10/12/19

problem ⟶ language

↓ ↓

program automata (Finite Automata, Turing Machine)

↓ ↓

execute output (Accepted / Rejected)

Symbol: Any input that is real or abstract given to a system is a symbol. e.g.: integer, digit, character, .jpg file, text file

Alphabet: The non-empty finite set of ~~attribute~~ symbols is called Alphabet set, denoted by Σ.

$\Sigma = \{1\}$ unary $\quad\quad$ $\Sigma = \{A, \ldots Z, a, \ldots z\}$

$\Sigma = \{0, 1\}$ binary $\quad\quad$ $\Sigma = \{\uparrow, \rightarrow, \searrow, \varnothing\}$

String: The sequence of symbols from a given alphabet set. e.g.: $\Sigma = \{0, 1\}$: 0, 1, 00, 01, 10, 11, 000, ...

Length of string: No. of elements present in a string, denoted by $|w|$. If length of string is 0, it's called empty string denoted by 'λ' or 'ϵ'. $|w| = 0$.

Substring: If 'u' & 'w' are two strings from alphabet set Σ and if 'u' is obtained from 'w', then 'u' is called substring of 'w'.

e.g.: $w = AFL$

$u = A, F, L, AF, FL, AFL, \lambda$

N.B. Every string is substring of itself.
Empty string 'λ' or 'ε' is a substring of every string.

Types of substring:

① Trivial or Improper: The substring 'w' itself and empty string 'λ' or 'ε'.

② Non-Trivial or Proper: All substrings except 'w' and 'λ'.

Suppose, 'w' is a string of length n of different symbols.
$|w| = n$.

total no. of substrings = $\sum n + 1$
total no. of trivial substrings = 2
total no. of non-trivial substrings = $\sum n + 1 - 2 = \sum n - 1$
total no. of non-empty substring = $\sum n$.

Prefix: Sequence of starting symbol.

Suffix: Sequence of ending symbol.

w = AFL

prefix: λ, A, AF, AFL, Proper: All except λ, w.
suffix: λ, L, FL, AFL. Improper: λ, w.

Power String (Set of all strings)

Set of all strings is denoted by Σ^k where Σ^k is a set of all strings of length k.

$\Sigma = \{0, 1\}$; $\Sigma^0 = \{\lambda\}$, $\Sigma^1 = \{0, 1\}$, $\Sigma^2 = \{00, 01, 10, 11\}$.

Σ^* = Set of strings of all combinations of all alphabets

Kleen's Closure: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots$

Positive Closure: $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots$

$\boxed{\Sigma^* - \Sigma^0 = \Sigma^+}$ $\boxed{\Sigma^* = \Sigma^+ \cup \lambda}$

Language: Collection of set of strings from a given alphabet set.

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

L_1 = set of all strings of length 2 = $\{aa, ab, ba, bb\}$

L_2 = set of all strings of length 3 = $\{aaa, aab, aba, abb, bab, baa, bba, bbb\}$

L_3 = set of all strings of length 4 = $\{aaaa, aaab, aaba, abaa, baaa, \ldots\ldots, \ldots\ldots, bbbb\}$

L_4 = set of all strings of all combinations of a's & b's

$= \{\lambda, a, b, aa, ab, ba, bb, \ldots\ldots\}$

$= \Sigma^*$

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

$1^* = \{\lambda, 1, 11, 111, 1111, \ldots\ldots\}$.

$11^* = \{\lambda, 11, 1111, 111111, \ldots\ldots\}$.

$L = \{1^n \text{ or } 1^n \mid n \geq 0\} = \{\lambda, 10, 1100, 111000, \ldots\ldots\}$.

Σ^* is called Universal set or language

12/01/19

Finite Automata

Applicable for regular language. It's a model to check if a language is regular or not.

Deterministic Finite Automata (DFA):

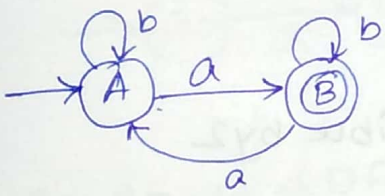
It's a 5 tuple representation.

Q: finite no. of states δ: transition function

Σ: finite set of alphabets q_0: initial state F: final state set

$$M = \{Q, \Sigma, \delta, q_0, F\}$$

$$\delta = Q \times \Sigma \rightarrow Q$$



Transition Table:

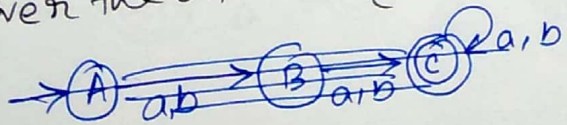
State	a	b
A	B	A
B	A	B

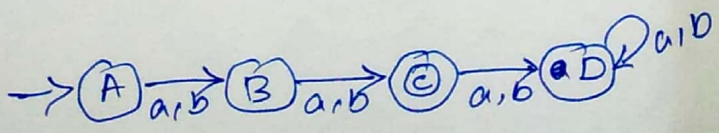
06/01/20

DFA construction:

Q: Construct a DFA where the strings are of length 2. over the alphabet {a, b}.

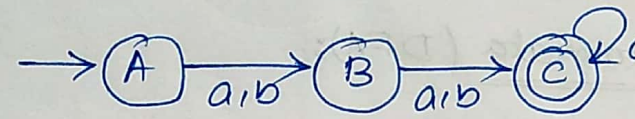
L = {aa, ab, ba, bb}.





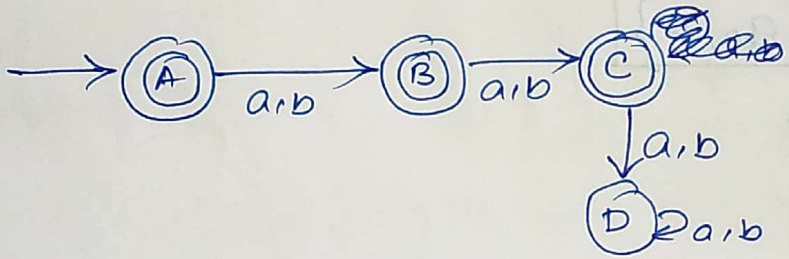
Q: Strings of length at least 2 over {a, b}.

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



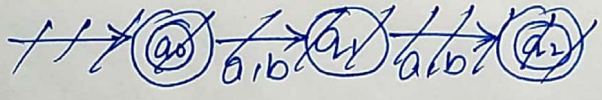
Q: Strings of length at most 2 over {a, b}.

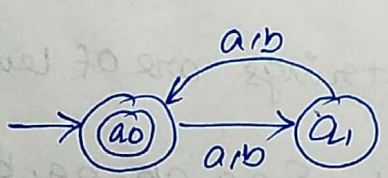
$L = \{\varepsilon, a, b, aa, ab, ba, bb\}$.



