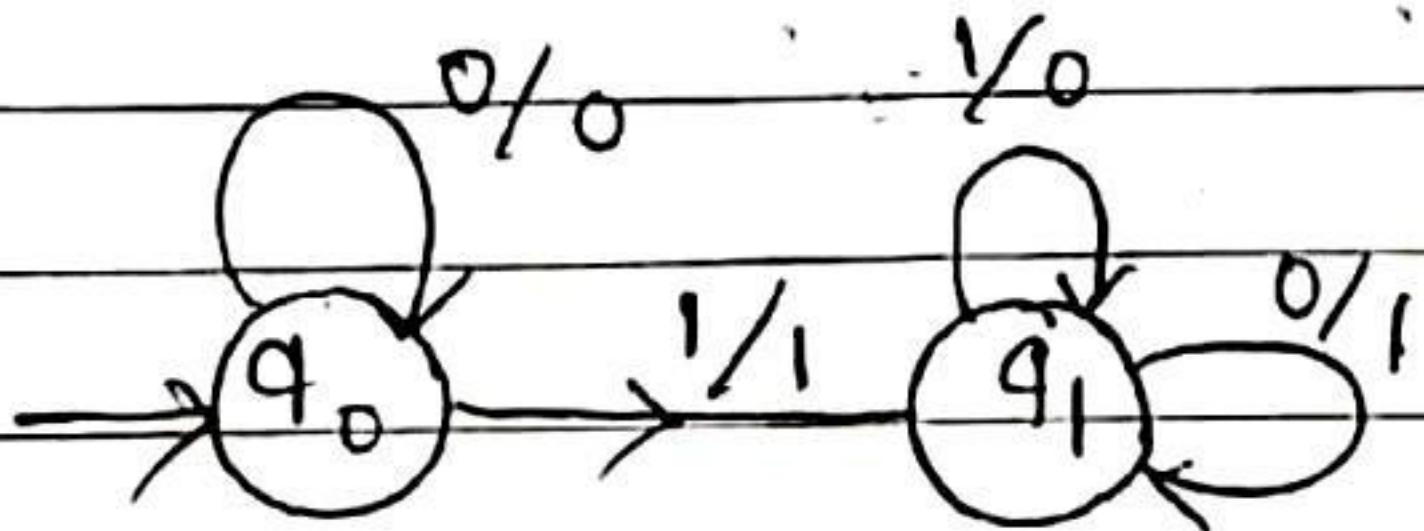
1's complement -



$$\begin{array}{r} 1 \ 1 \ 1 \ 0 \ 1 \boxed{1} \ 0 \ 0 \\ \text{is} - 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \\ + 1 \\ \hline 2^{\text{s}} - 0 \ 0 \ 0 \ 1 \ 0 \boxed{1} \ 0 \ 0 \end{array}$$

$$\begin{array}{r} 1 \ 1 \ 1 \ 1 \ 0 \ 0 \\ \text{is} - 0 \ 0 \ 0 \ 1 \ 1 \\ + 1 \\ \hline 2^{\text{s}} - 0 \ 0 \ 1 \ 1 \boxed{0} \ 0 \end{array}$$

2's complement -

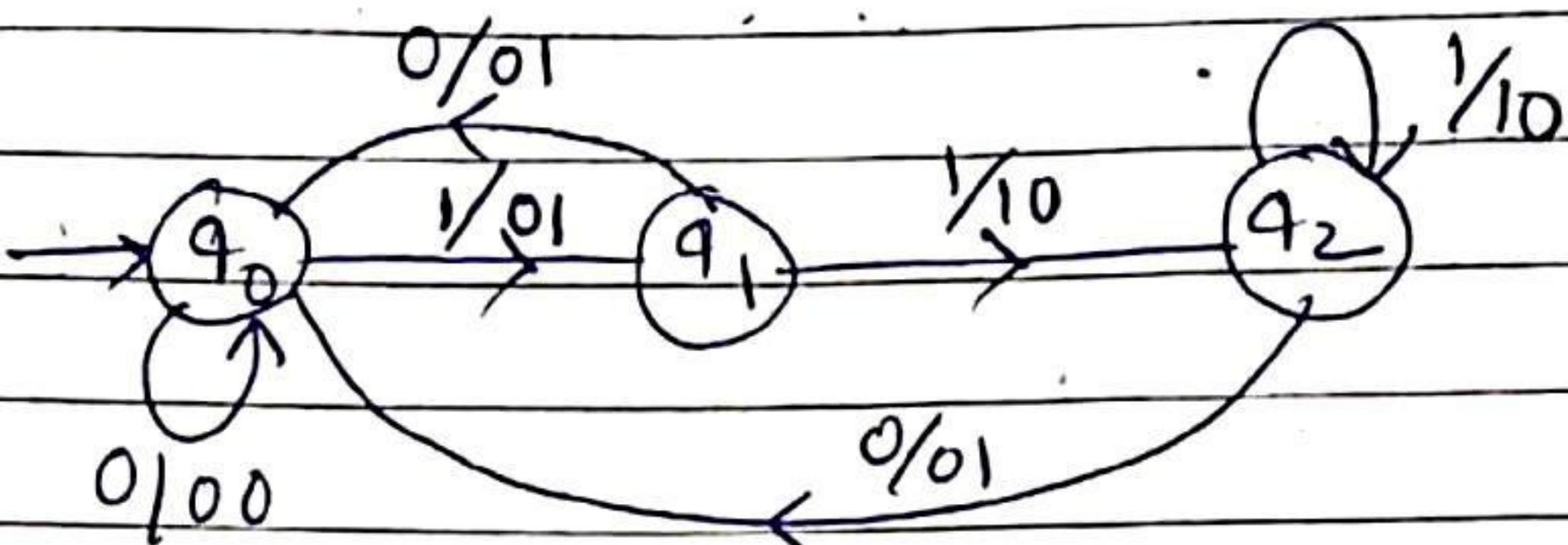


$$\begin{array}{r} 1 \ 0 - 1 \ 1 \ 0 \ 0 \\ 0 \ 0 \ 1 \ 1 \\ + 1 \\ \hline 0 / p - 0 \ 1 \ 0 \ 0 \end{array}$$

$$\begin{array}{r} 0 \ 0 \ 1 \ 1 \\ q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow \\ 0 \ 0 \ 1 \ 0 \end{array}$$

Ex-6

What is the O/p produced by the following state m/c



(a) $11 \rightarrow 01$ (b) $10 \rightarrow 00$

~~(c)~~ sum of present & previous bits.

~~(d)~~ Not.

$$\rightarrow 0 + 1 + 0 + 1 + 1 + 0$$

$$q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_0 \\ 01 \quad 01 \quad 01 \quad 10 \quad 010$$

Conversion of Moore machine to Mealy machine.

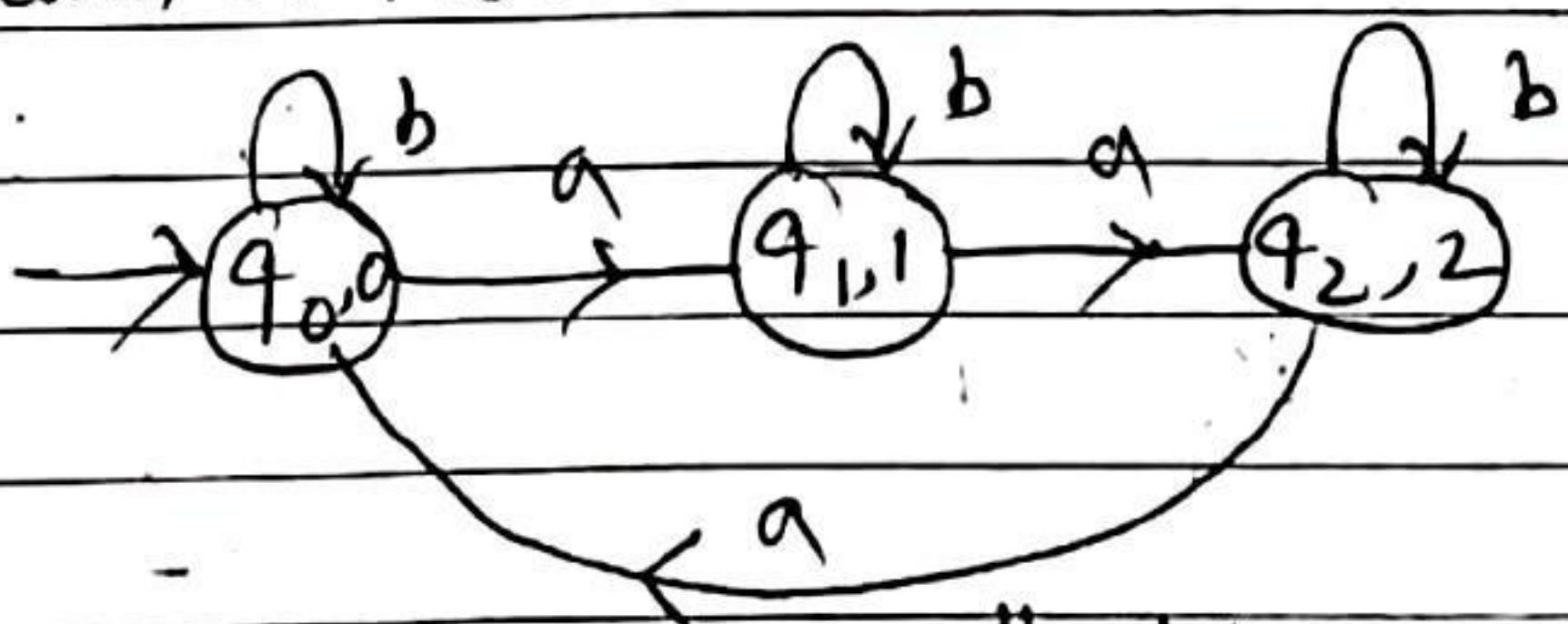
Example-7

→ Moore M/c and Mealy m/c both are same in power.

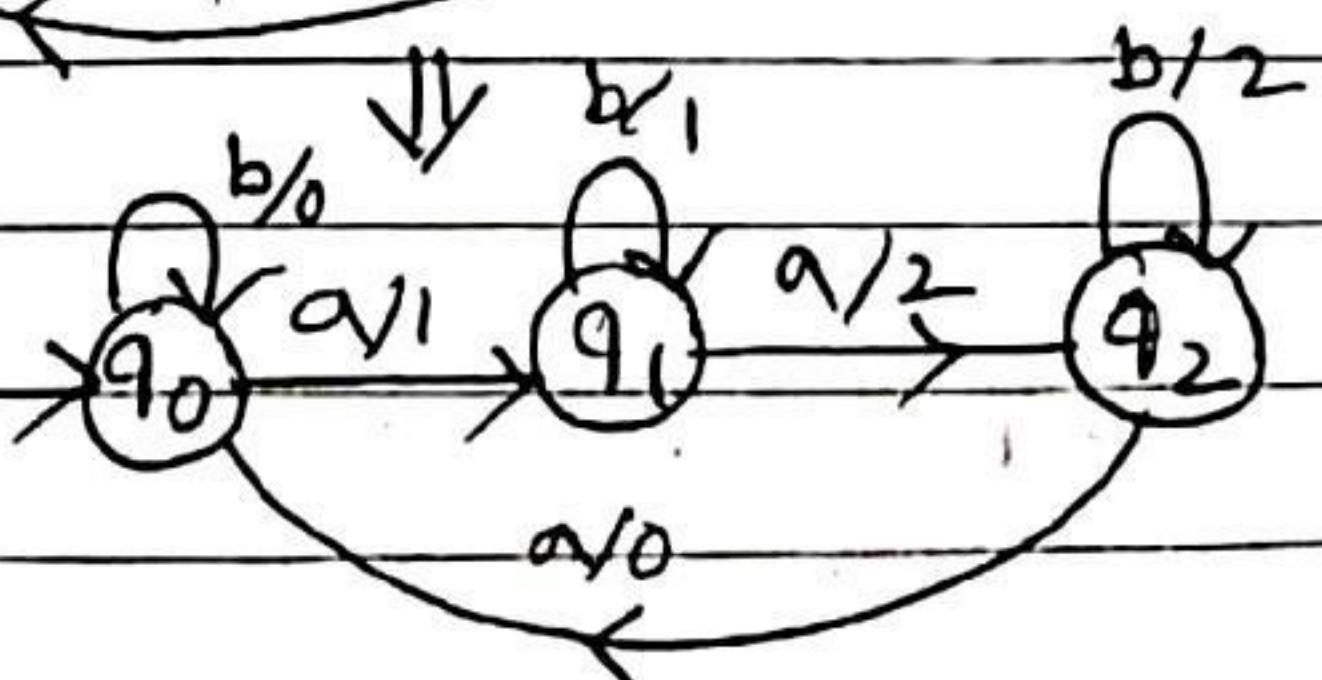
(Moore \rightarrow Mealy)

Count no of a's $\% 3$.

ST-Diagram of Moore -



ST-Diagram of Mealy -



ST-Table of Moore -

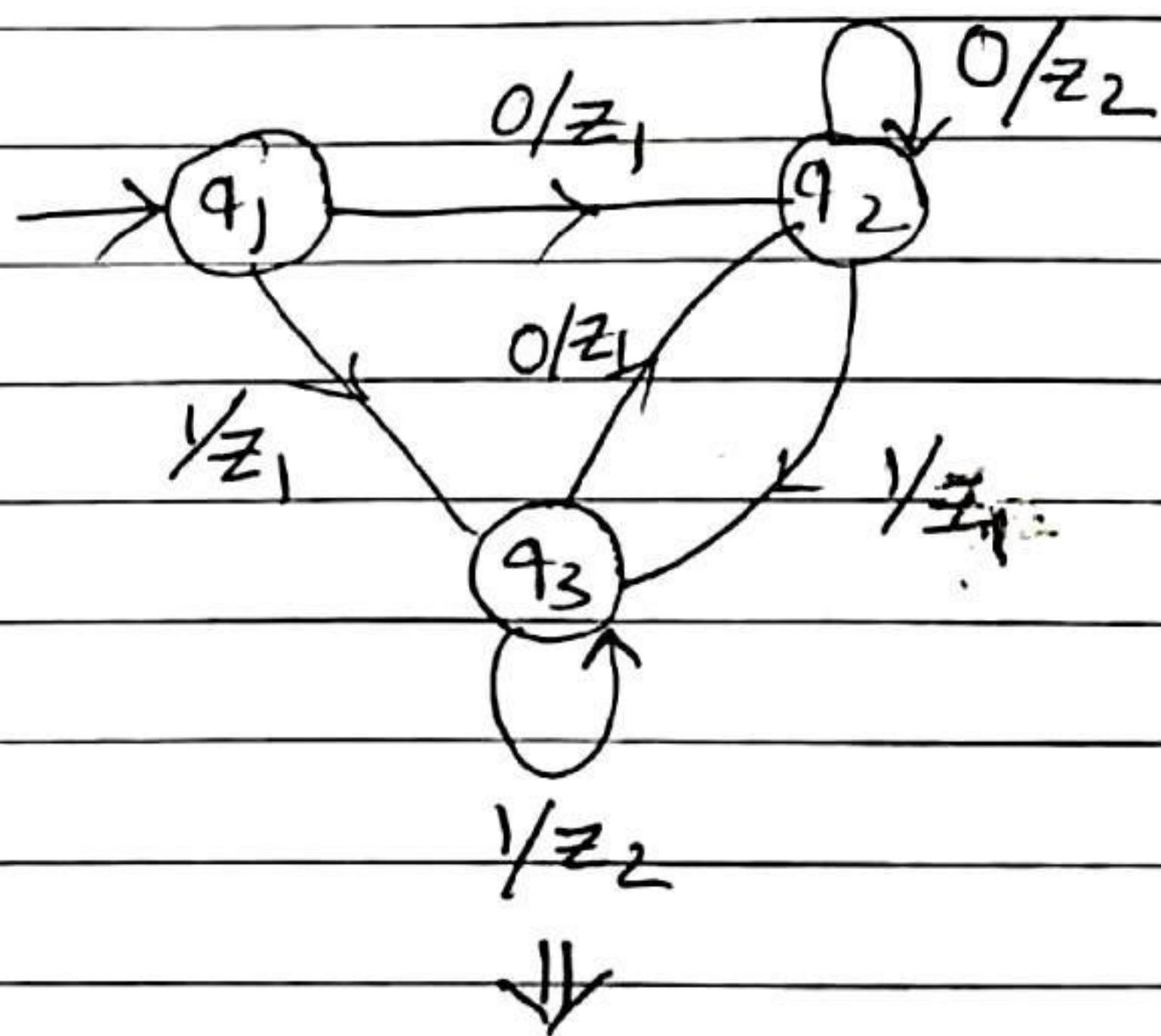
	a	b	Δ	
q_0	q_1	q_0	0	\Rightarrow
q_1	q_2	q_1	1	
q_2	q_0	q_2	2	

ST-Table of Mealy -

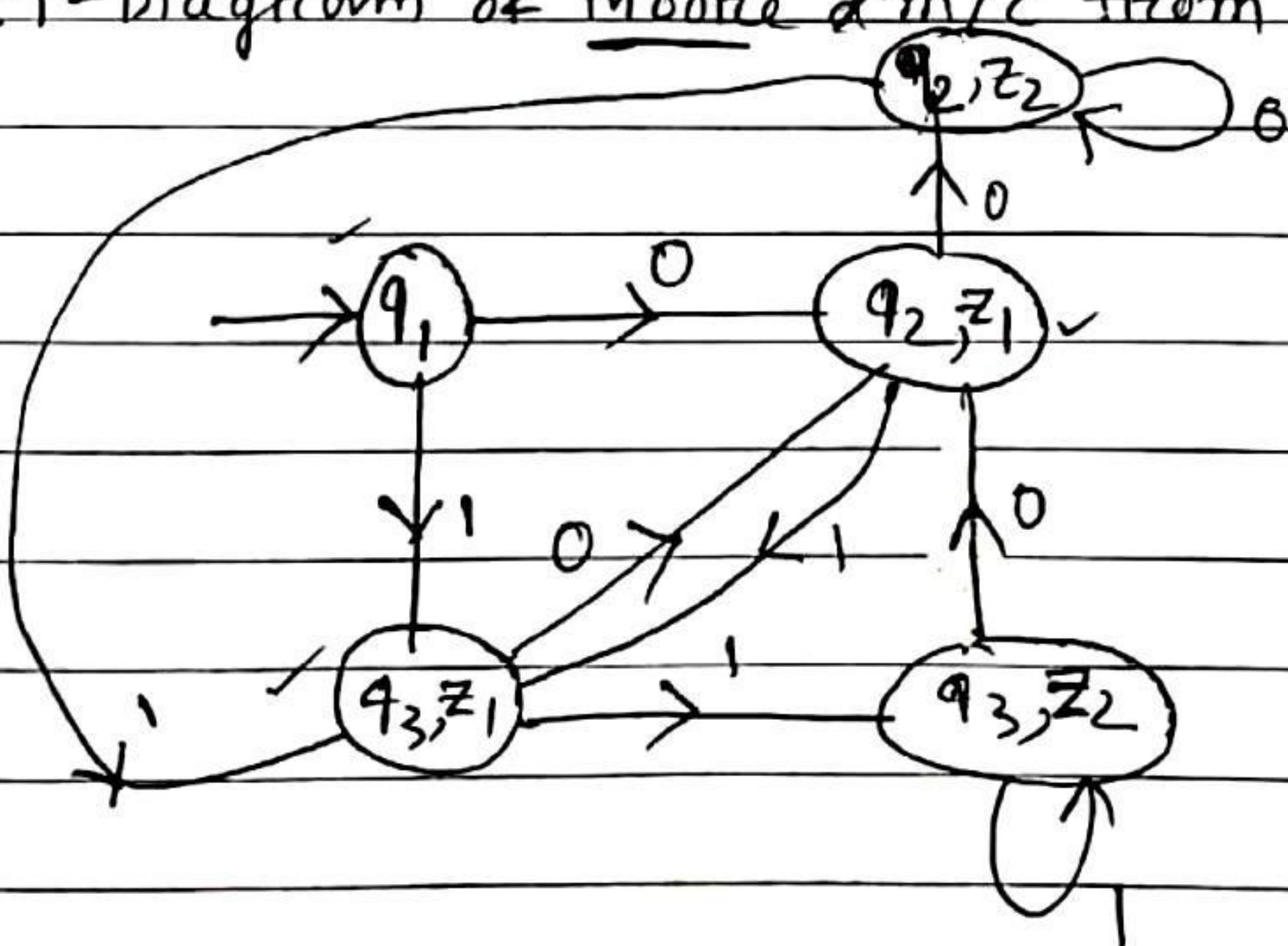
	a	b
q_0	$(q_1, 1)$	$(q_0, 0)$
q_1	$(q_2, 2)$	$(q_1, 1)$
q_2	$(q_0, 0)$	$(q_2, 2)$

[Ex-8] conversion ~~Moore to Mealy~~ to Mealy to Moore Machine.