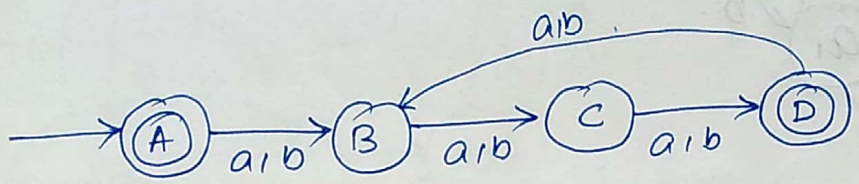
Q: Length of string should be divisible by 2.

$L = \{\varepsilon, ab, aa, bb, ab, bbbb, aaaa, \ldots\}$

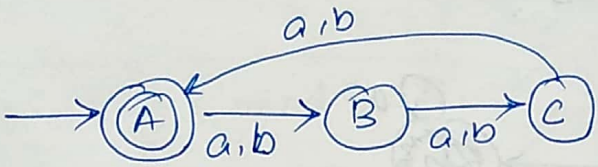




Q: String of length divisible by 3.

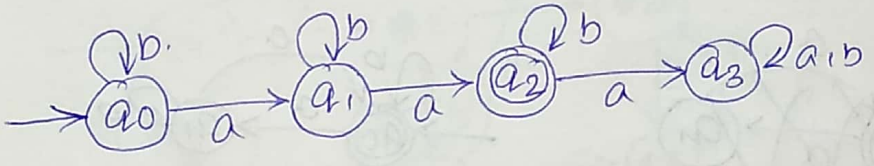


or

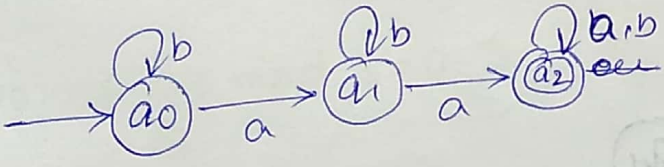


07/01/20

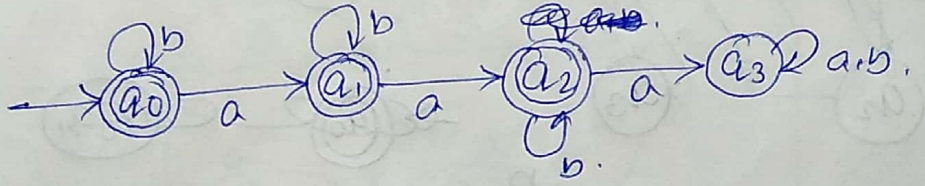
Q: Construct a DFA over alphabet $\{a,b\}^*$ where no. of a's = 2.



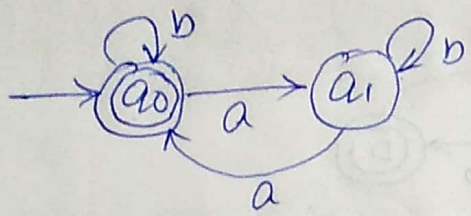
Q. Construct a DFA accepts at least 2 a's.



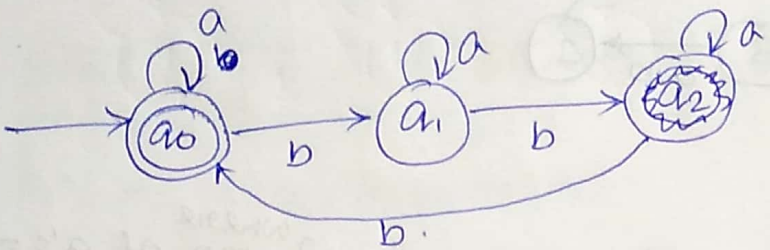
Q: At most 2 a's.



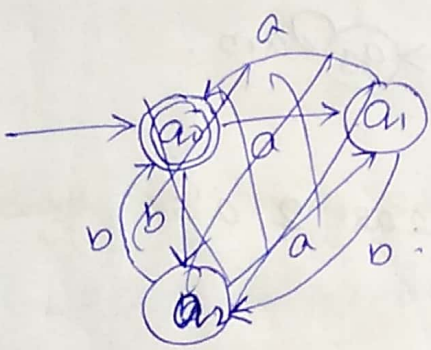
Q: No. of a's divisible by 2.

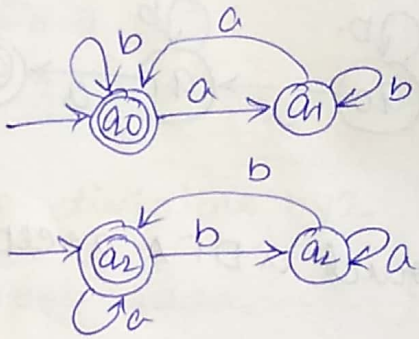


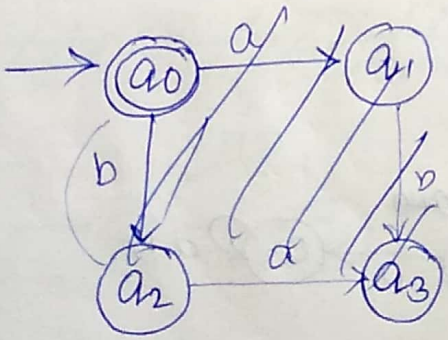
Q: No. of b's divisible by 3.

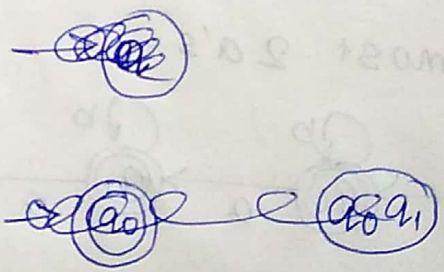


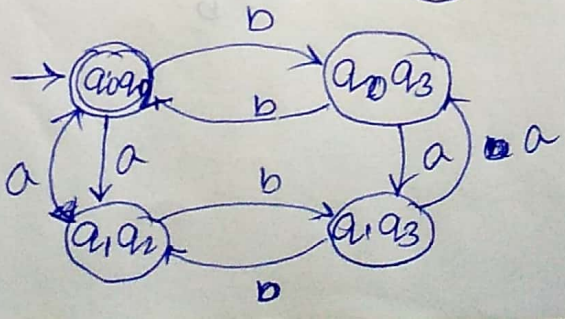
Q: No. of a's divisible by 2 & no of b's. divisible 2

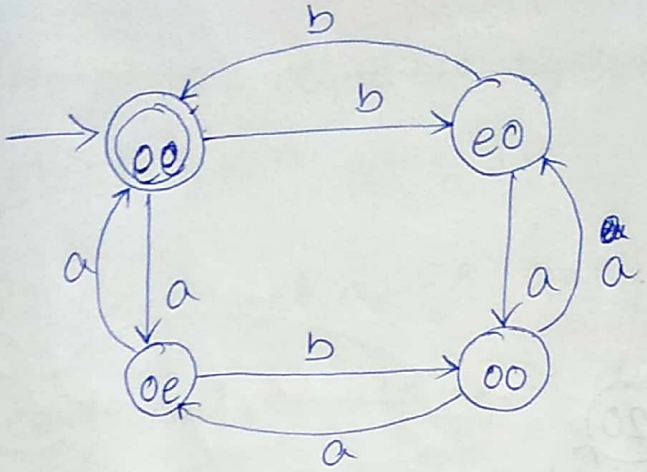




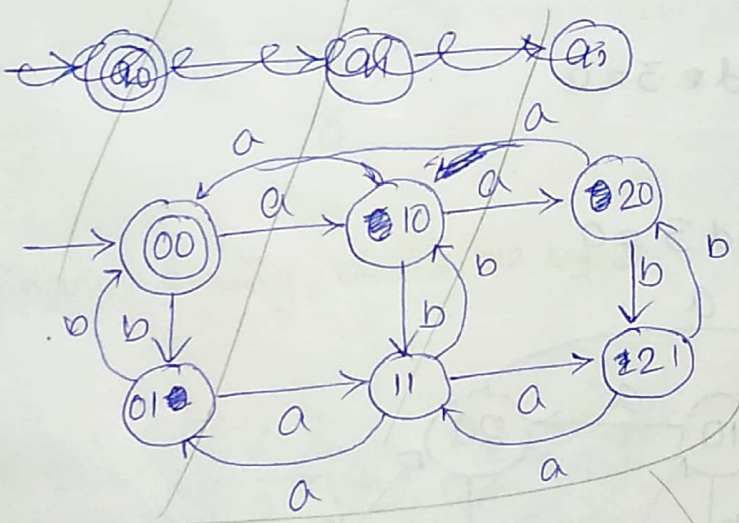




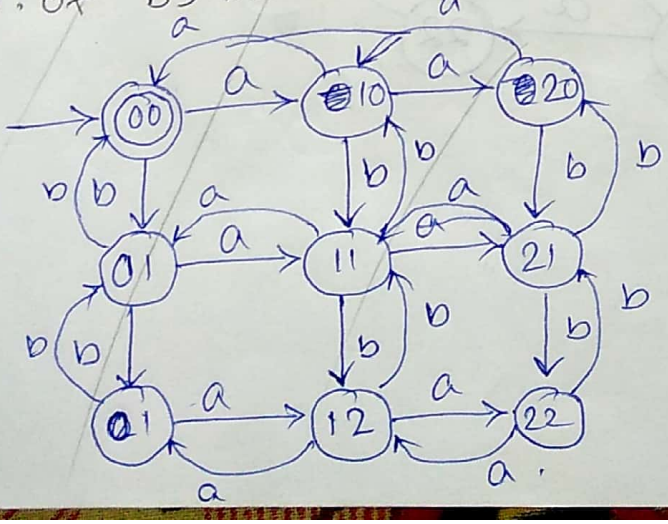




Q: No. of a's mod 3 = 0.
&
No. of b's mod 2 = 0



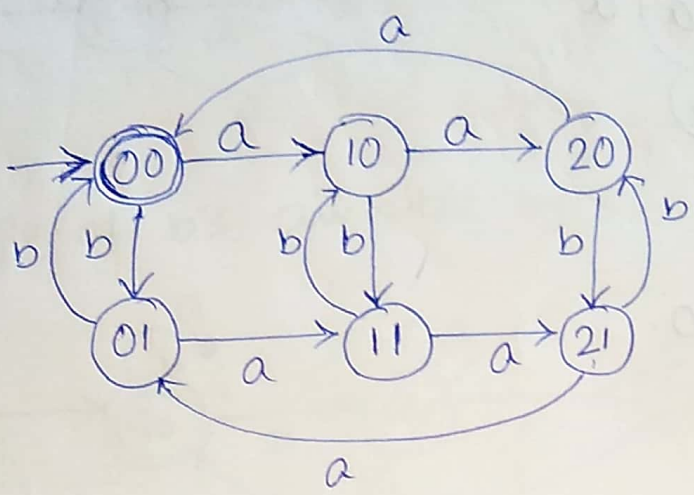
Q: No. of a's mod 3 = 0.
&
No. of b's mod 3 = 0