ST-Diagram of Mealy m/c -

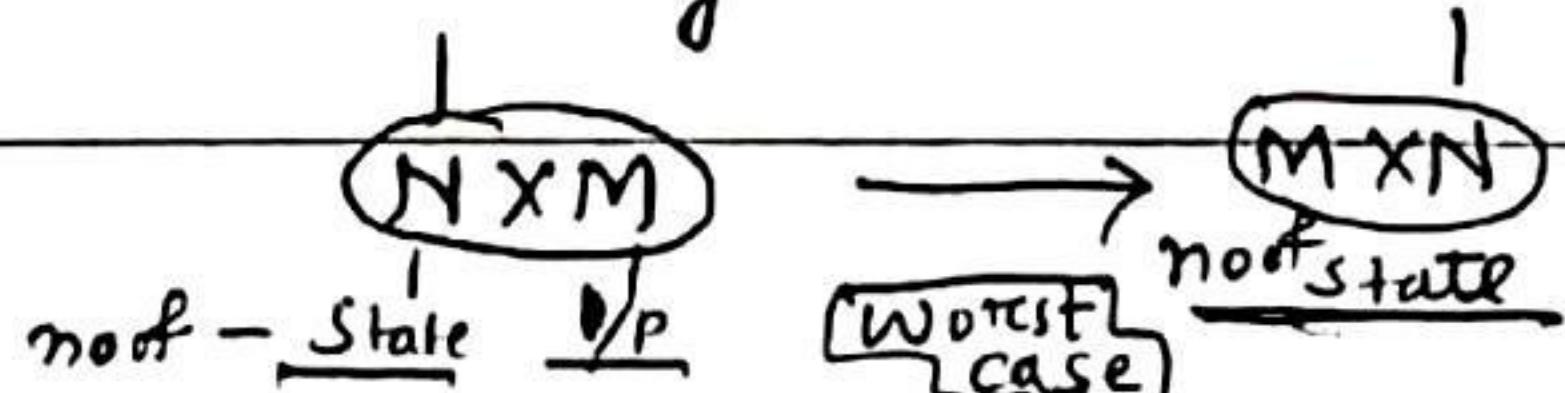


ST-Diagram of Moore m/c from STD of Mealy machine -

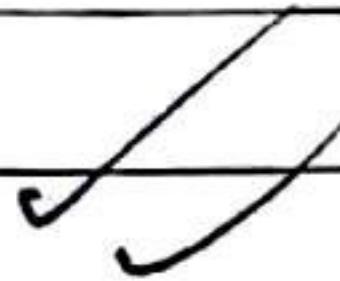
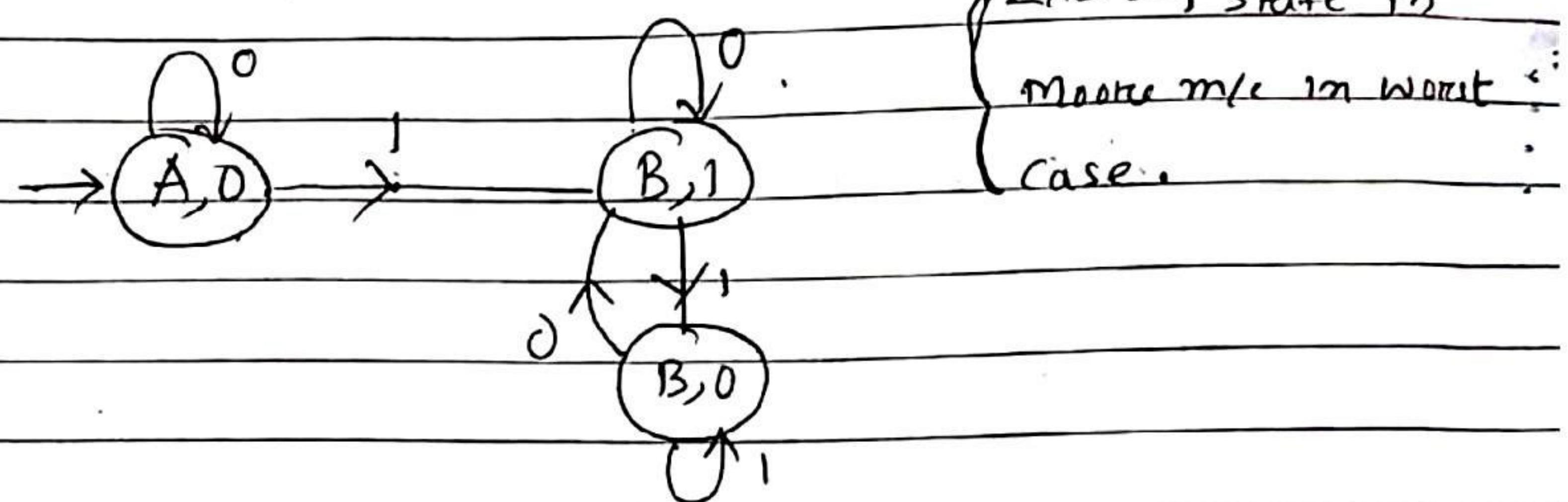
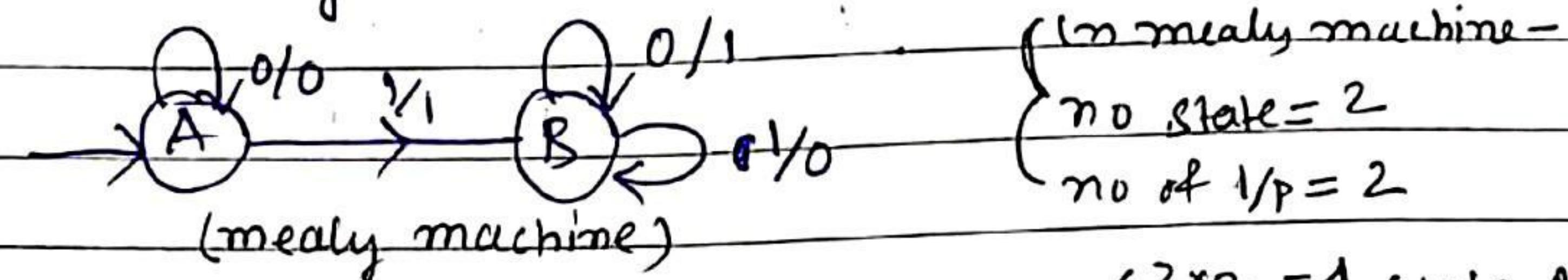


* \Leftrightarrow Moore m/c \rightarrow mealy m/c. (No of state same)

* Mealy m/c \rightarrow moore m/c. (no of state may - increase)

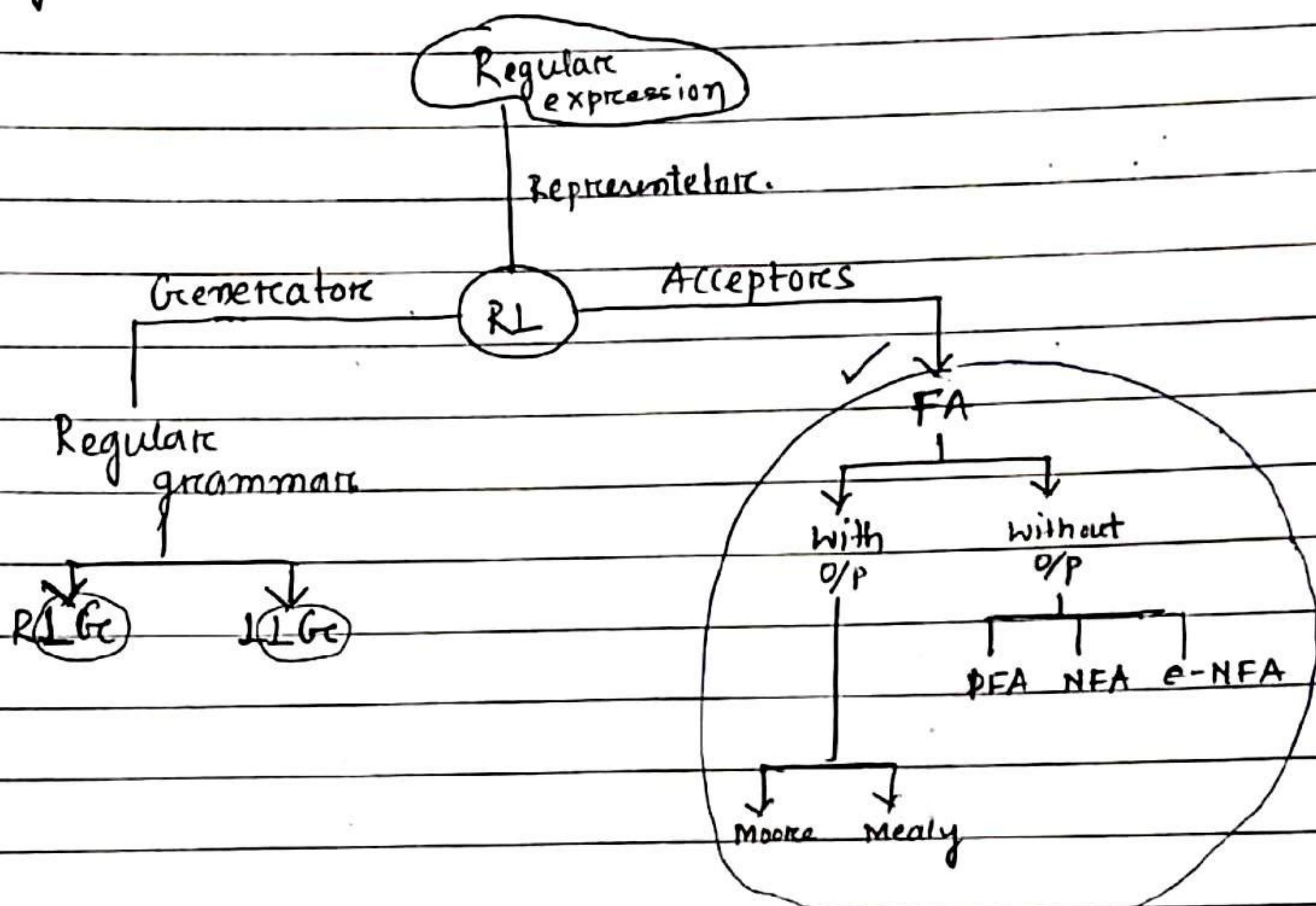


[Ex-9] Mealy m/c \rightarrow Moore m/c



F.

• Regular Expression



→ Regular expression mean to represent certain set of strings
In some algebraic fashion.

→ Regular expression contain 3- operations -

- (I) + (union)
- (II) • (concatination)
- (III) * (Kleene closure)

∴ A RE over the alphabet Σ is defined as follows -

(a) $\emptyset, \epsilon, a \in \Sigma$ (primitive)

$$\downarrow \quad \downarrow \quad \begin{matrix} a \\ 1 \end{matrix} \quad \begin{matrix} b \\ 1 \end{matrix}$$

$$\{\} \quad \{\epsilon\} \quad \{a\} \quad \{b\}.$$

(b) $r_1 + r_2, r_1 \cdot r_2, r_1^*$.

→ A set represented by a regular expression is called Regular set.

Different of regular sets and their Expression -

<u>Regular expression</u>	<u>Regular Set</u>
\emptyset	$\{\}$
ϵ	$\{\epsilon\}$
a	$\{a\}$
a^*	$\{\epsilon, a, aa, aaa, \dots\}$
a^+	$\{a, aa, aaa, aaaa, \dots\}$
$(a+b)^*$	$\{\epsilon, a, b, aa, ab, ba, bb, \dots\}$
$(a+b)^+$	$\{a, b\}$

Ex-1

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

① Find out regular expression for set of all strings whose length is exactly '2'.

$$\rightarrow L = \{aa, ab, ba, bb\}$$

$$= a(a+b) + b(a+b)$$

$$= a\underbrace{(a+b)}_{\text{length exactly '2'}} + b\underbrace{(a+b)}_{\text{length exactly '2'}}$$

$$= \underbrace{(a+b)(a+b)}$$

length exactly '2'

② Find out RE for set of all strings whose length is exactly '3'.

$$\rightarrow L = \{aaa, aab, aba, bba, aab, abb, bab, bbb\}$$

$$= \underbrace{(a+b)(a+b)(a+b)}$$

length exactly '3'

③ Find out RE for set of all strings whose length is at least '2'.

$$\rightarrow L = \{ aa, ab, ba, bb, aaa, \dots \}$$

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

④ Find out RE for set of all strings whose length is at most '2'.

$$\rightarrow L = \{ \underbrace{\epsilon}_{0}, \underbrace{a}_{1}, \underbrace{b}_{2}, aa, ab, ba, bb \}$$

$$= (\epsilon + a + b + aa + ab + ba + bb)$$

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

Ex-2

$$\textcircled{1} \quad \Sigma = \{a, b\}$$

Find out Regular expression for set of all strings of even length.

$$\rightarrow L = \{ \text{set of all strings of even length} \}$$

$$\begin{aligned} & ((a+b)^2)^* \\ & = (a+b)^{2n} / n \geq 0 \end{aligned}$$

$$= \{ \epsilon, aa, ab, ba, bb, \dots \}$$

$$= \boxed{(a+b)(a+b)^*}$$

② Find out Regular expression for set of all strings of odd length.

$$\rightarrow L = \{ a, b, aaa, \dots \}$$

$$(a+b)^{2n+1} / n \geq 0$$

$$= \boxed{(a+b)(a+b)^* (a+b)}$$

[Ex-3]

① Find out RE for set of all string whose length is divisible by '3'.

$\rightarrow L = \{0, 3, 6, 9, 12, \dots\}$ length of the string divisible by 3.

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

② Find out RE for set of string whose length $\equiv 2 \pmod{3}$.

$$\rightarrow (a+b)^{\textcircled{3n+2}} / n \geq 0$$

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

[Ex-4]

① Find out RE for set of all strings in which no of 'a's are exactly exactly '2'.

$$b^n a b^n a b^n / n \geq 0$$

$$\rightarrow b^* a b^* a b^*$$

$$L = \{aa, babab, bbabb, abb, \dots\}$$

② no of a's atleast '2'.

$$\rightarrow b^* a^2 b^* a (a+b)^*$$

③ no of a's atmost '2'.

$$\rightarrow b^* (\epsilon + a) b^* (\epsilon + a) b^*.$$

④ a's are even.

$$\rightarrow (\cancel{a^2})^* (b^* a b^* a b^*)^* + b^*$$

[Ex-5]

RE for

① Find out set of all string which starts with 'a' .

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

② set of all string which end's with 'a' .

$$\rightarrow (a+b)^* a.$$

③ set of all string containing 'a' .

$$\rightarrow (a+b)^* a (a+b)^*.$$

④ set of all string starting and ending with diff symbols.

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

+

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

⑤ starting and ending with same symbol.

$$\rightarrow \text{set } L = \{ \epsilon, a, b, aa, bb, aaa, bbb, \dots \}$$

Regular expression, = $a(a+b)^* + b(a+b)^* b + a + b + \epsilon$.**[Ex-6]**① Find out Regular language for set of all string doesn't contain ~~any~~^{expression} two a's together.

$$\rightarrow L = \{ \epsilon, b, bb, bbb, aba, a, abab, bba, ba \dots \}$$

$$\begin{aligned} RE &= (b+ab)^* + (b+ab)^* a \\ &= (b+ab)^* (a+\epsilon) \end{aligned}$$

② Find out a RE for all set of all string: \$ doesn't contain no 2 a's and 2 b's together.