Q: No. of a's mod 3 = 0

&

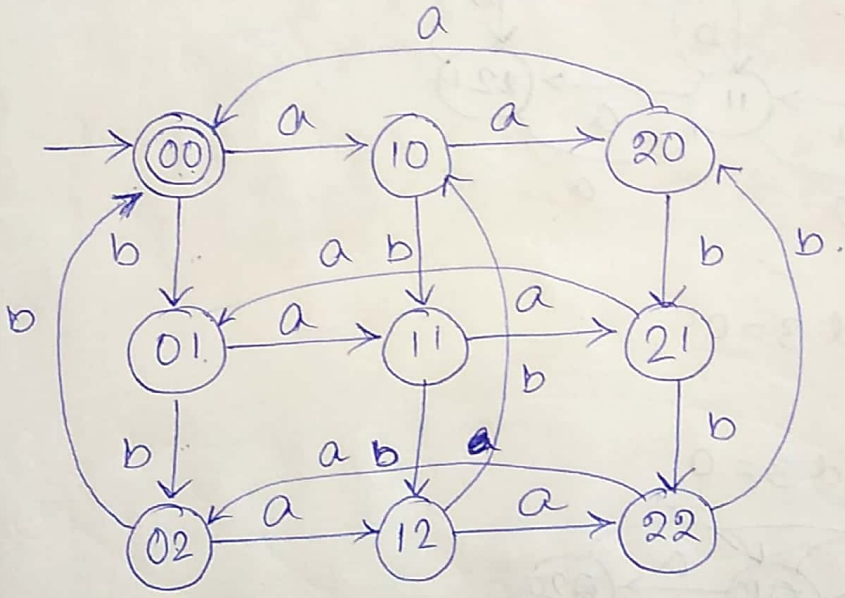
No. of b's mod 2 = 0



Q: No. of a's mod 3 = 0

&

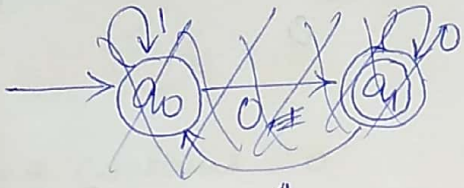
No. of b's mod 3 = 0

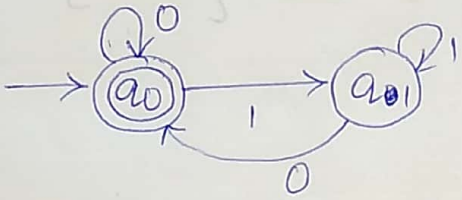


~~No. of a's divisible by 3 & No~~

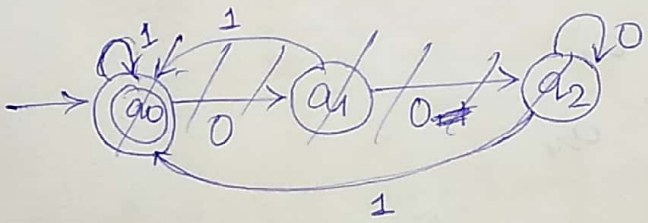
09101120

Q: Construct a DFA over {0,1} where ~~strings~~ the binary equivalent of string is divisible by 2.

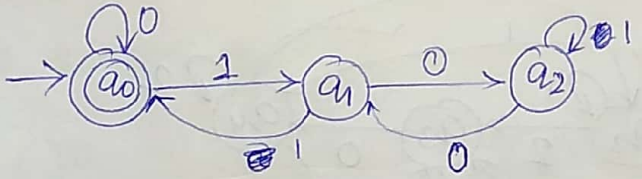




Q: Binary string divisible by 3.

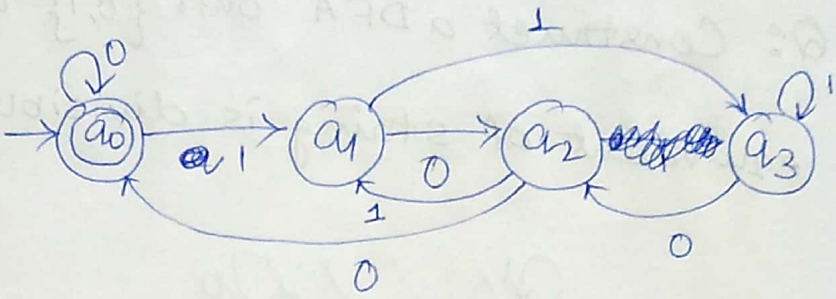


	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2



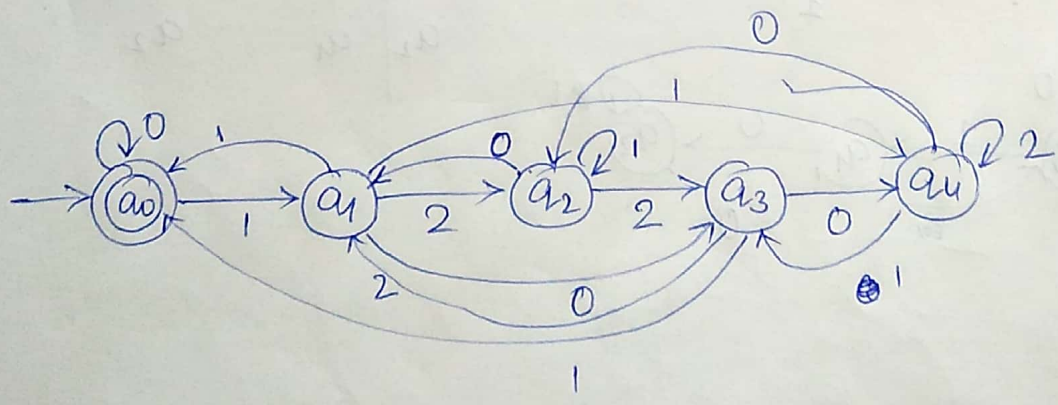
Q: Binary string divisible by 4

	0	1
→ a_0	a_0	a_1
a_1	a_2	a_3
a_2	a_0	a_1
a_3	a_2	a_3

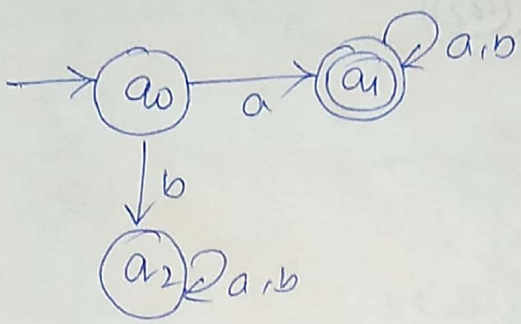


Q: ~~Binary string~~ A number divisible by 5 over $\{0,1,2\}$.

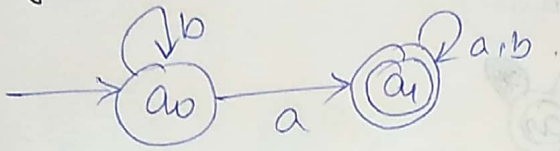
	0	1	2
→ a_0	a_0	a_1	a_2
a_1	a_3	a_4	a_0
a_2	a_1	a_2	a_3
a_3	a_4	a_0	a_1
a_4	a_2	a_3	a_4



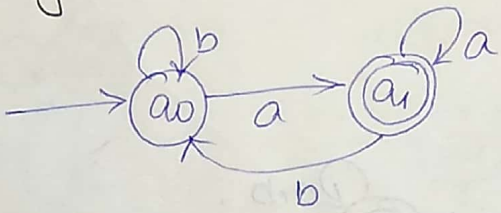
Q: Construct a DFA where the string always starts with 'a' over {a, b}



Q: String contains 'a'.

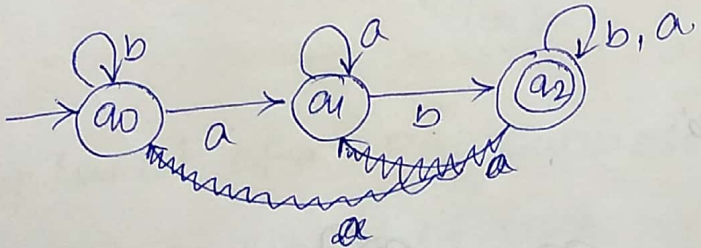


Q: String ends with 'a'.