$$\rightarrow L = \{ \epsilon, a, b, ab, ba, aba, bab, \dots \}$$

RE (Regular expression) :-

RE	(Starts with)	(ends with)	Language
<u>$(ab)^*a/a(ab)^*$</u>	<u>a</u>	<u>a</u>	$\{a, aba, bab, abab\}$
<u>$(ba)^*b/b(ba)^*$</u>	<u>b</u>	<u>b</u>	$\{b, bab, babab\}$
<u>$a(ab)^*/(ab)^*$</u>	<u>a</u>	<u>b</u>	$\{ab, bab\}$
<u>$b(ab)^*/(ba)^*$</u>	<u>b</u>	<u>a</u>	$\{ba, bab\}$

$$RL = \underline{(ab)^*a} + b(ab)^* + \underline{(ab)^*} + b(ab)^*\underline{a}.$$

$$= (ab)^*(a+\epsilon) + b(ab)^*(a+\epsilon)$$

$$= (a+\epsilon)(\epsilon+b)(ab)^*(a+\epsilon).$$

$$= (\epsilon+a)(ba)^*(b+\epsilon)$$

• IDENTITIES OF RE :-

$$\rightarrow \emptyset + R = R + \emptyset = \emptyset R$$

$$\rightarrow RR^* = R^+$$

$$\rightarrow \emptyset \cdot R = R \cdot \emptyset = \emptyset$$

$$\rightarrow R^* R^* = R^*$$

$$\rightarrow \epsilon \cdot R = R \cdot \epsilon = R$$

$$\rightarrow \epsilon^* = \epsilon$$

$$\rightarrow [\emptyset^* = \epsilon]$$

$$\rightarrow \epsilon + RR^* = R^*R + \epsilon = R^*$$

$$\rightarrow (a+b)^* = (a^* + b^*)^*$$

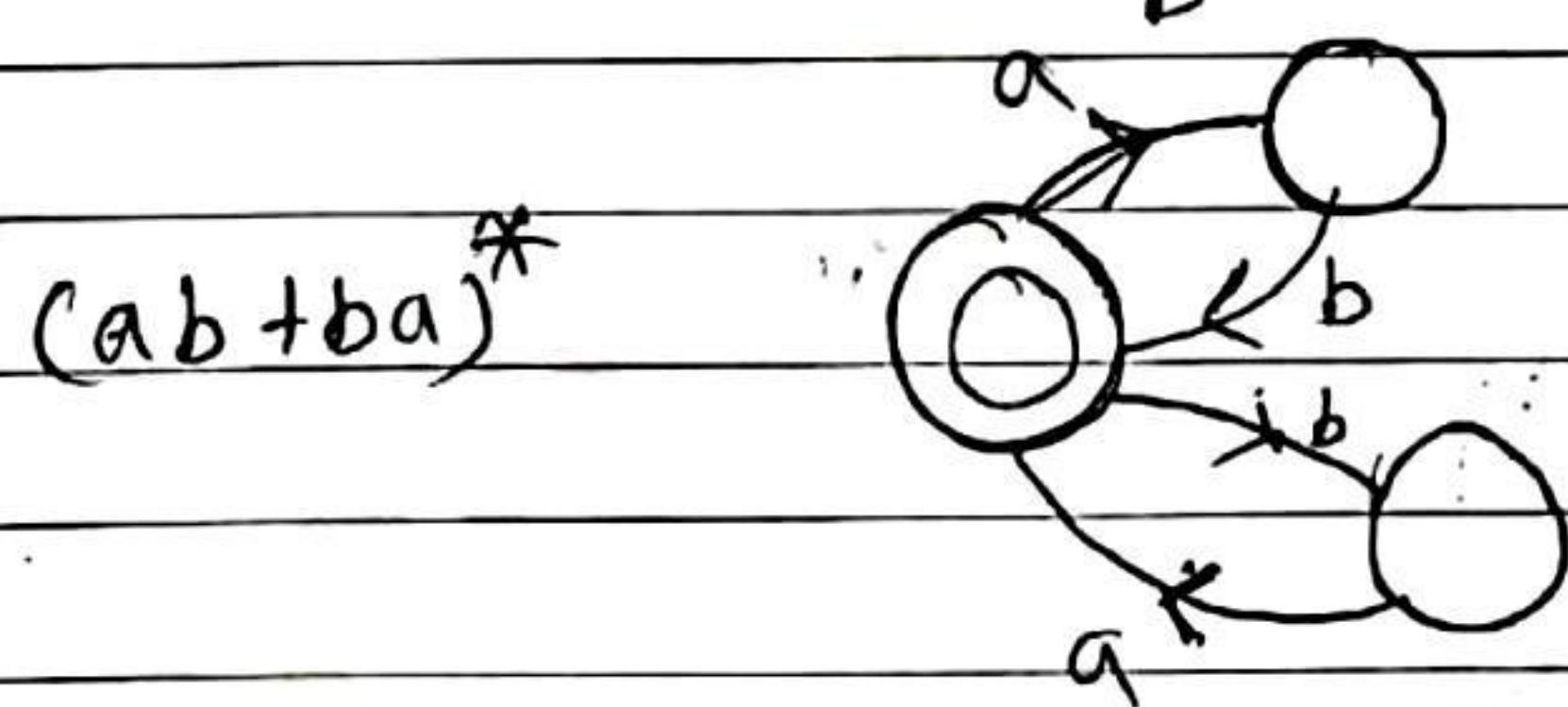
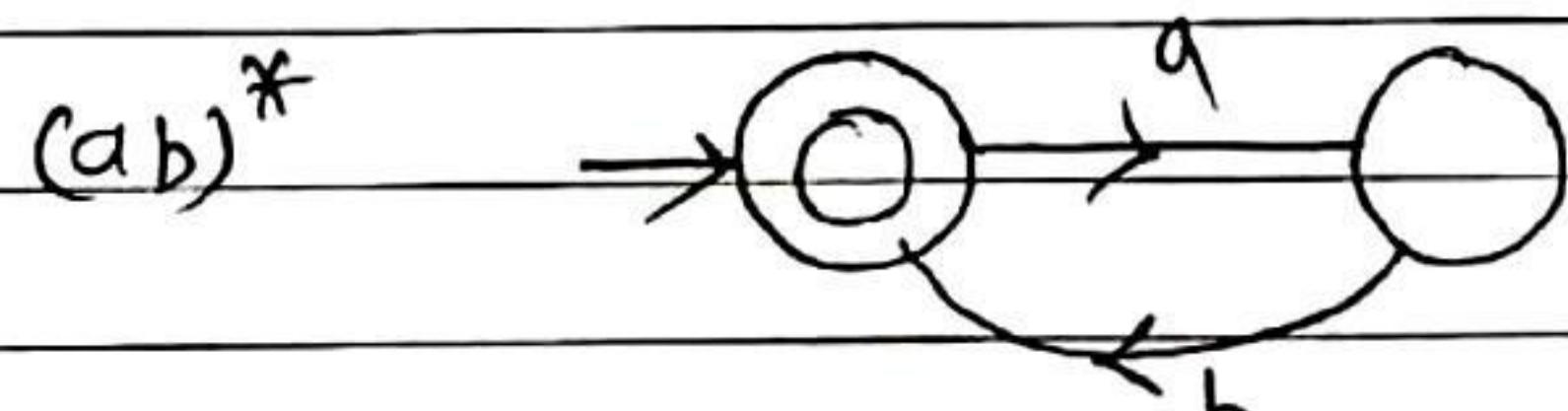
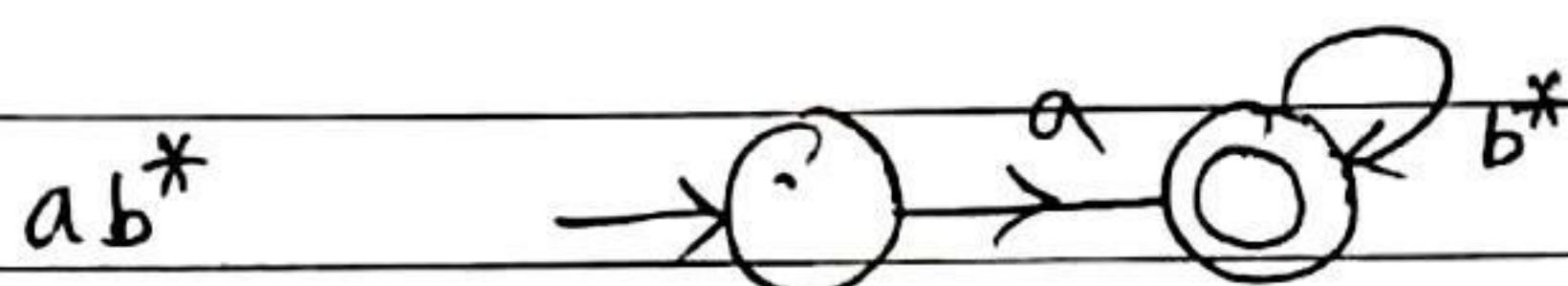
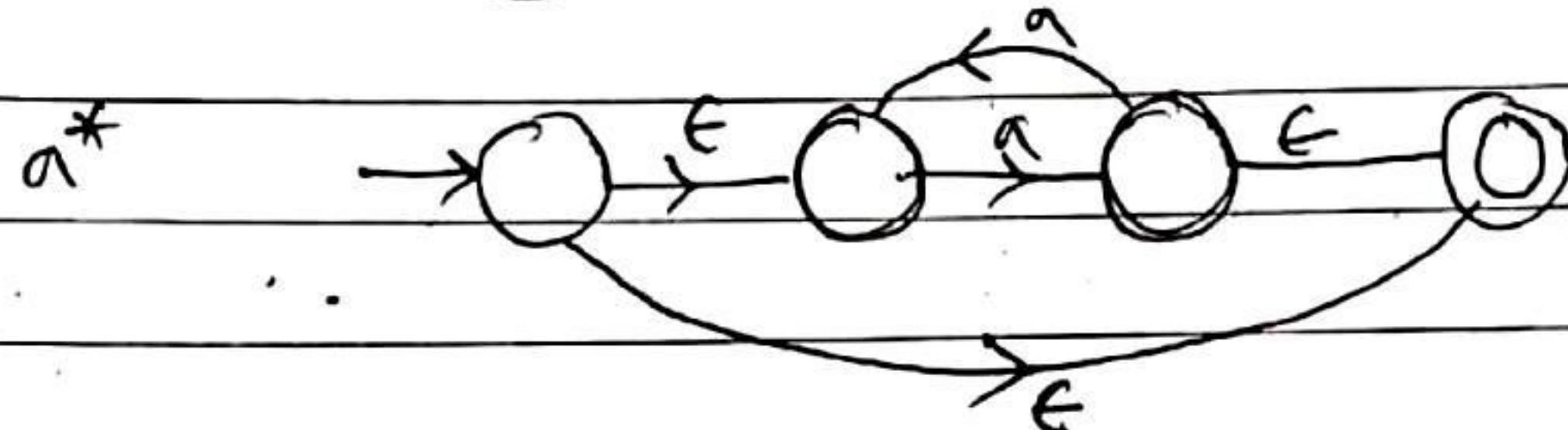
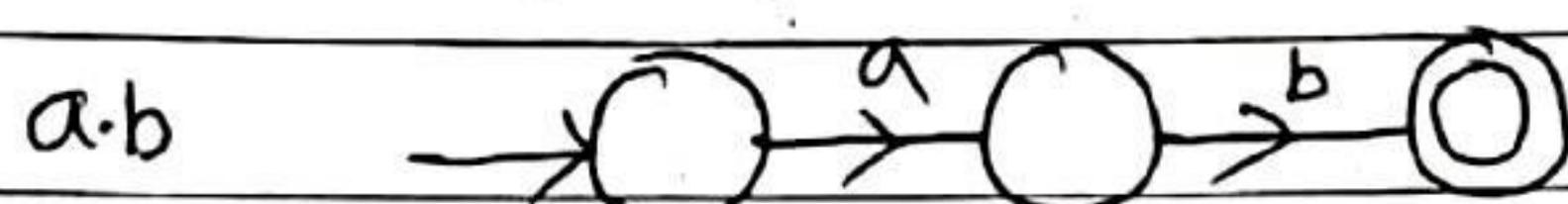
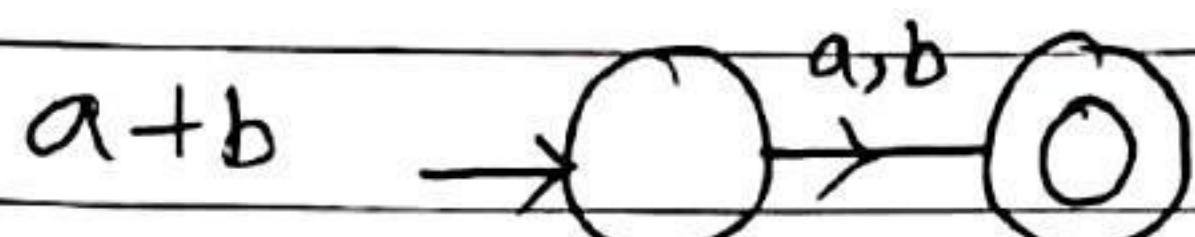
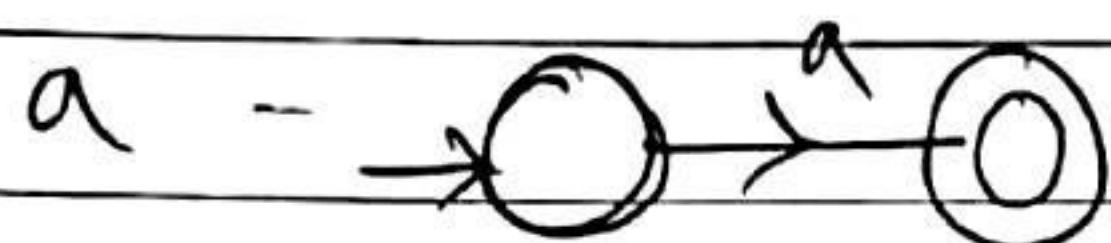
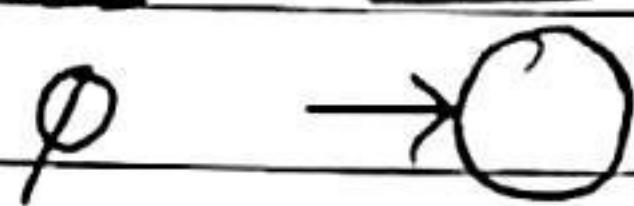
$$= (a^* b^*)^*$$

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

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

• Conversion RE \rightarrow FA :

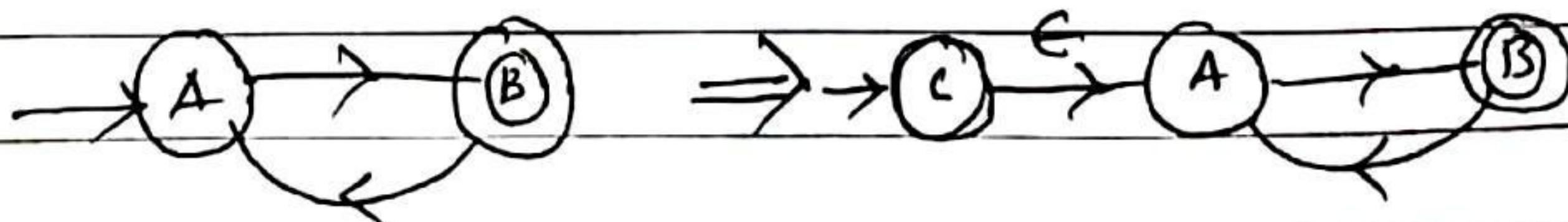
$\boxed{RE} \rightarrow \boxed{FA}$



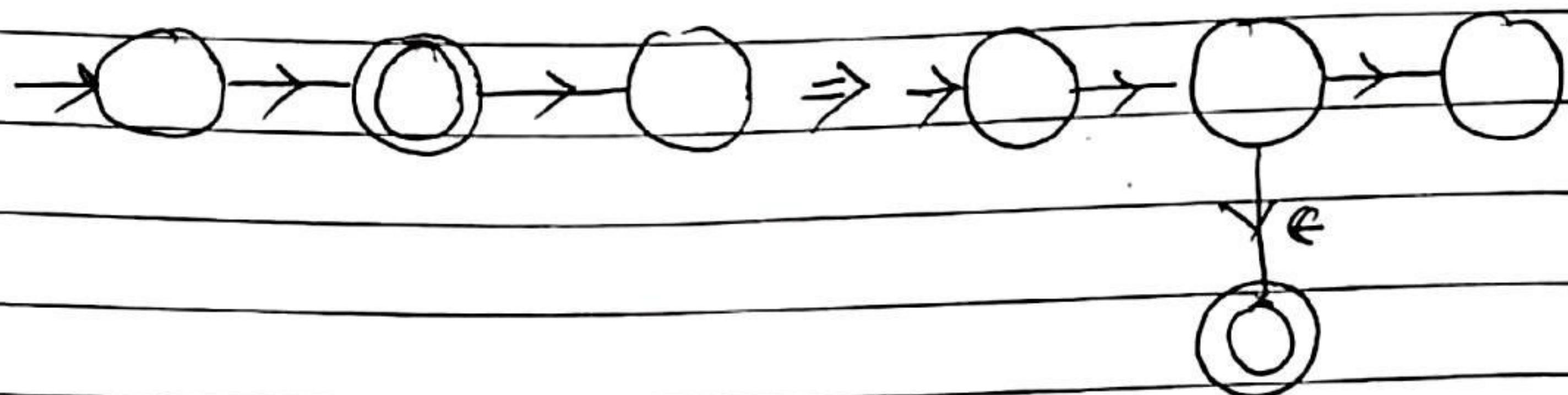
• Conversion FA \rightarrow RE :

• State Elimination Method:- (3 rules)

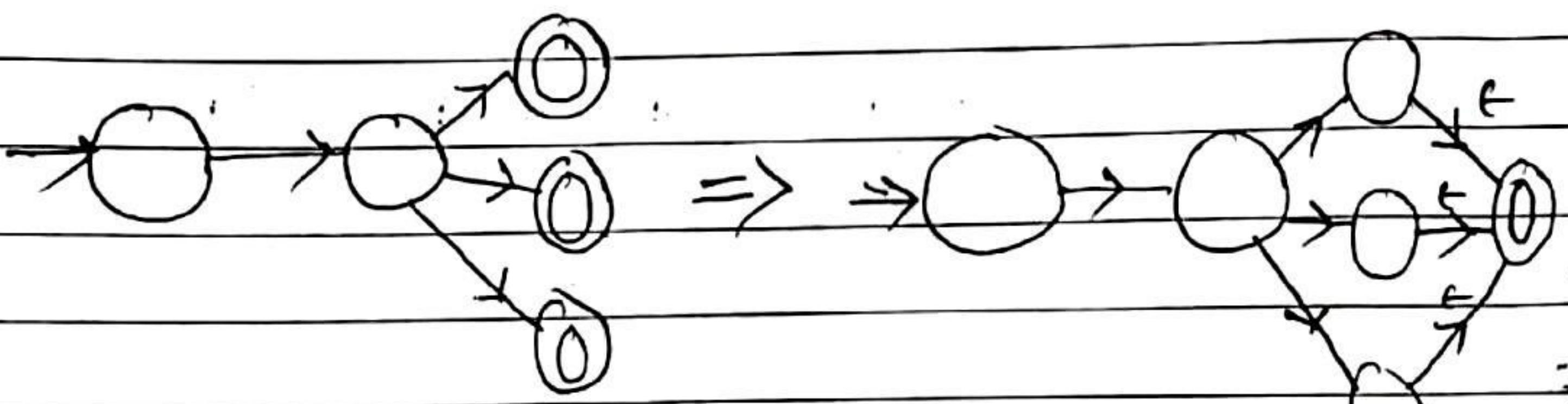
(1) Initial state should not have any incoming edge.



(2) - The final state should not have any outgoing edges.



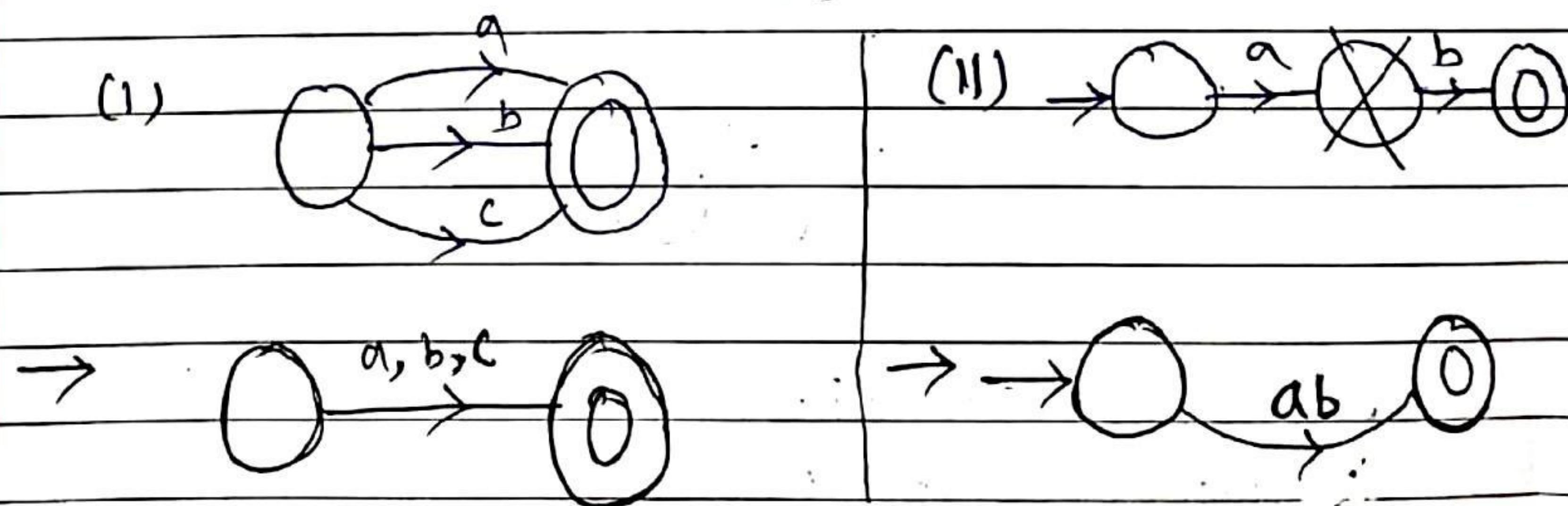
- If have more than one final state then make a new final state then remove previous final states.