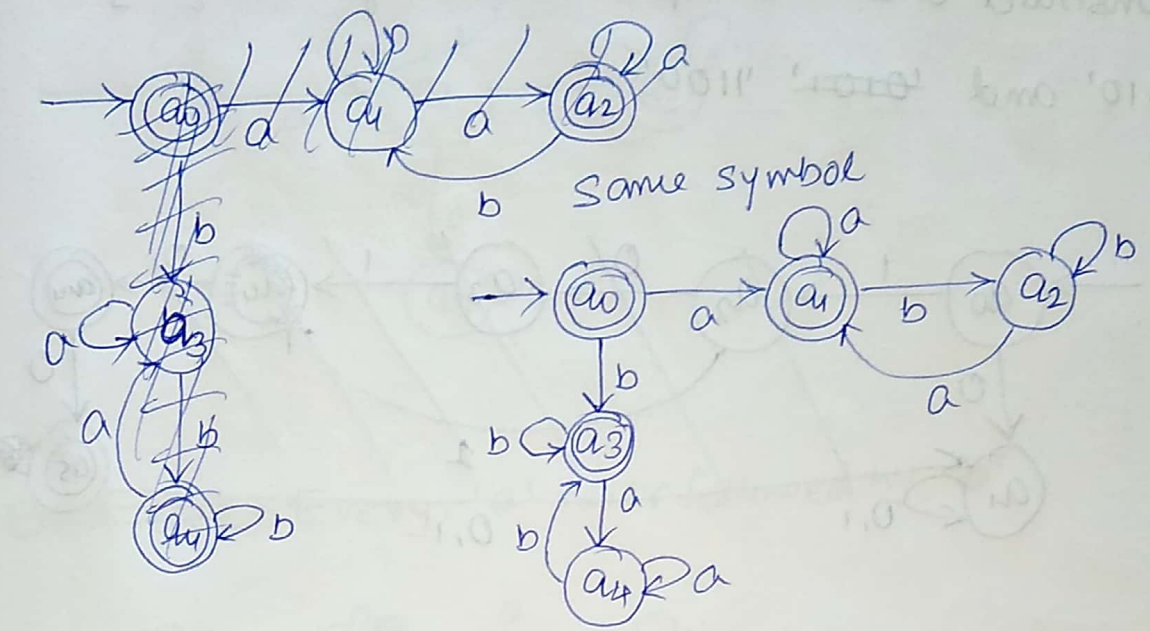


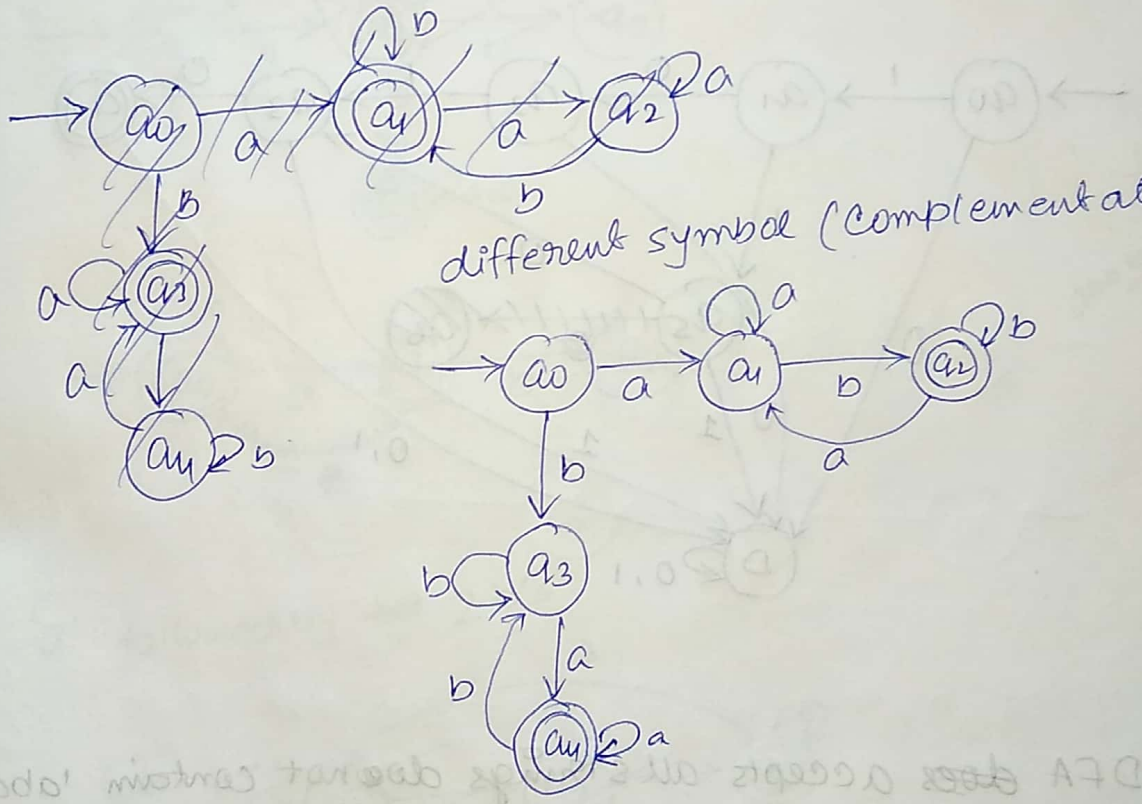
10/01/20

Q: Construct a DFA over {a, b} where strings contain 'ab' as substring.



Q: Starting and ending with diff. symbol.