(3) → other than initial and final state eliminate ~~other~~ remaining states one by one.

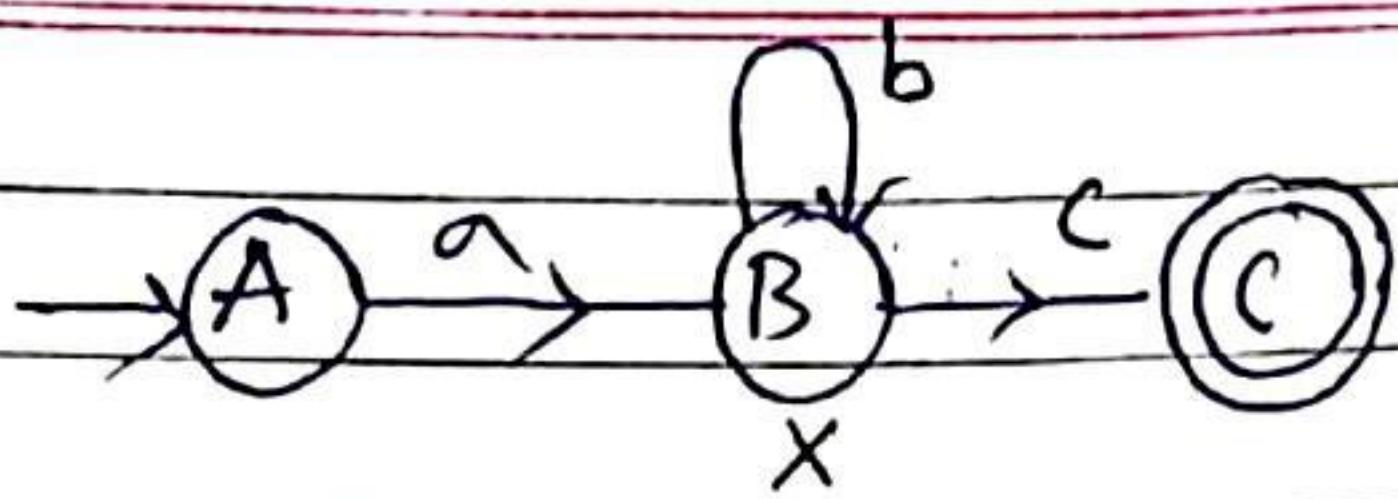
[Ex-1]

Find RE from FA:



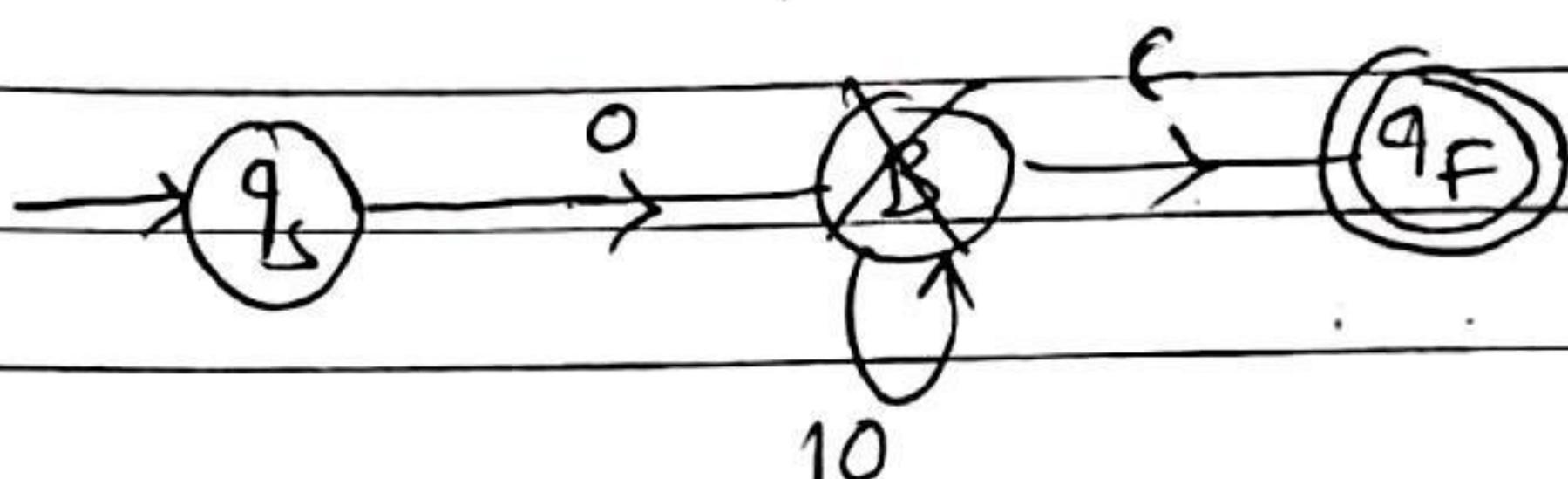
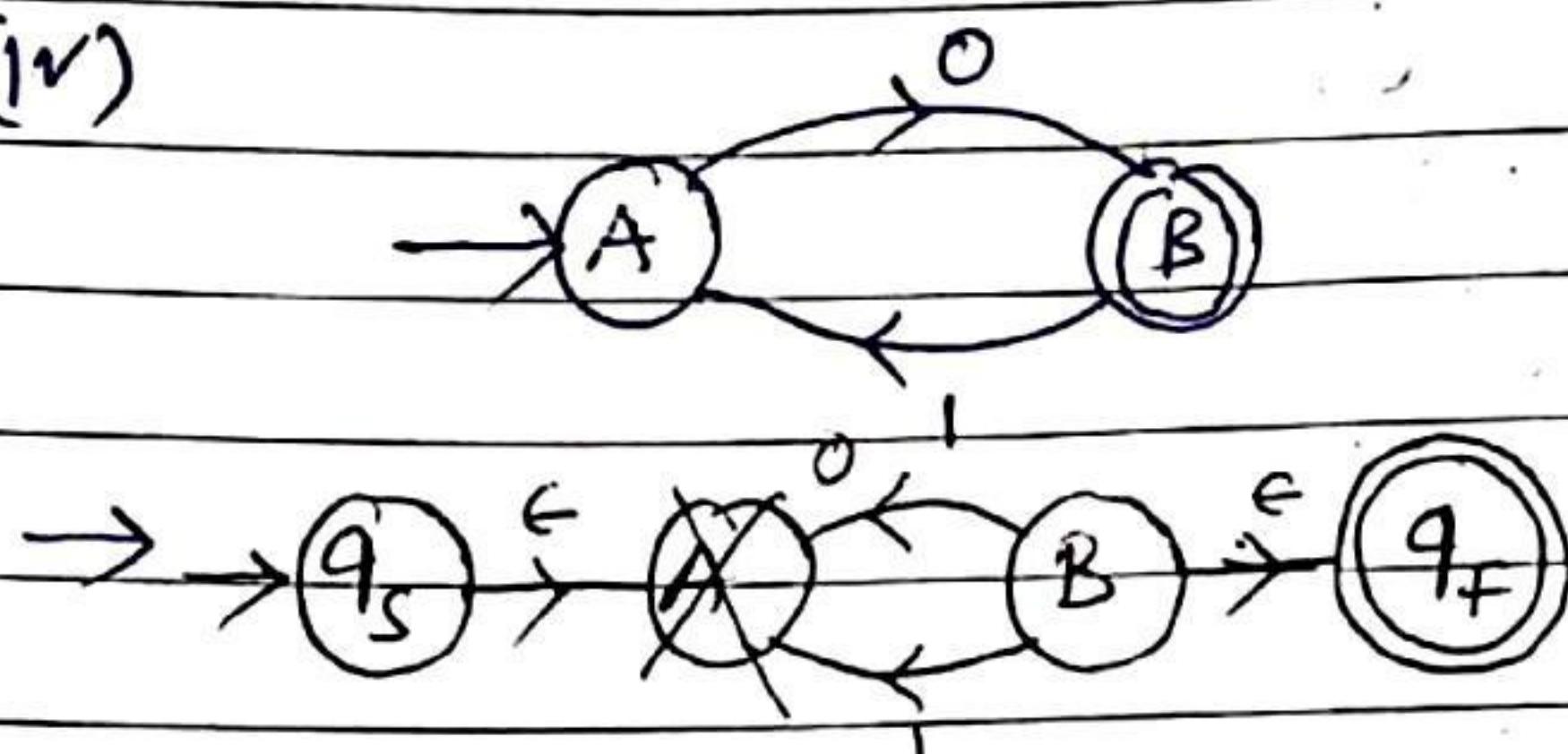
Regular Expression = $(a+b+c)$

(11)



So, RE is = $a b^* c$.

(1✓)

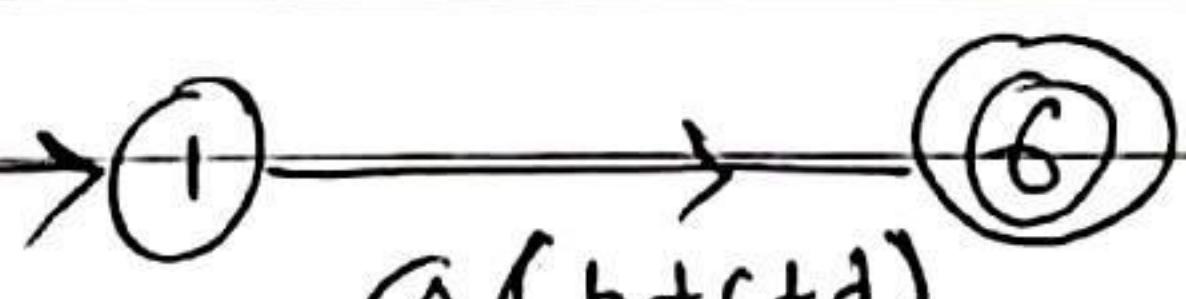
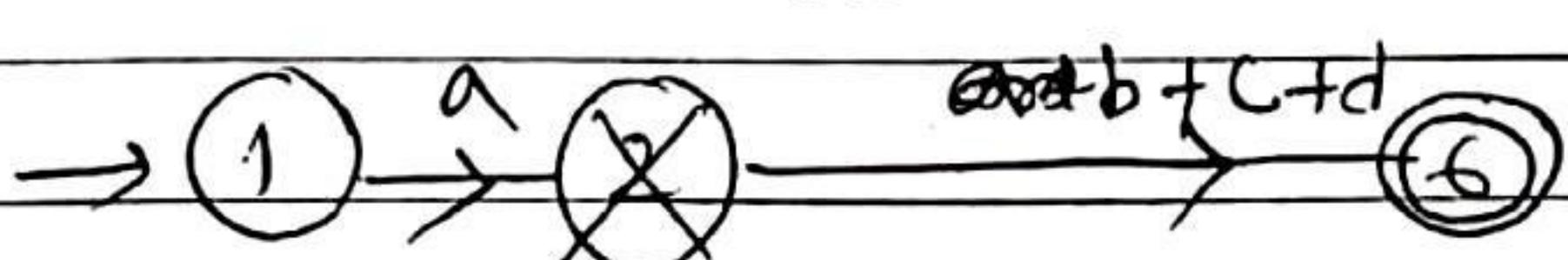
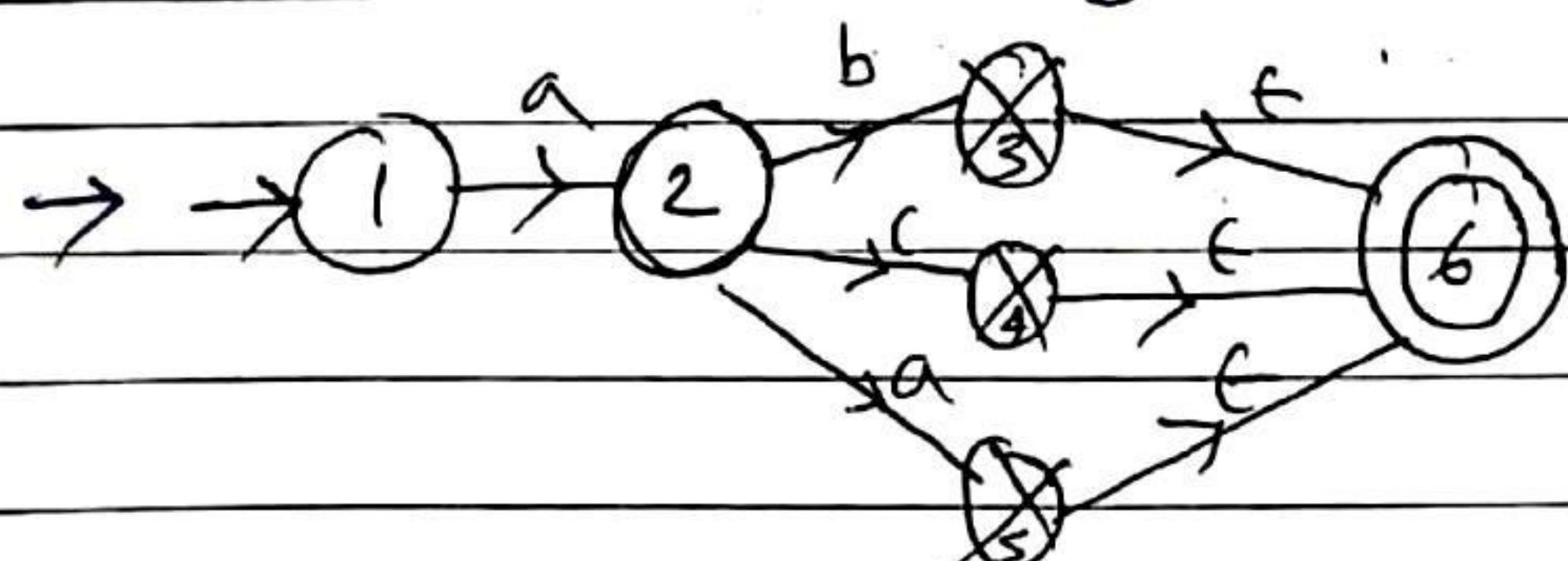
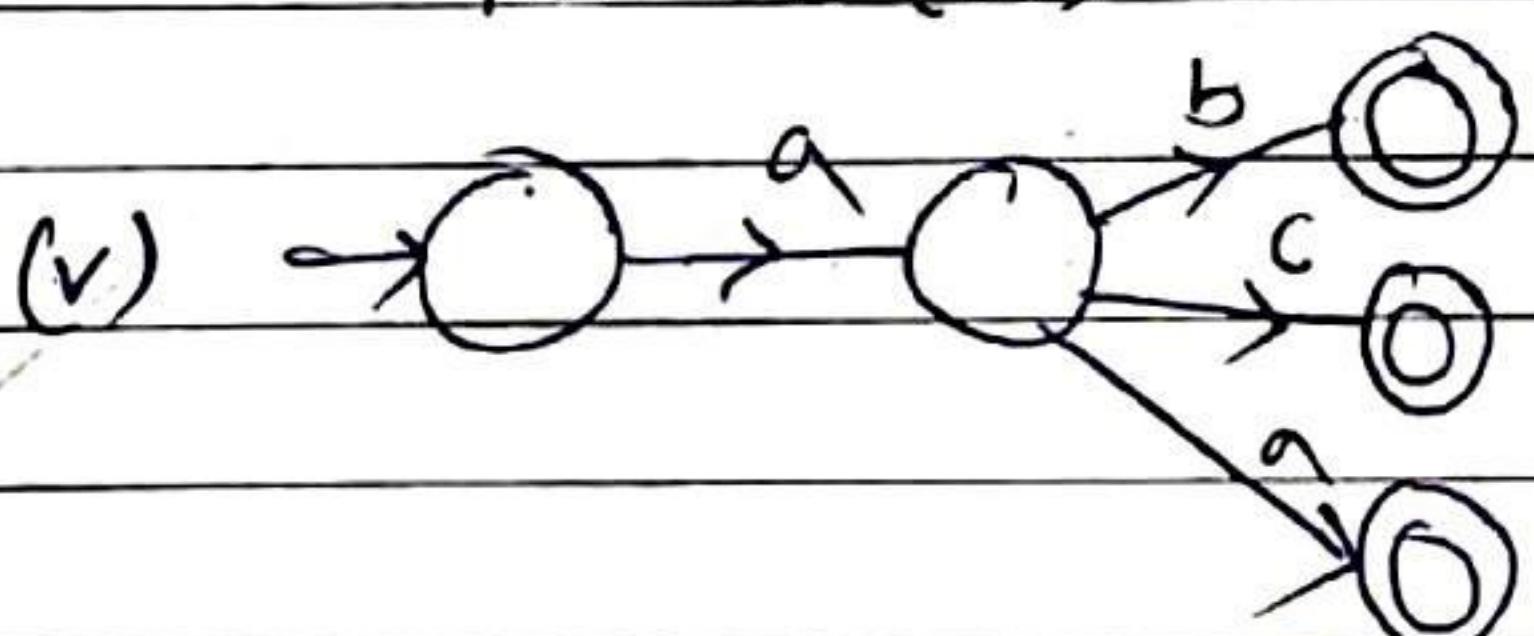