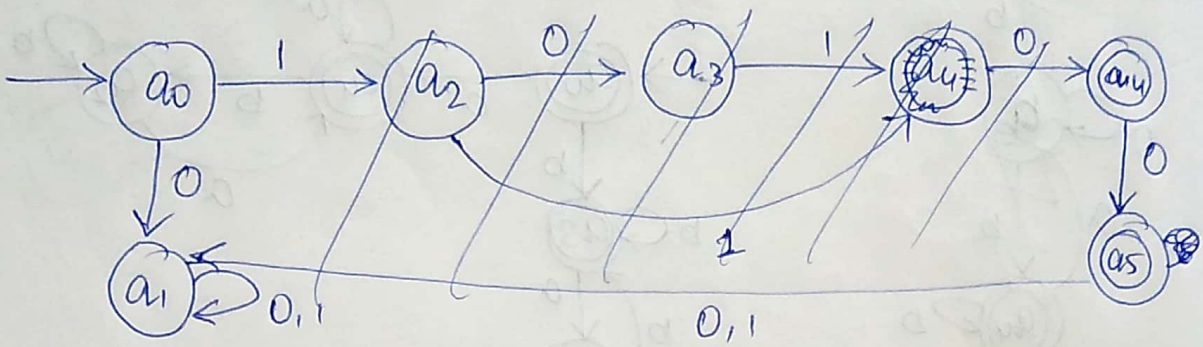


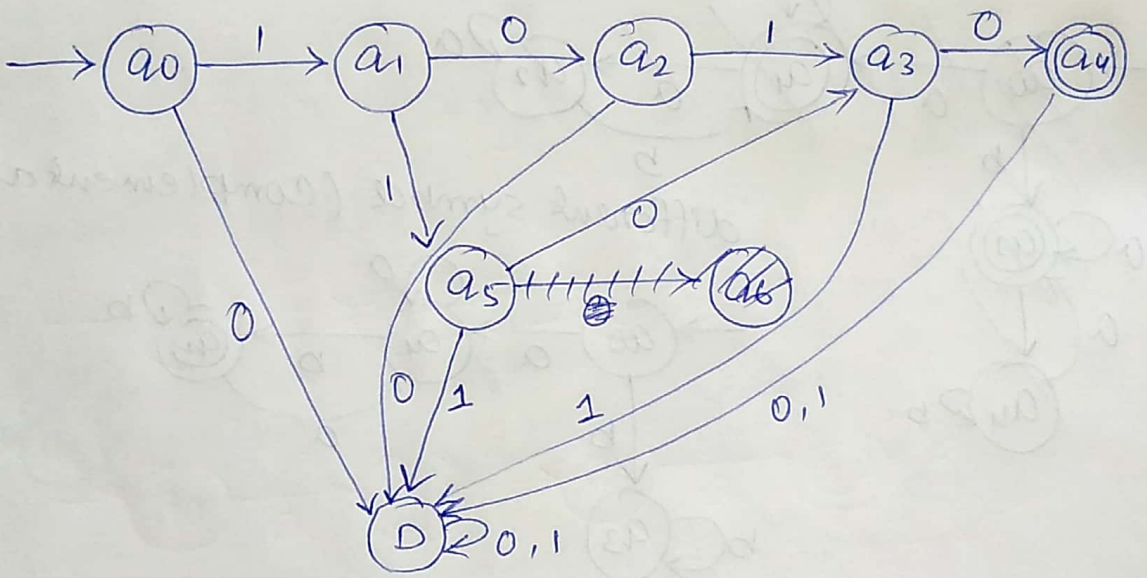
Complement of a DFA: Suppose M is a DFA and $\overline{M}$ is the complement of DFA.

(i) ~~Init~~ Final states should be non-final & non-final states should be final states.

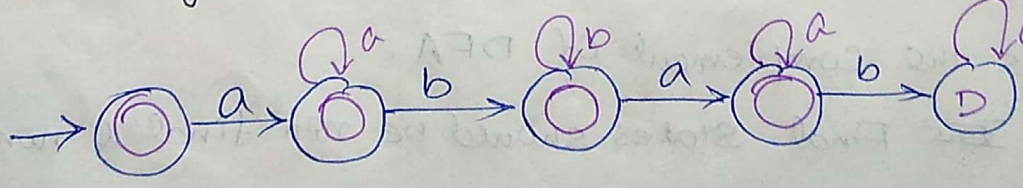
Q: Construct a DFA accepts the strings over {0,1}.

'1010' and ~~'0101'~~ '1100'.

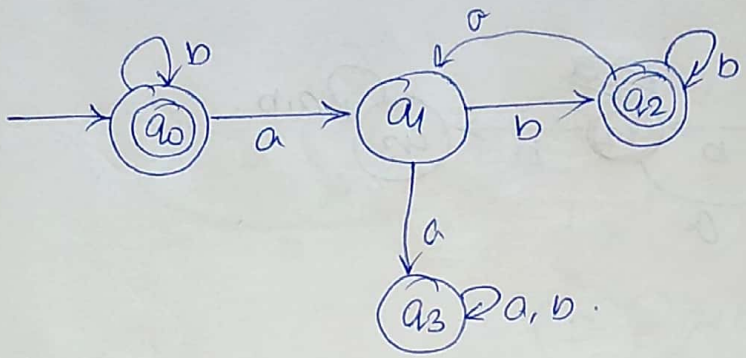




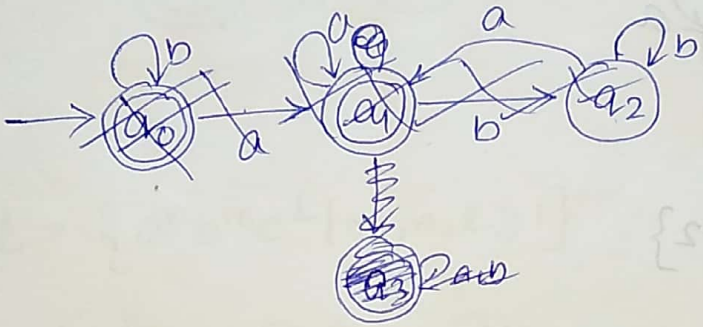
Q: DFA ~~does~~ accepts all strings do not contain 'abab' as substring.

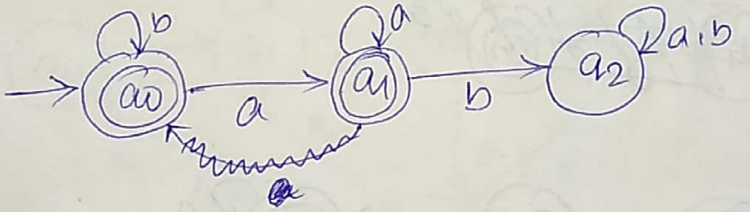


Q: DFA accepts strings every a followed by b.

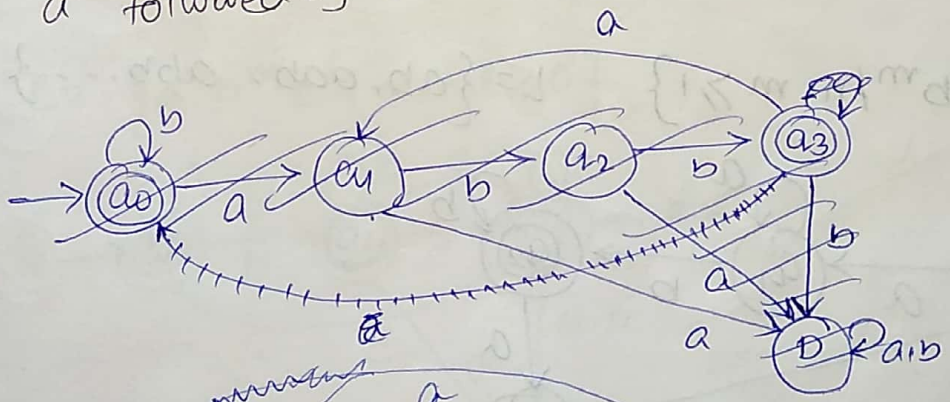


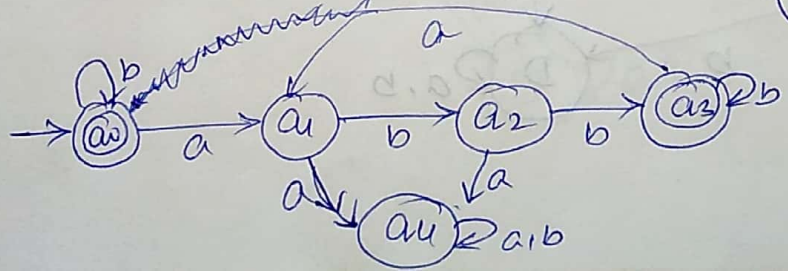
Q: ~~DFA accept~~ Every 'a' never followed by 'b'.