q_s q_F

$O(10)^*$

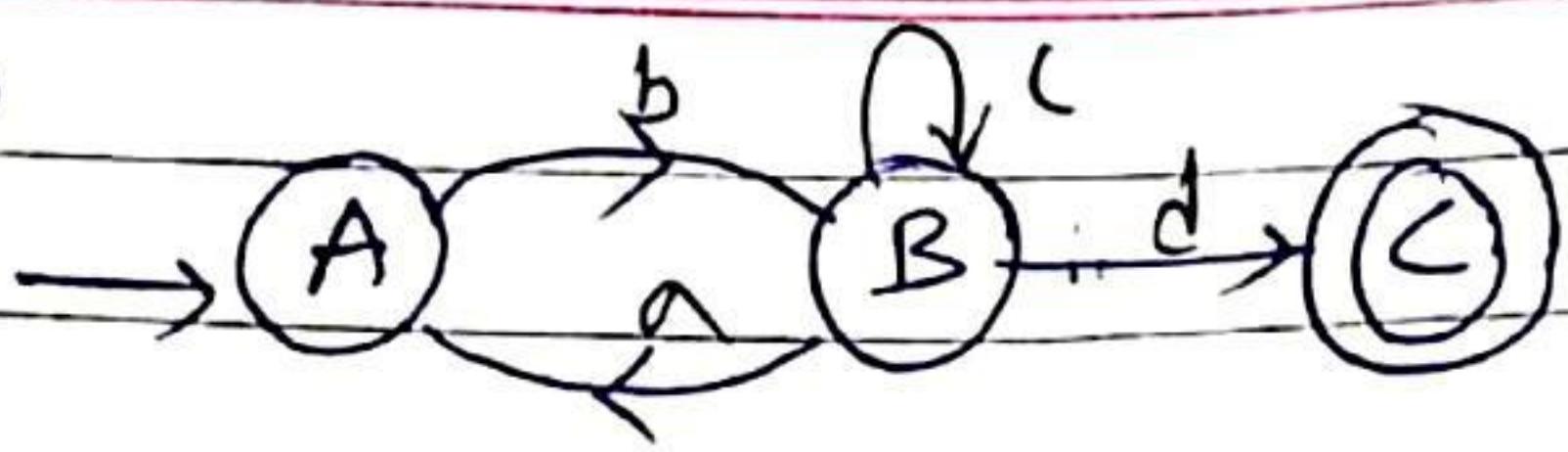
$$RE = \mathcal{O}(10)^*$$

6

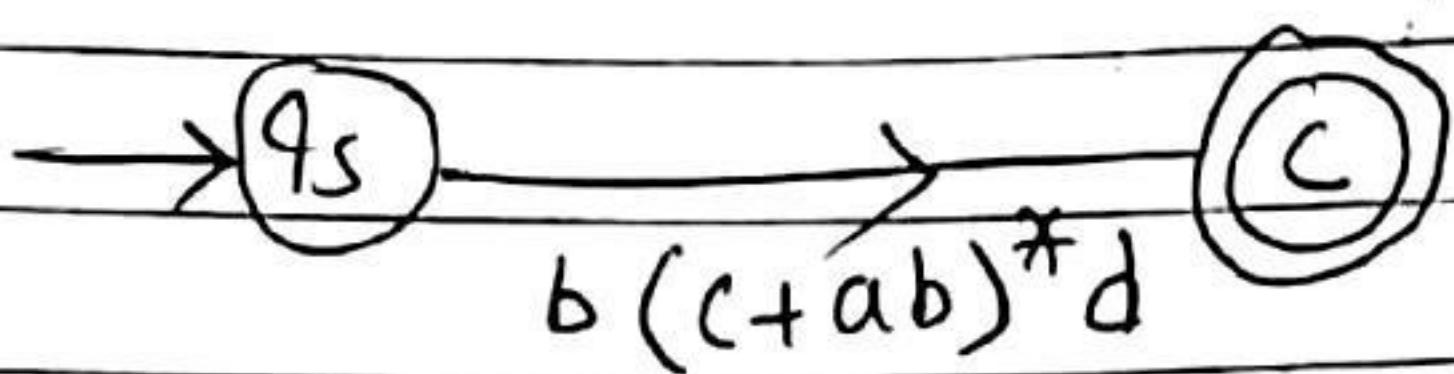
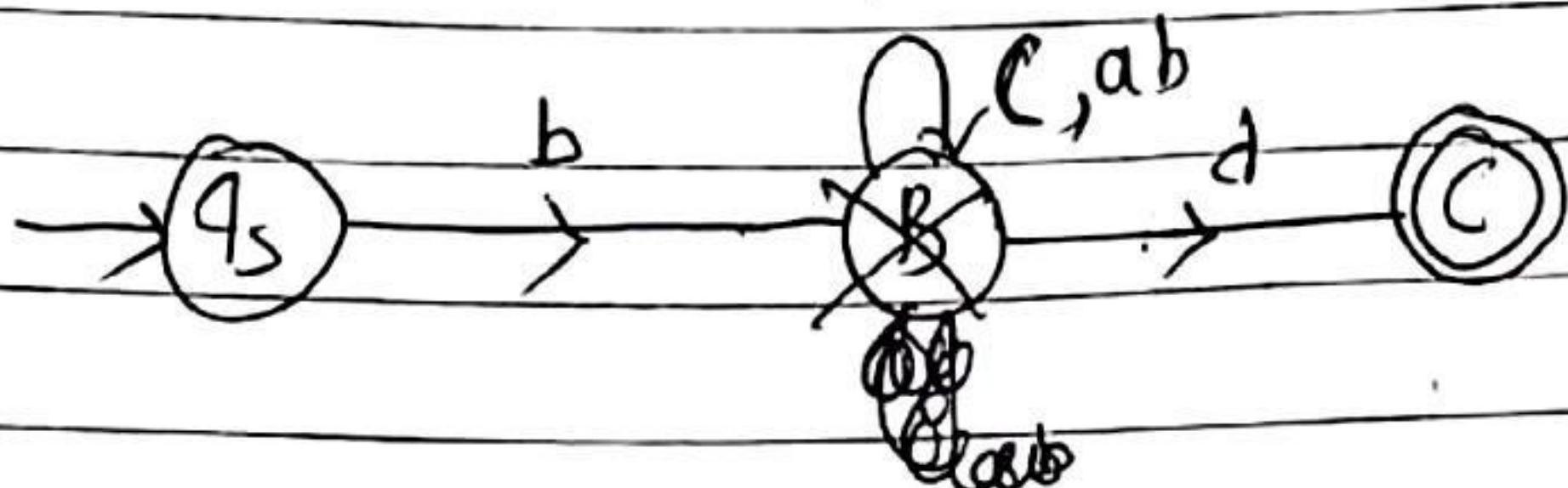
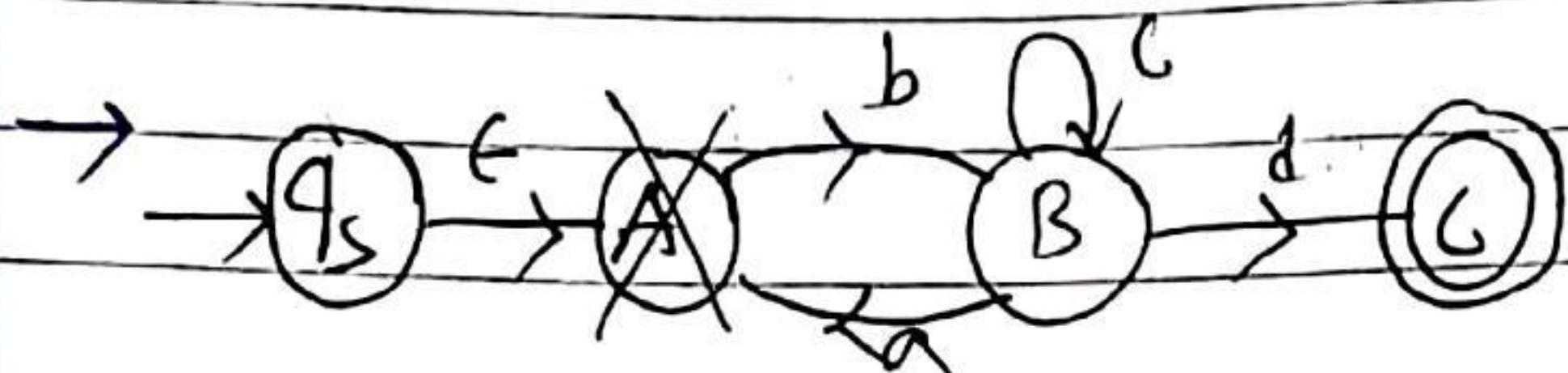


$$RF = a(b+c+d)$$

(V)

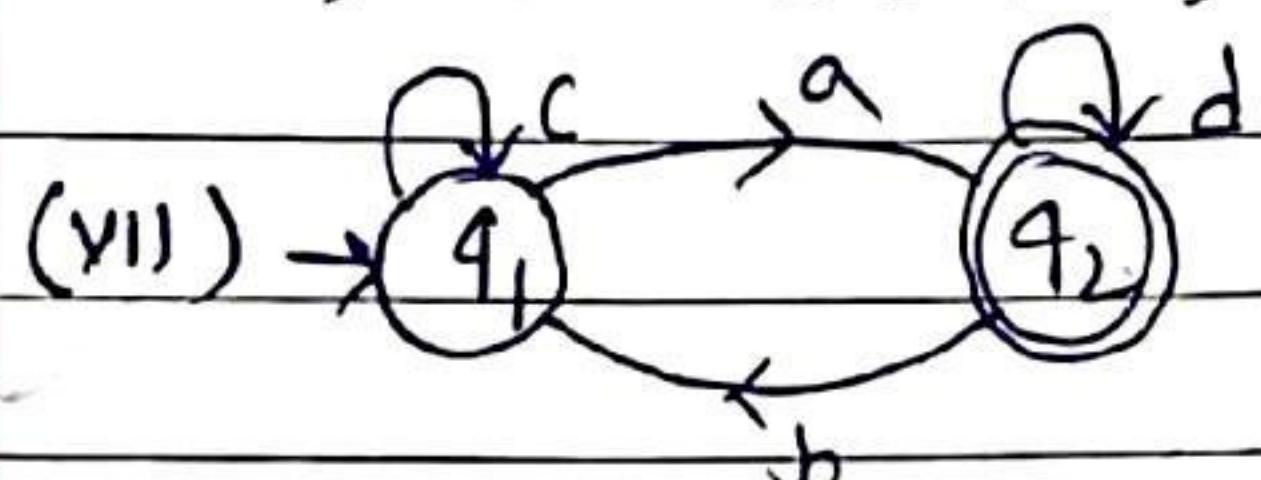


(VI)

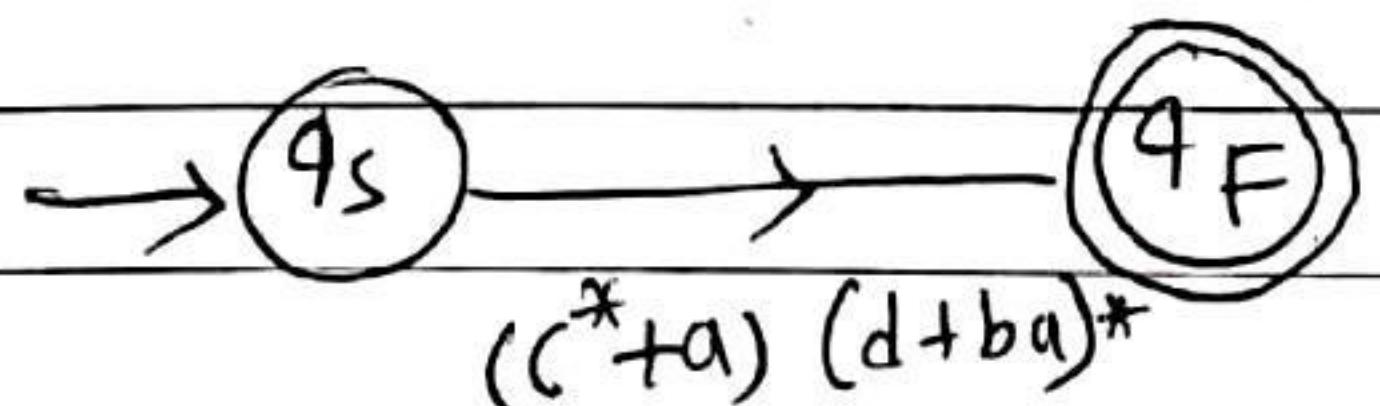
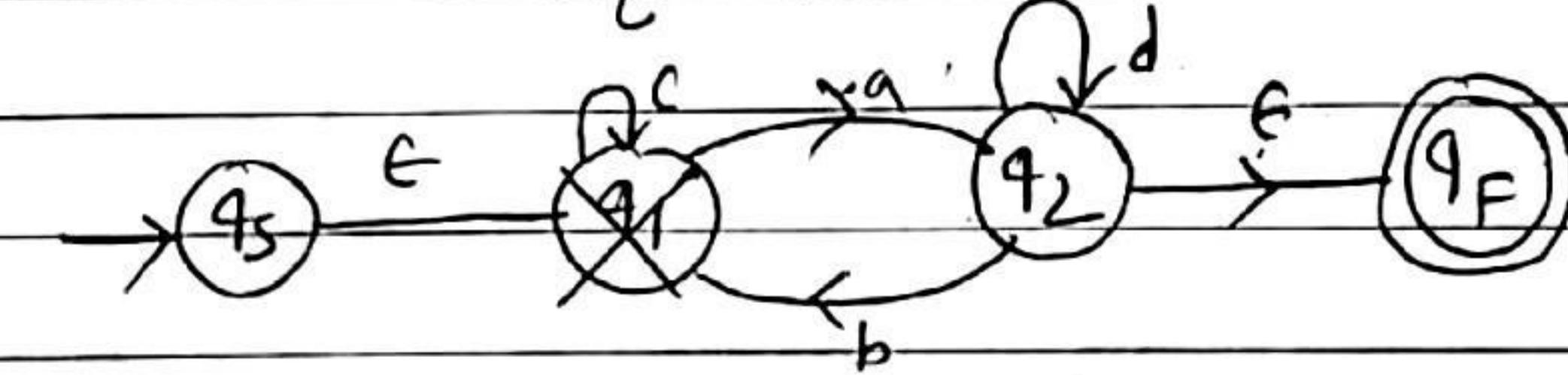
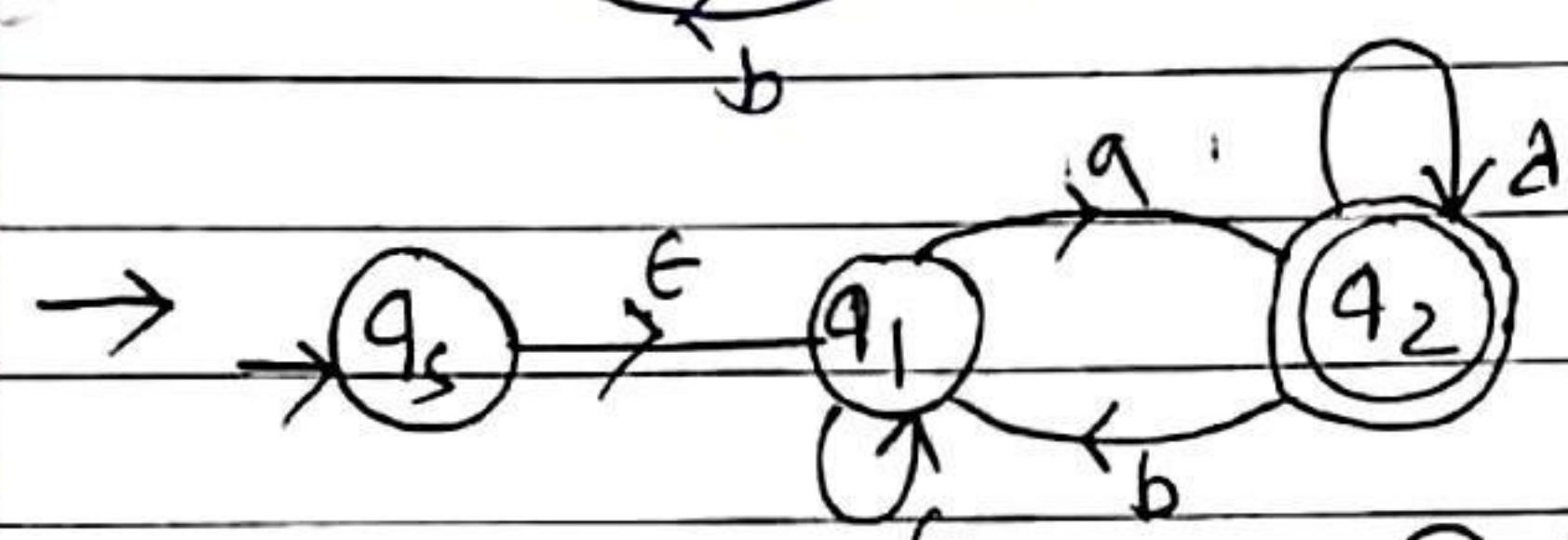


$$\text{hence RF} = b(c+ab)^*d.$$

(VII)

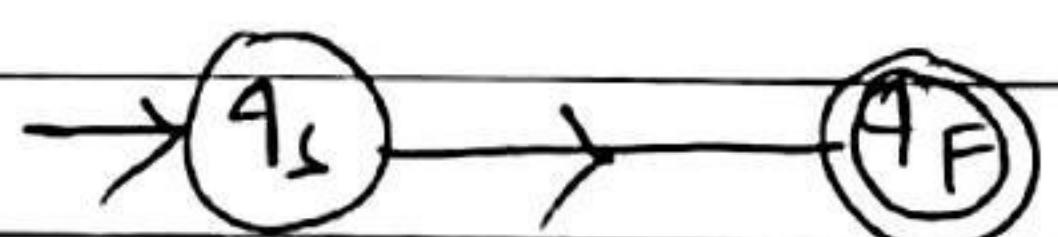
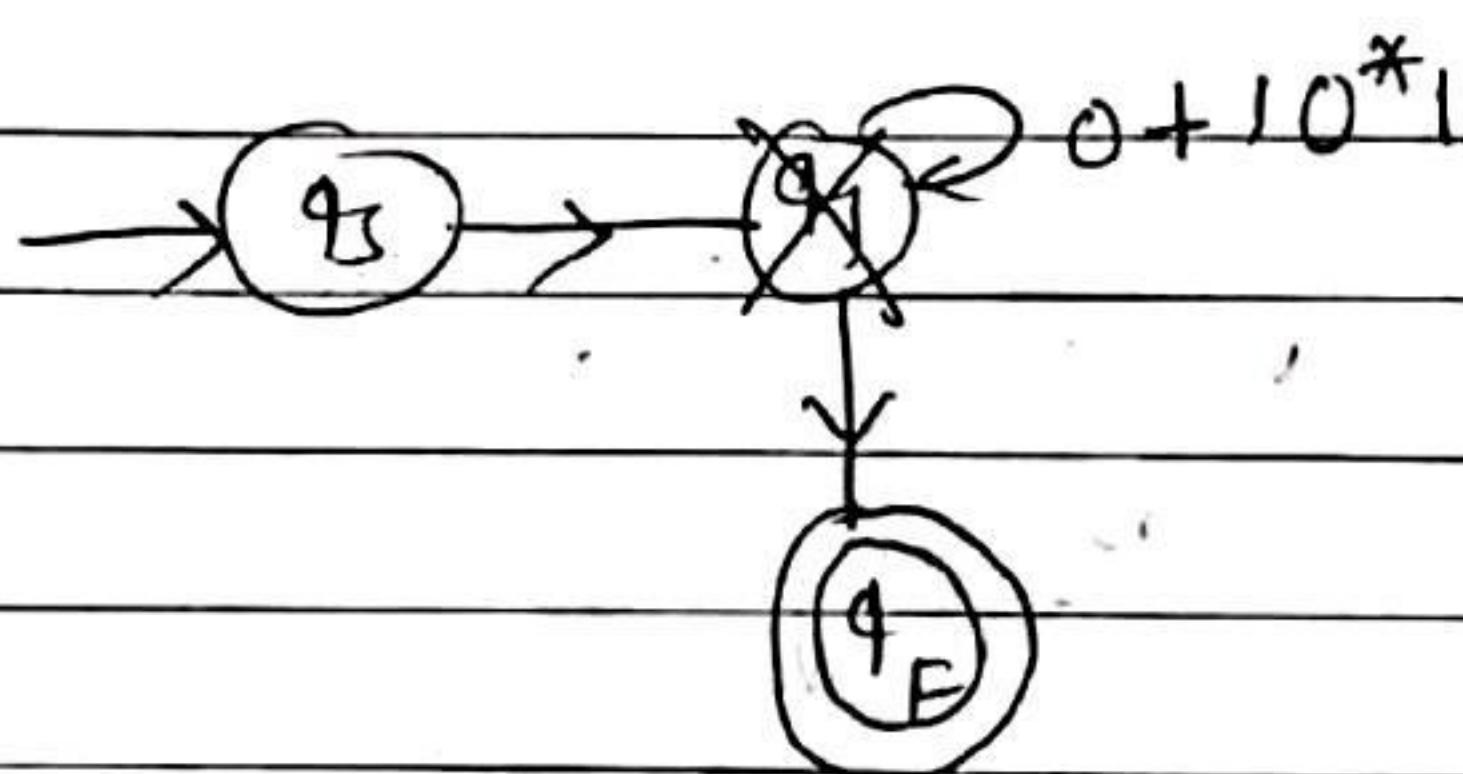
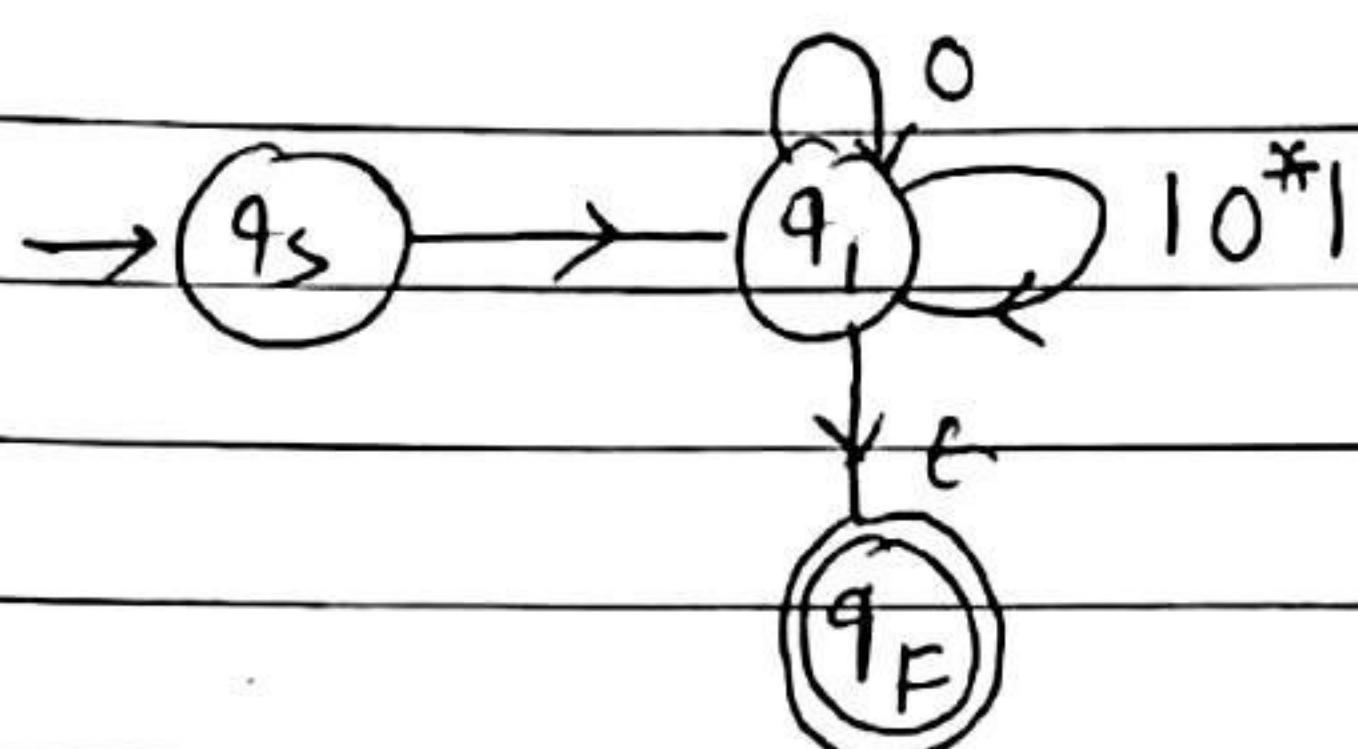
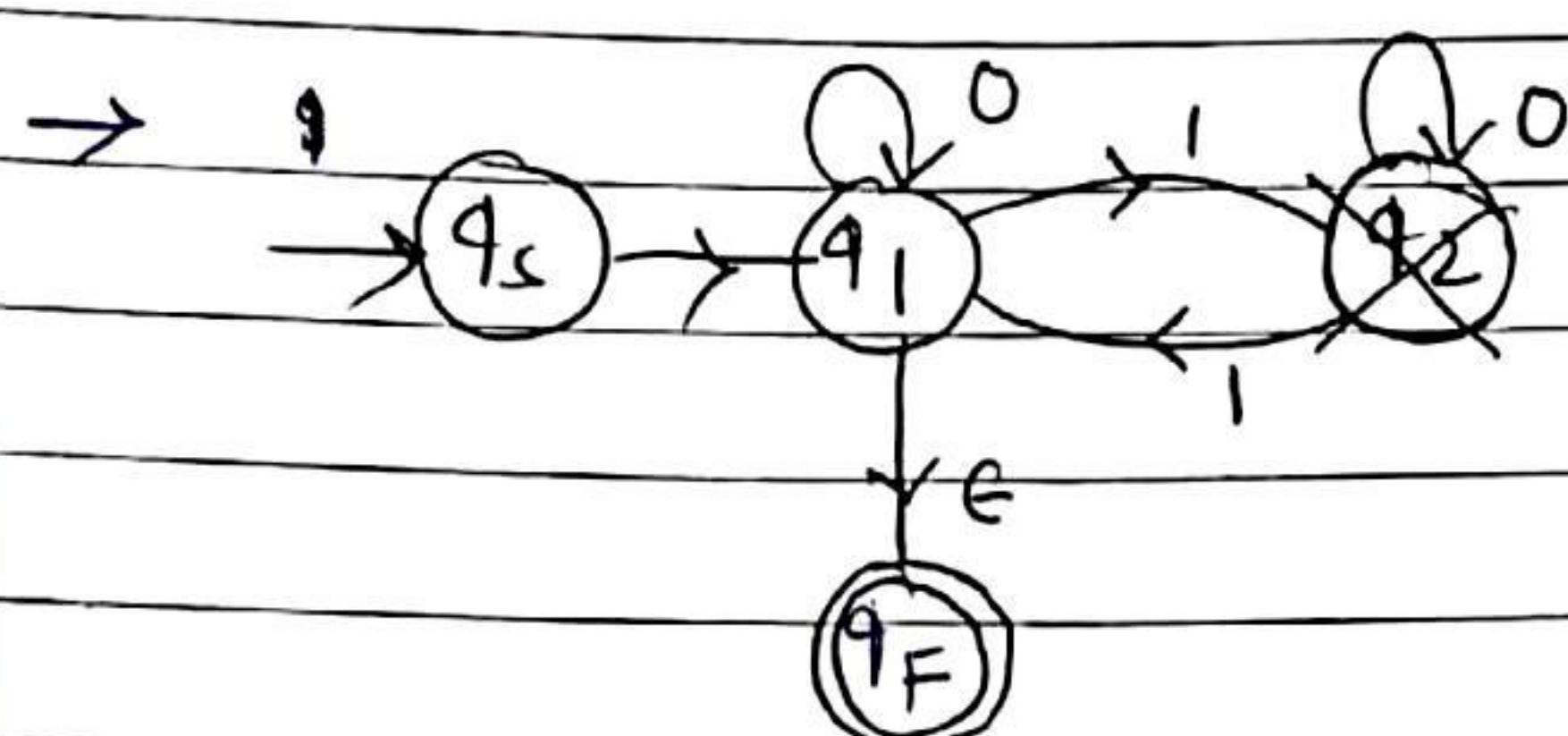
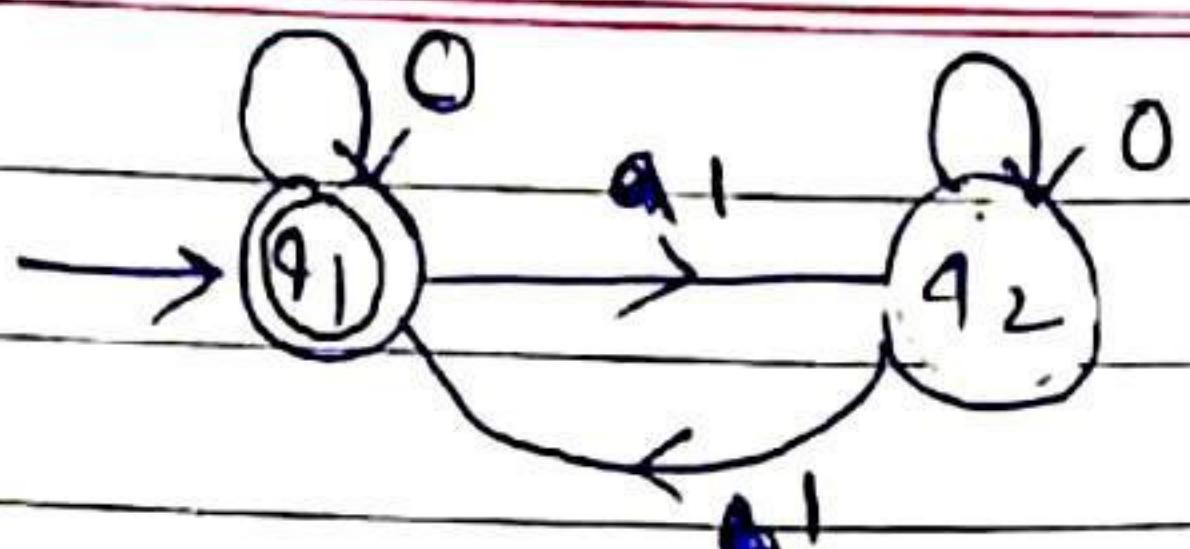


(VIII)

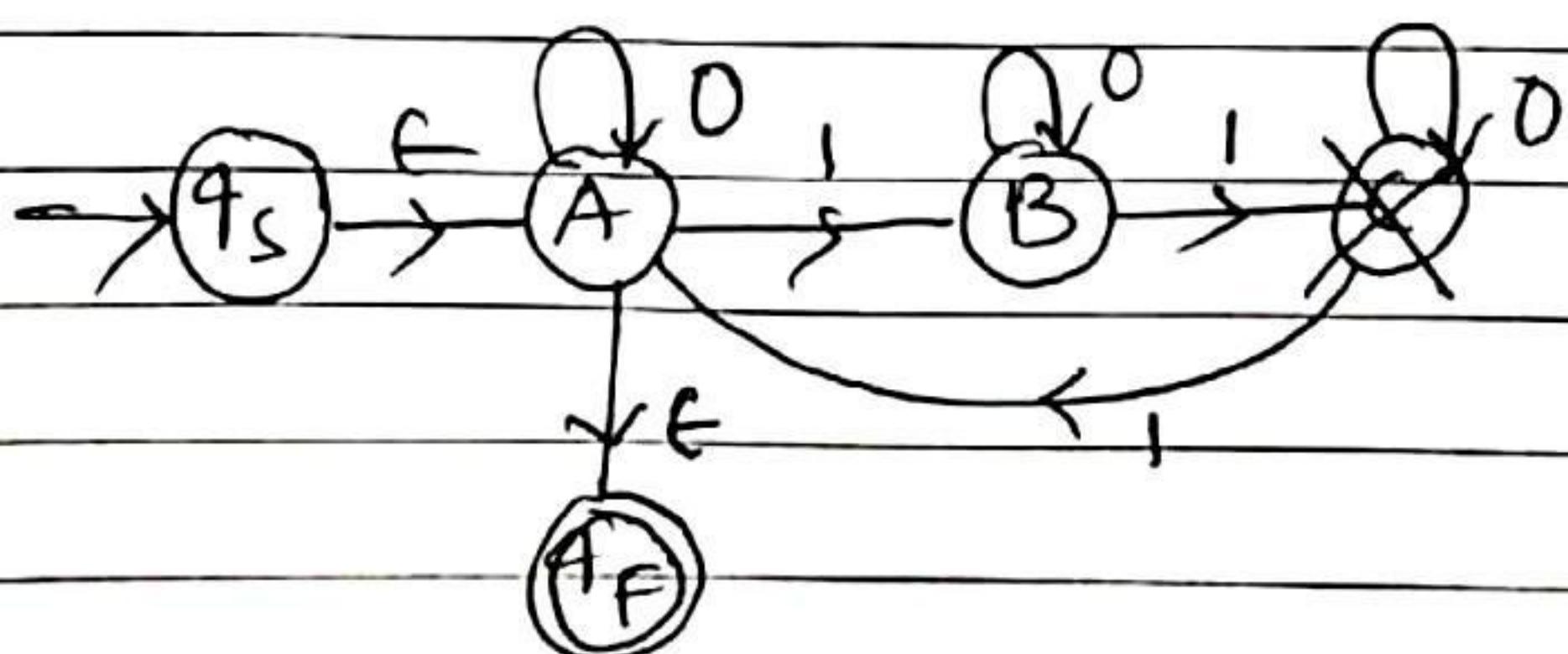
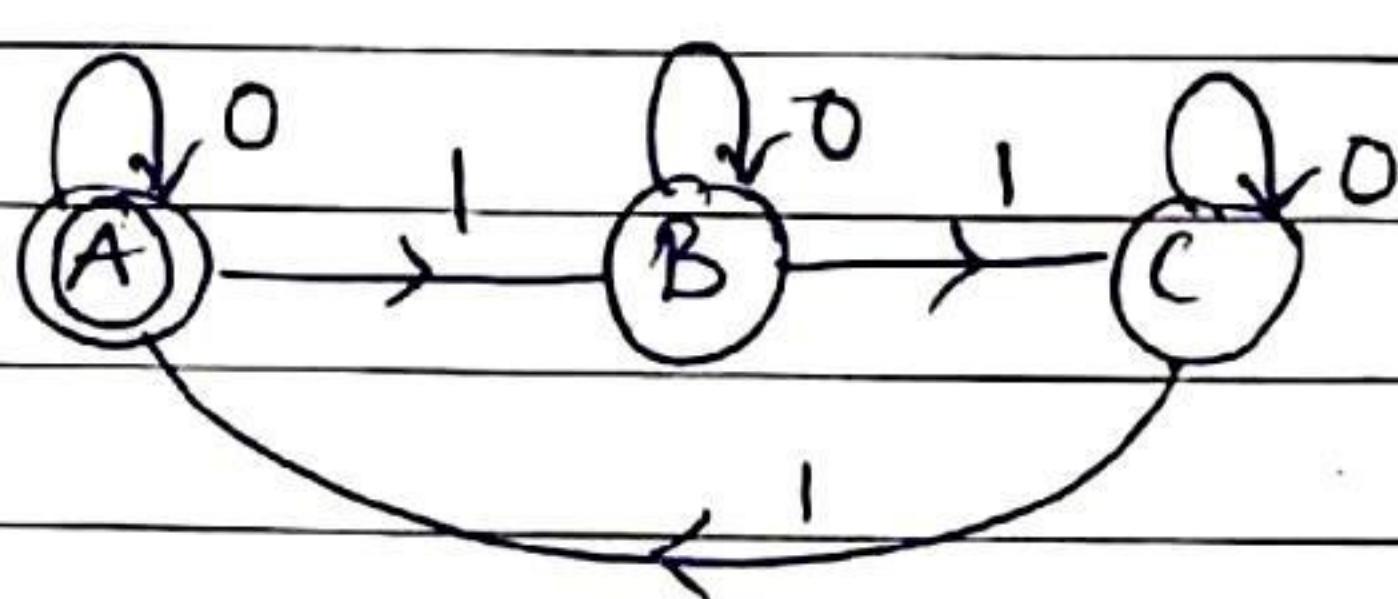