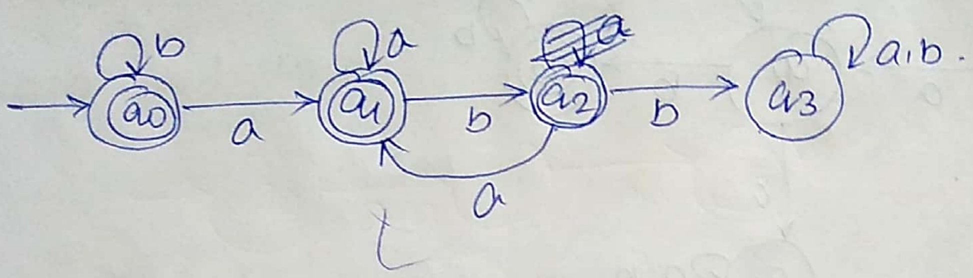


Q: 'a' followed by ~~'bb'~~ 2 b's



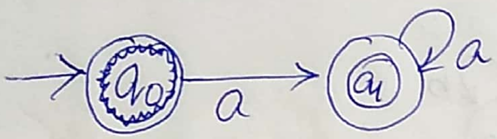


Q: a Never followed by 2 b's.

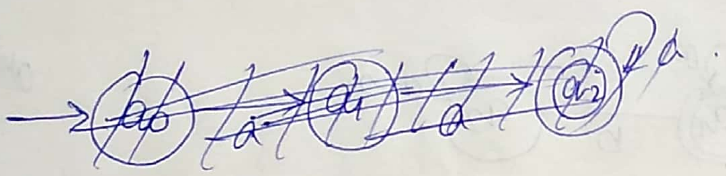


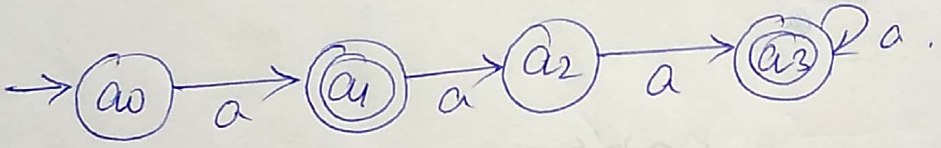
14/01/20

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

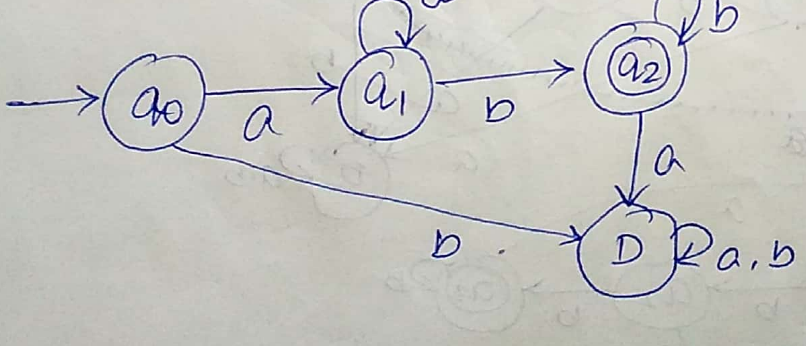


Q: $L = \{a^n,\ n \geq 1, n \neq 2\}$

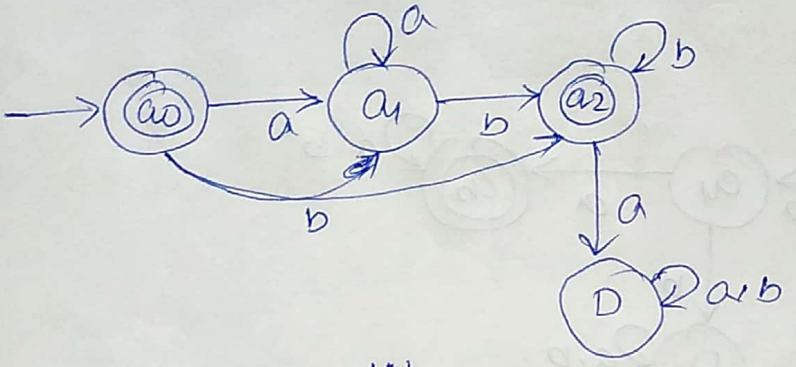




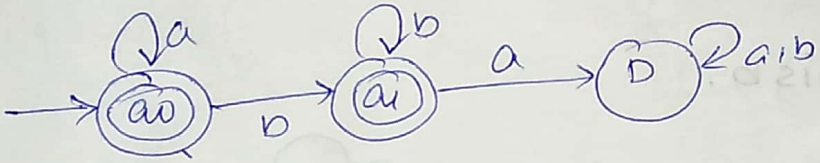
Q: $L = \{a^n b^m \mid n, m \geq 1\}$ $L = \{ab, aab, abb \ldots\}$.



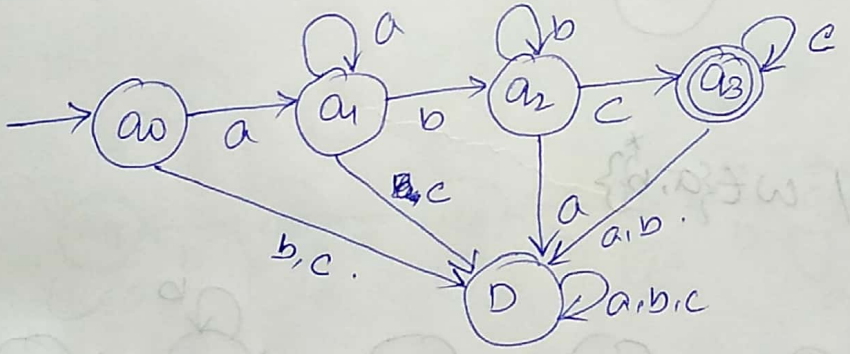
Q. $L = \{a^n b^m \mid m, n > 0\}$ $L = \{\epsilon, a, b, ab, aab, \cdots\}$.