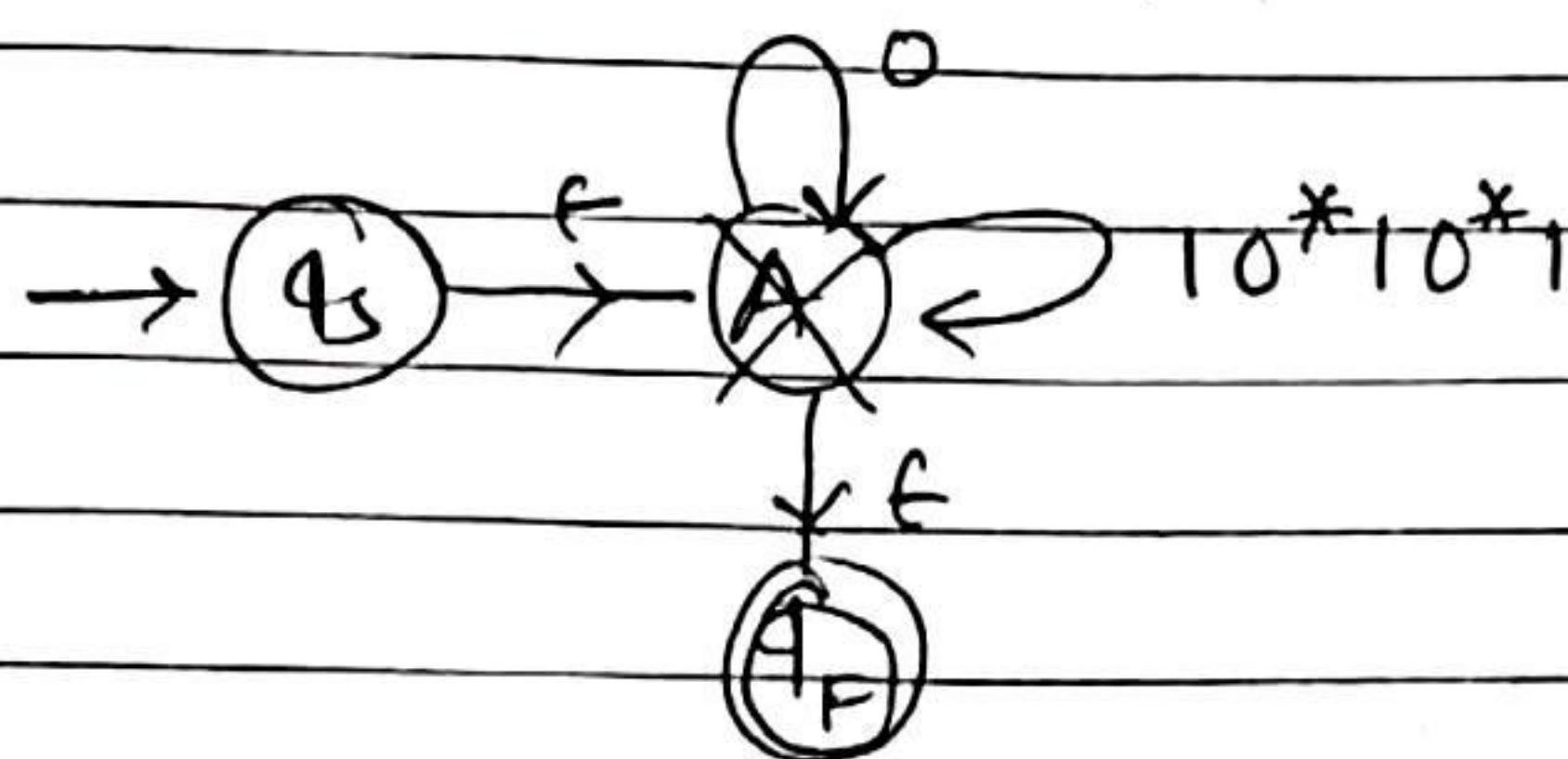
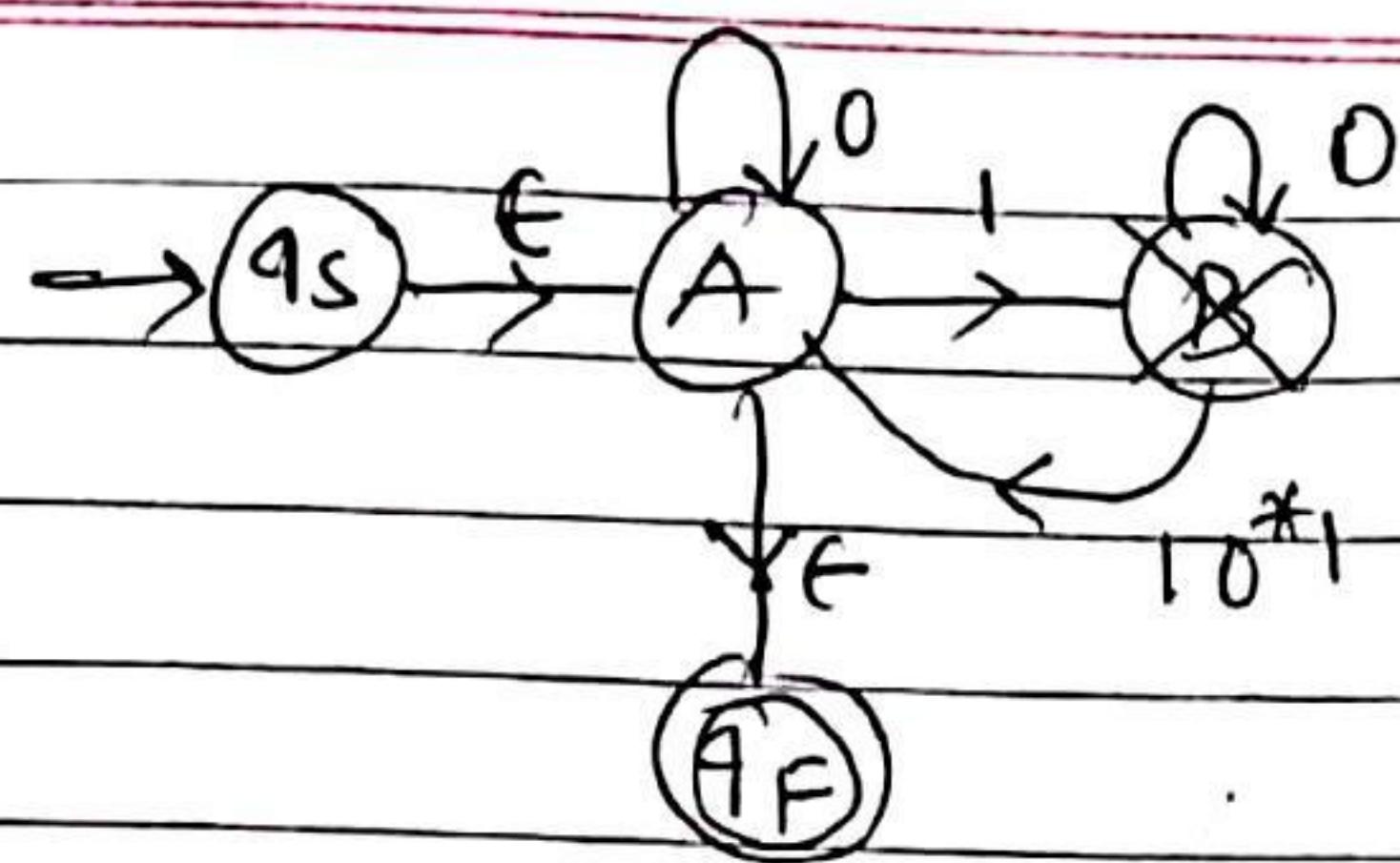
$$RF = ((c+a)*(d+ba))^*.$$

(VIII)



(IX)





$$RE = (10^0 10^1 + 0)^*$$

• Translating regular expression into equivalent DFA

Temporary strings in stack

Stack

✓ Arden's Theorem:

$$\textcircled{*} R = Q + RP \quad \text{where, } P, Q, R \text{ are RE.}$$

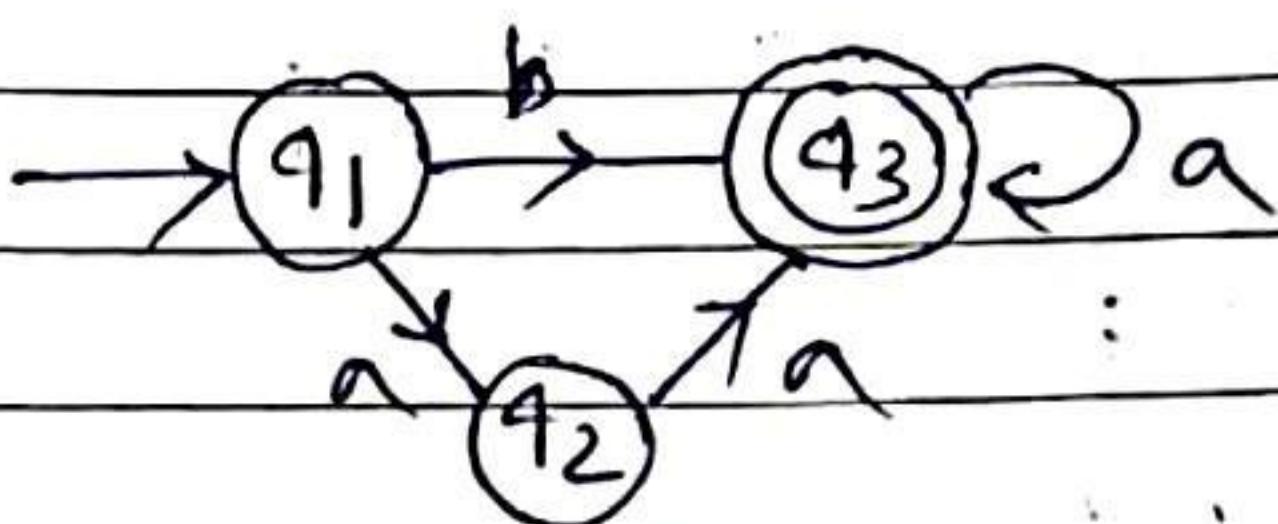
The above equation has a unique solution,

$$R = QP^* \quad \text{where, } \lambda \notin P$$

→ Arden's theorem is used to determine regular expression requested by a transition diagram.

$$\begin{aligned} R &= Q + RP \Rightarrow Q + QP^*P \\ &\Rightarrow Q(1 + P^*) \Rightarrow QP^*. \end{aligned}$$

[Ex-1] Find RE wif from FA using Arden's theorem :-



$$\rightarrow q_1 \Rightarrow \lambda$$

$$q_2 \Rightarrow q_1 \cdot a \Rightarrow \lambda a$$

$$q_3 \Rightarrow q_1 \cdot b + q_2 \cdot a + q_3 \cdot a \Rightarrow \lambda b + \lambda a a + q_3 a$$

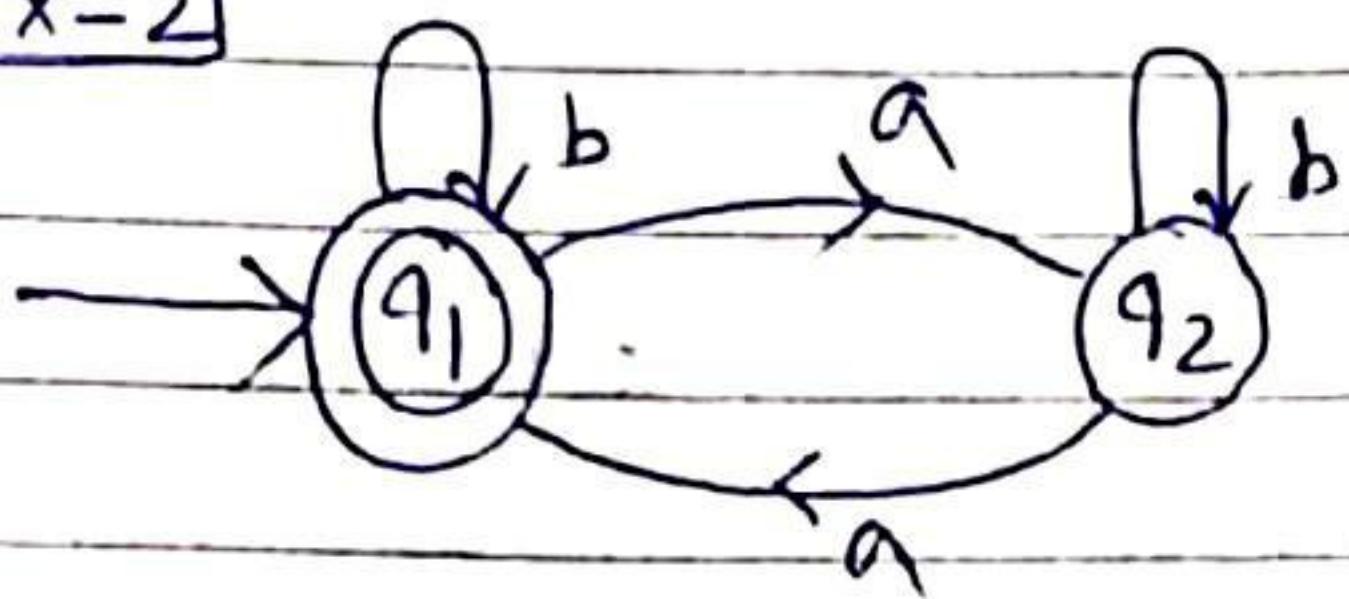
$$\frac{q_3}{r} \Rightarrow \left(\frac{b + aa}{q} \right) + \frac{q_3 a}{r}$$

$$q_3 \Rightarrow \boxed{(b + aa) a^*}$$

$$\therefore r = qp^*$$

→ If here more than 1 finial state , then find RE for every finial state then add every RE .

Ex-2



$$q_1 = \lambda + q_1 \cdot b + q_2 \cdot a \quad (I)$$

$$\frac{q_2}{n} = \frac{q_2 \cdot b}{n} + \frac{q_1 \cdot a}{n} \quad (II)$$

use Arden's theorem on eq-(II)

$$\therefore R = \boxed{\lambda + RP}$$

$$q_2 = q_1 a + q_2 b$$

$$q_2 = (q_1 a) b^*$$

put the value of q_2 in q_1 equation -

$$q_1 = \lambda + q_1 b + (q_1 a) b^* a$$

$$\frac{q_1}{n} = \lambda + \frac{q_1}{n} (b + a b^* a) \quad (\text{again used Arden theorem})$$

$$\text{if } q_1 = \lambda (b + a b^* a)^*$$

$$q_1 = \boxed{(b + a b^* a)^*}$$

(RE)

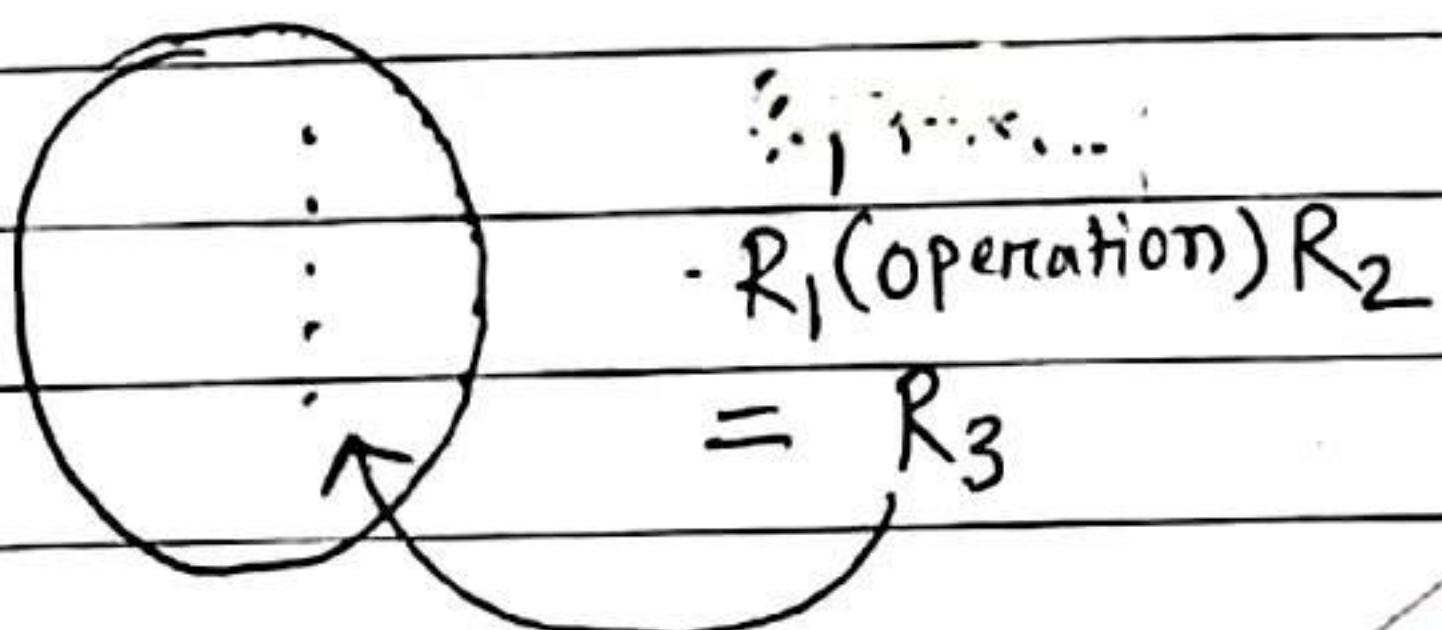
Properties of Regular Language:

→ Regular languages are closed under following Operations -

- ✓ ① Union
- ✗ ② Intersection
- ✓ ③ Concatenation.
- ✓ ④ Reversal.
- ✗ ⑤ Kleene closure.
- ✓ ⑥ Difference
- ✓ ⑦ Homomorphism.
- ✗ ⑧ Inverse Homomorphism.
- ✓ ⑨ Quotient Operation.
- ✓ ⑩ INIT Operation.
- ✗ ⑪ Substitution.
- ✗ ⑫

✗ not under infinite union. (13)

RL



$$(1) L_1 \cup L_2$$

$$\downarrow \quad \downarrow$$

$$R_1 + R_2$$

$$(5) I_1 = \Sigma^* - L_1$$

$$(3) L_1 \cdot L_2$$

$$\downarrow \quad \downarrow$$

$$R_1 \circ R_2$$

$$(2) L_1 \cap L_2$$

$$= \overline{L_1} \cup \overline{L_2}$$

$$(4) L_1^*$$

$$\downarrow$$

$$R_1^*$$

$$(6) L_1 - L_2 = L_1 \cap \overline{L_2}$$

$$(7) \text{ DFA } \xrightarrow{(R)} \text{ FA } (L^R)$$

⑧ The function $h: \Sigma \rightarrow \Gamma^*$ is called homomorphism.
RL are closed under homomorphism.

Ex: $\Sigma = \{a, b\}$ $\Gamma = \{0, 1, 2\}$

$$h(a) = 101$$

$$h(b) = 112$$

homomorphic image of string are.

$$h(ab) = 101112101$$

homomorphic image of language -

$$h(L_1) = a^* b = (101)^* 112$$

⑨ Inverse homomorphism -

→ The inverse of homomorphic image of a language
is called IH.

$$\bar{h}^{-1}(L) = \{x / h(x) \text{ is in } L\}$$

for a string w , $\bar{h}^{-1}(w) = \{x / h(x) \in L\}$

Ex- Let $\Sigma = \{0, 1, 2\}$ and $\Delta = \{a, b\}$. Define 'h' by

$$h(0) = a, h(1) = ab, h(2) = ba$$

$$\text{Let, } L_1 = \{ababa\}$$

$$\Rightarrow \bar{h}^{-1}(L_1) = \{110, 022, 102\}$$

$$\begin{array}{c} ababa \\ \Phi \quad | \quad 0 \end{array}$$

$$\begin{array}{c} ab \quad ab \quad a \\ \underline{\underline{0}} \quad 2 \quad 2 \end{array}$$

$$\begin{array}{c} ab \quad ab \quad a \\ \underline{1} \quad 0 \quad 2 \end{array}$$

$$\text{Let, } L_2 = a(ba)^*$$

$$= \{a, aba, ababa, \dots\}$$

$$\bar{h}^{-1}(L_2) = \left\{ \underset{02}{0}, \underset{022}{10}, \underset{012}{011}, \underset{022}{022}, \underset{02}{102}, \dots \right\}$$

exhibit

ab ab ab a
 1 1 1 0

ab ab ab a
 0 2 2 2

ab ab ab a
 1 0 2 2

$1^* 0$
 +
 $0^2 *$
 +
 $1^* 0^2 *$

$$h^{-1}(L_2) = \{1^* 0^2 *\}$$

(Ex-2) Let $\Sigma = \{0, 1\}$ and $\Delta = \{a, b\}$.

Define 'h' by $h(0) = aq$, $h(1) = aba$,

Let, $L = (ab + ba)^* a$.

$$\rightarrow L = \{ \underset{1}{\overset{x}{a}}, \underset{1}{\overset{x}{aba}}, \underset{1}{\overset{x}{baa}}, \underset{1}{\overset{x}{ababa}}, \underset{1}{\overset{x}{babaa}}, \dots \}$$

$$h^{-1}(L) = \{1\}$$

$$h(h^{-1}(L)) = \{aba\}$$

$\neq L$

** $h(h^{-1}(L)) \subset L$

(10) Let L_1 and L_2 be languages on the same alphabet. Then the right quotient of L_1 with L_2 is defined as,

$$L_1 / L_2 = \{ u : \exists y \in L_1 \text{ for some } y \in L_2 \}$$

(Ex-) $L_1 = \{01, 001, 101, 0001, 1101\}$
 $L_2 = \{01\}$

$$L_1 / L_2 = \{\epsilon, 0, 1, 00, 11\}$$

$$\text{Ex- } L_1 = 10^* 1, \quad L_2 = 0^* 1$$

$$L_1 / L_2 = ?$$

$$L_1 = \{11, 101, 100, 1, 10001, \dots\}$$

$$L_2 = \{1, 01, 1001, 0001, 00001, \dots\}$$

$$L_1 / L_2 = \{1, 10, 100, 1000, \dots\} \\ = 10^*$$

(11) INIT Operation:

Ex:- $L = \{a, ab, aabb\}$ (apply prefix operation)

$$\text{Init}(L) = \{\epsilon, a, \overline{a}, \overline{ab}, \overline{a}, \overline{aabb}\}$$

(12) Under substitution:

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

$$f(a) = 0^*, f(b) = 01^*$$

$$L = a + b^*$$

$$\Rightarrow f(L) = 0^* + (01^*)^*$$

$$(13) \quad L_1 = \{a^1 b^1\}$$

$$L_1 \cup L_2 \cup L_3 \dots$$

$$L_2 = \{a^2 b^2\}$$

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

$$L_3 = \{a^3 b^3\}$$

so, RL not closed under infinite union.

REGULAR GRAMMAR

- Type 3 Grammar called Regular grammar.

→ If the grammar has all the production of the form -

$$A \rightarrow \alpha(B)/\beta$$

here, $A, B \in V$ (variables)

& $\alpha, \beta \in T^*$ (Terminals)

Right linear grammar
(RLG)

$$A \rightarrow B\alpha/\beta$$

here, $A, B \in V$

$\alpha, \beta \in T^*$

Left linear grammar
(LLG)

Ex of LLG -

$$\text{here, } A \rightarrow B\alpha/a$$

$$B \rightarrow B\alpha/Bb/a/b$$

Ex of RLG -

$$A \rightarrow aB/a$$

$$B \rightarrow aB/bB/a/b$$

→ Either they should be RLG or LLG, they should not be both of them.

Ex 01 -

$$A \rightarrow B\alpha/a$$

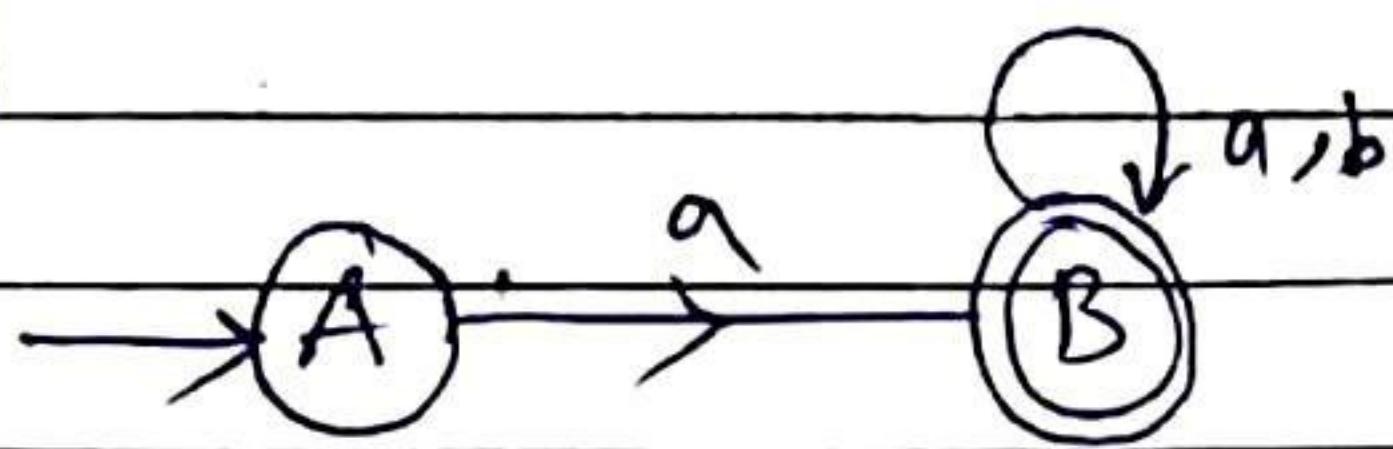
$$B \rightarrow aB/a$$

not type-3 grammar.

→ Whatever language are generated by type-3 grammar they are Regular Language.

Conversion FA to RG (Regular grammar): (Type-3)

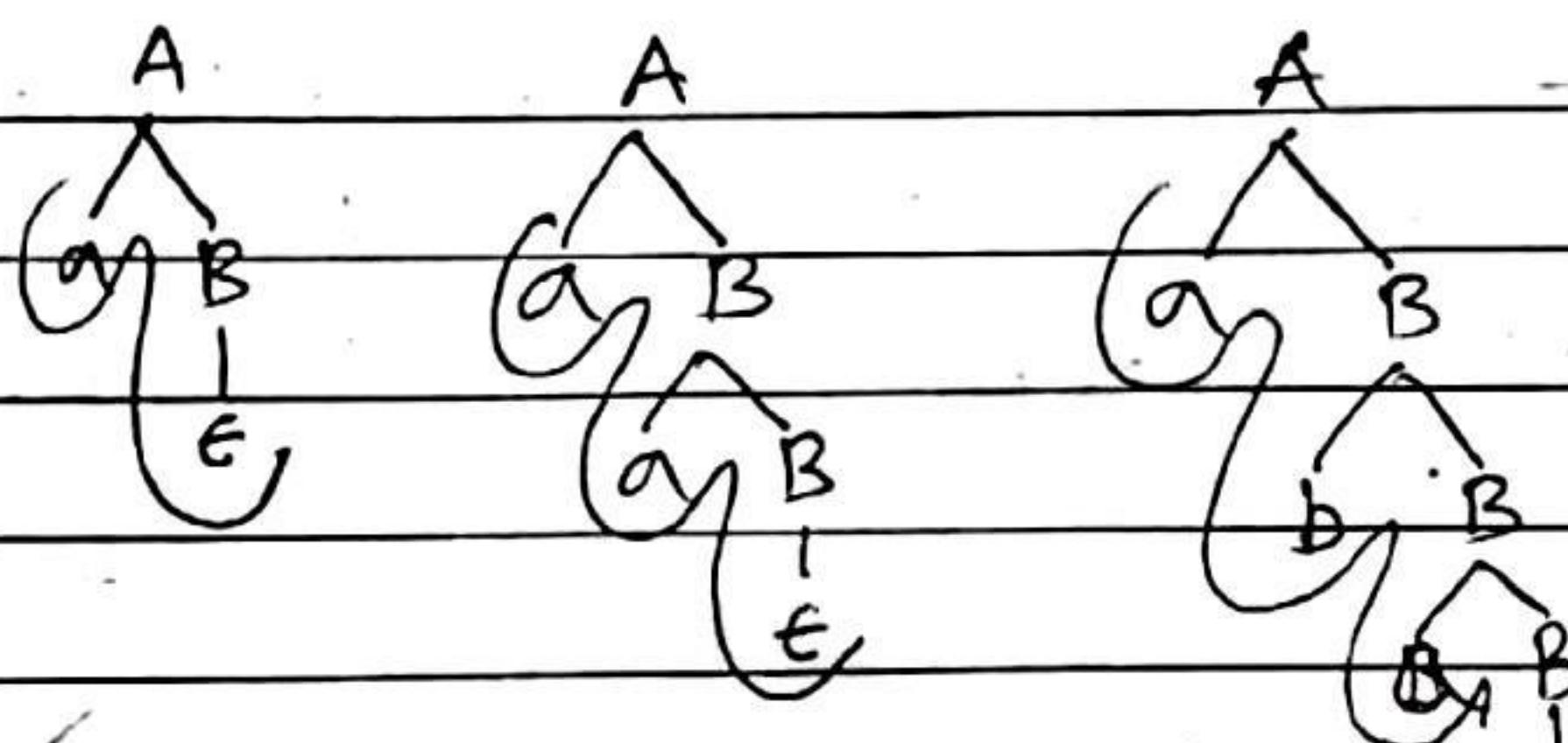
Ex-1



→ FA represent set of all string start with 'a'.

$$\boxed{A \rightarrow aB} \quad \boxed{B \rightarrow aB/bB/\epsilon} \quad \text{RLG}$$

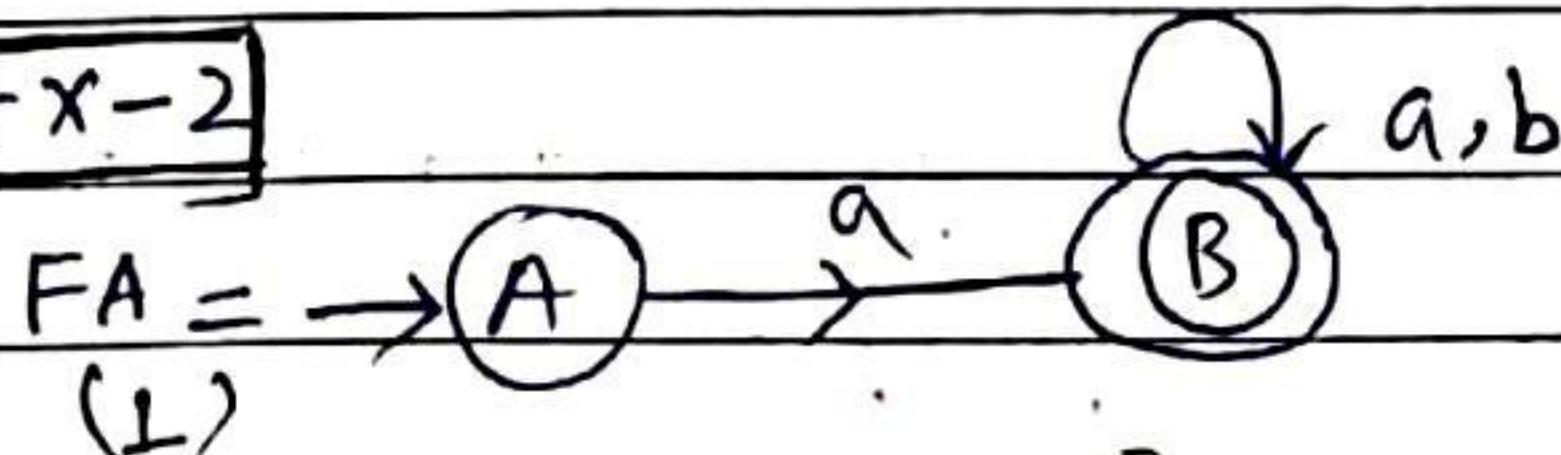
→ for the give FA it is the RLG.



* $\boxed{\begin{array}{l} \text{FA} \rightarrow \text{RLG} \\ (\text{L}) \end{array}} \xrightarrow{R} \boxed{\begin{array}{l} \text{LLG} \\ (\text{LR}) \end{array}}$

* $\boxed{\begin{array}{l} \text{FA} \xrightarrow{R} \text{FA} \rightarrow \text{RLG} \xrightarrow{R} \text{LLG} \\ (\text{L}) \quad (\text{LR}) \quad (\text{LR}) \end{array}} \rightarrow \text{conversion FA to LLG.}$
 $\boxed{\begin{array}{l} \text{U} \\ (\text{II}) \end{array}} \quad \boxed{\begin{array}{l} \text{III} \\ =(\text{L}) \end{array}}$

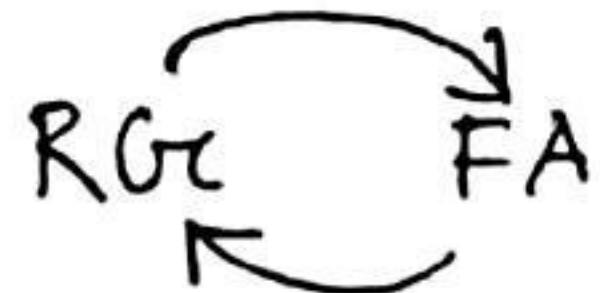
Ex-2



Reversal of FA = (L^R)



$(R\text{-FA} \rightarrow \text{RLG}), \boxed{\begin{array}{l} B \rightarrow aB/bB/a \\ A \rightarrow \epsilon \end{array}} \quad \text{RLG of FA.}$



Now, Reversal of RLGr,

$$\boxed{\begin{array}{l} B \rightarrow Ba / Bb / a \\ A \rightarrow e \end{array}} - LLGr \quad (I^R)^R = I$$

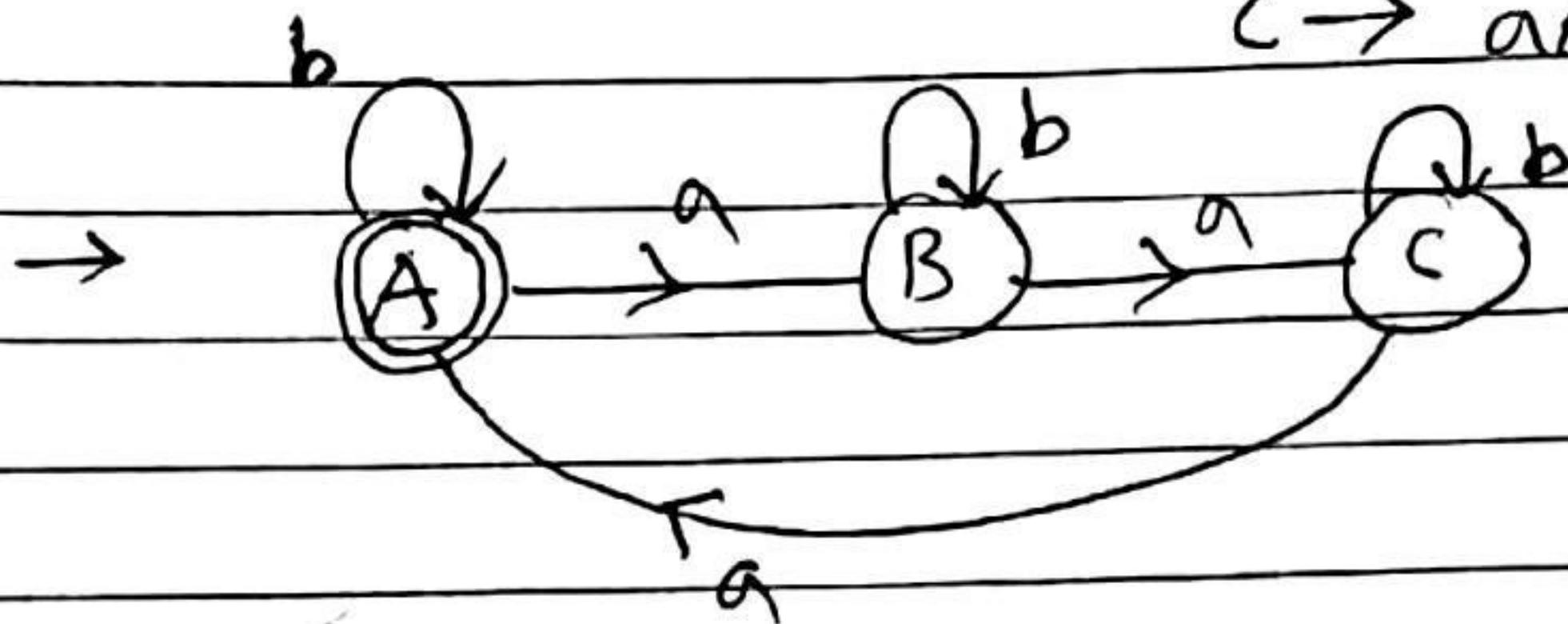
Conversion RLGr to FA :

[Ex-1] given RLGr are,

$$A \rightarrow aB / bA / b$$

$$B \rightarrow ac / bB$$

$$C \rightarrow aA / bC / a$$



[Ex-2] Converting LLGr \rightarrow FA

$$\boxed{\begin{array}{c} LLGr \xrightarrow{R} RLGr \xrightarrow{(I^R)} FA \xrightarrow{R} FA \\ (L) \qquad \qquad \qquad (I^R) \qquad \qquad \qquad (I^R)^R = I \end{array}}$$

\rightarrow RLGr and FA both are equivalent in powers.

- Decidable problem on RL :-

1. Emptiness problem of FA:

Algo -

s-1) Select all the states which are not reachable in final state from initial state and delete those states and corresponding transitions.

s-2) In the remaining FA, if we find at least one final state, then the language accepted by the given FA is non empty otherwise empty.

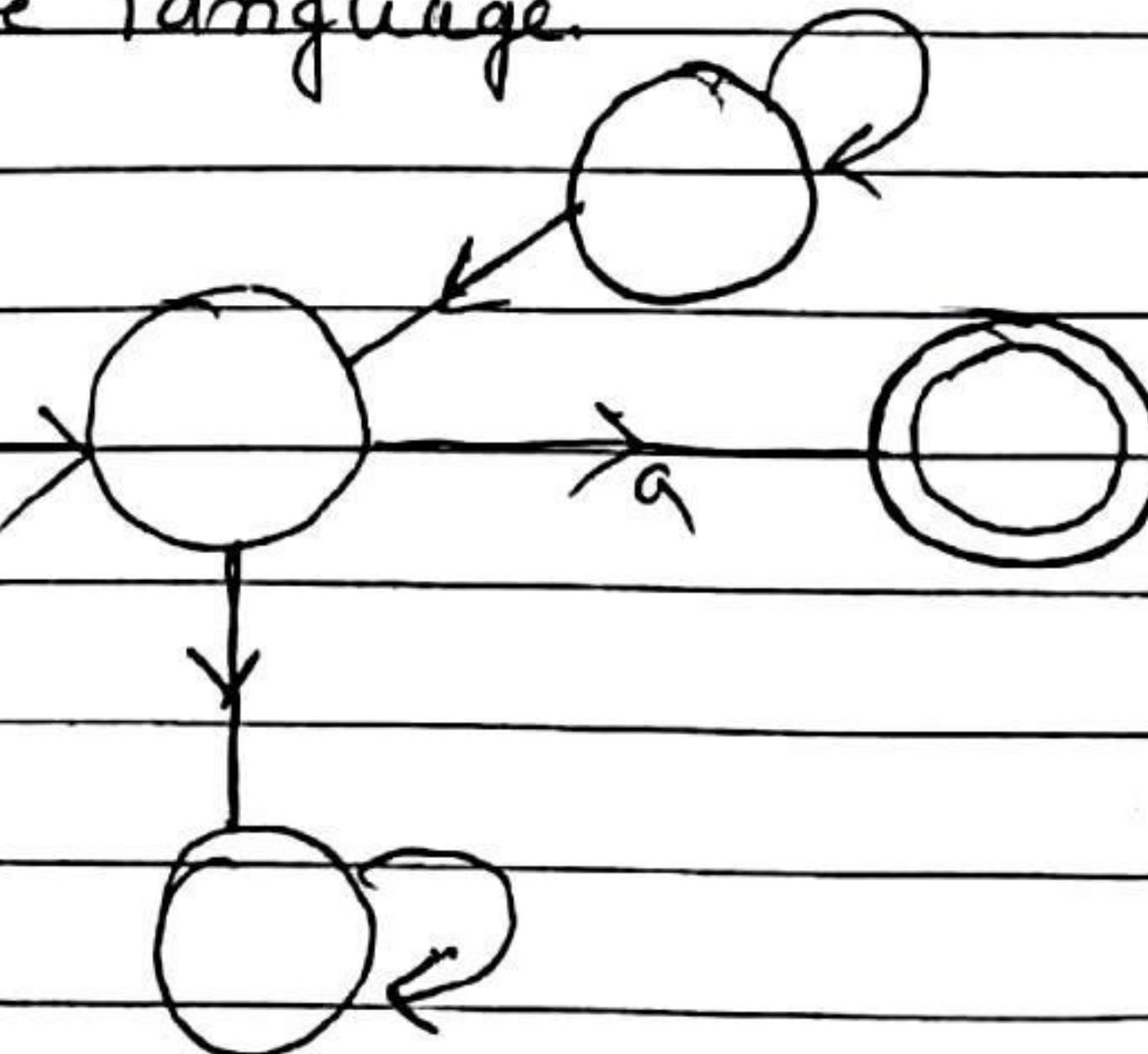
2. Infiniteness problem of RL:

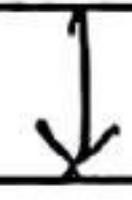
Algo -

s-1) Remove the states which are not reachable from initial state and corresponding transition.

s-2) Delete the states and corresponding transitions, from which we cannot reach final state.

s-3) In the remaining FA, if we find at least one loop and one final state then it is accepting infinite language.



3. equality problem :-IS $L_1 = L_2$ DFA₁ DFA₂DFA_{m1} DFA_{m2}

→ Given Languages are converted into
DFA, then minimize them → After minimization
compare them to check both are equal or not.