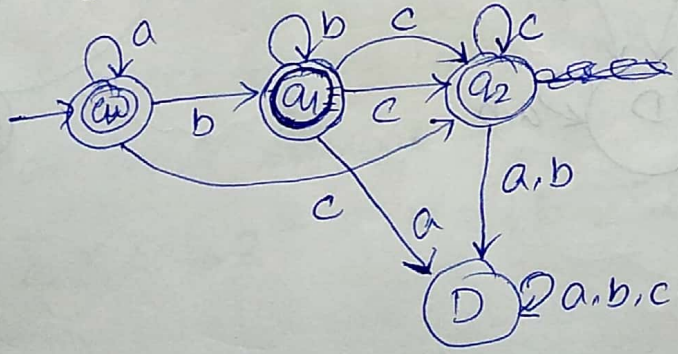
(or)



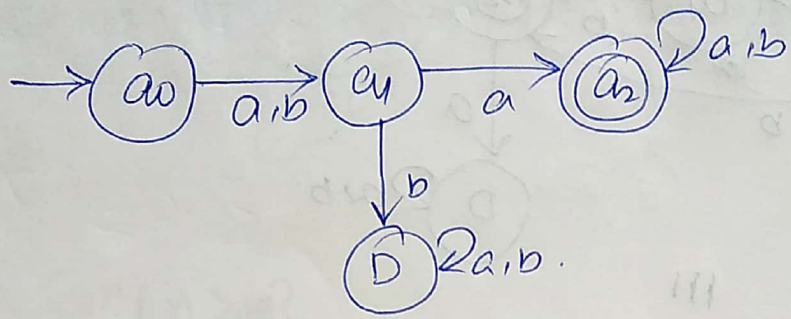
Q: $L = \{a^n b^m c^l \mid m, n, l \geq 1\}$



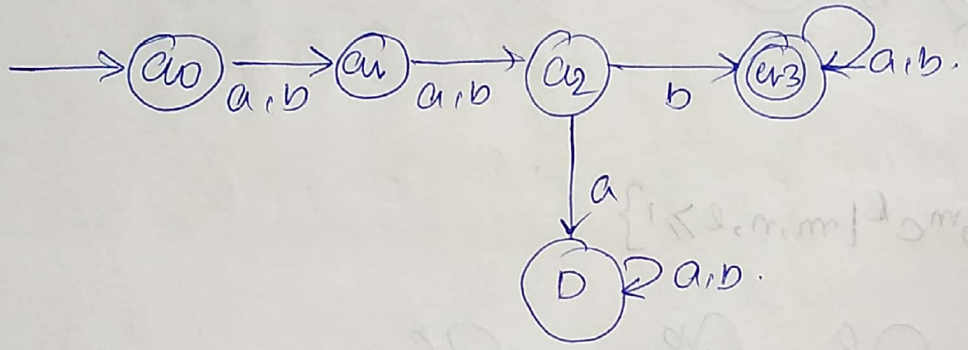
Q. $L = \{a^n b^m c^l \mid m, n, l \geq 0\}$



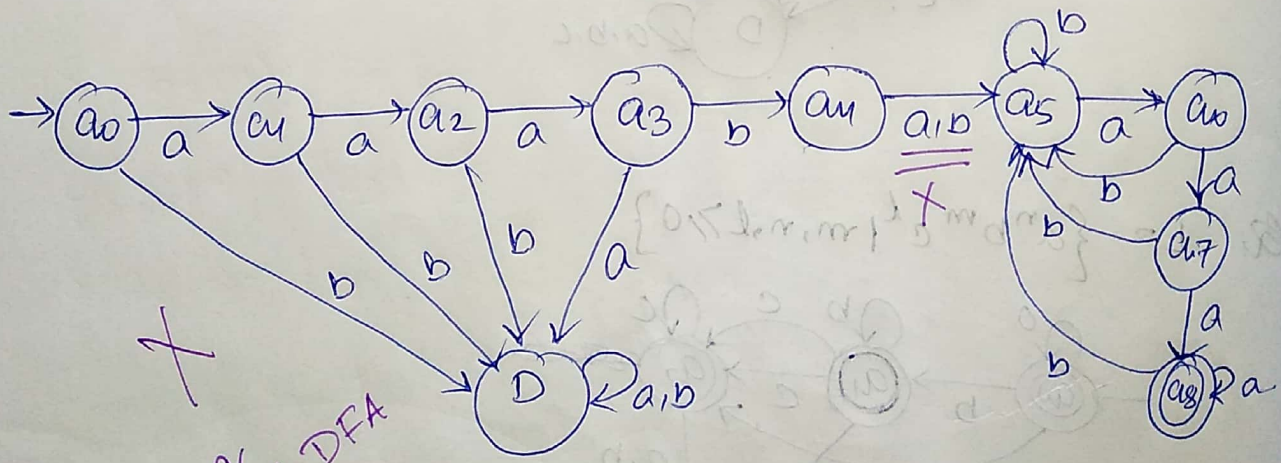
Q: DFA accepts all strings * where 2nd symbol is 'a'.



Q. 3rd symbol is b.



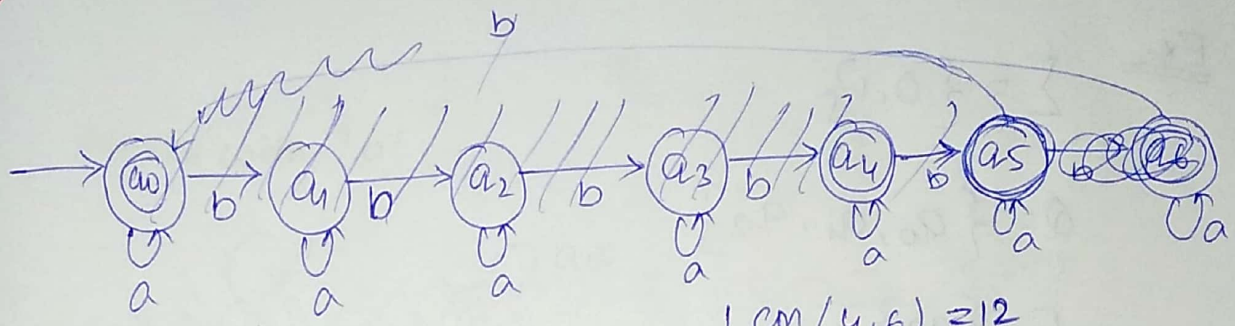
Q. $L = \{a^3 b w a^3 \mid w \in \{a,b\}^*\}$

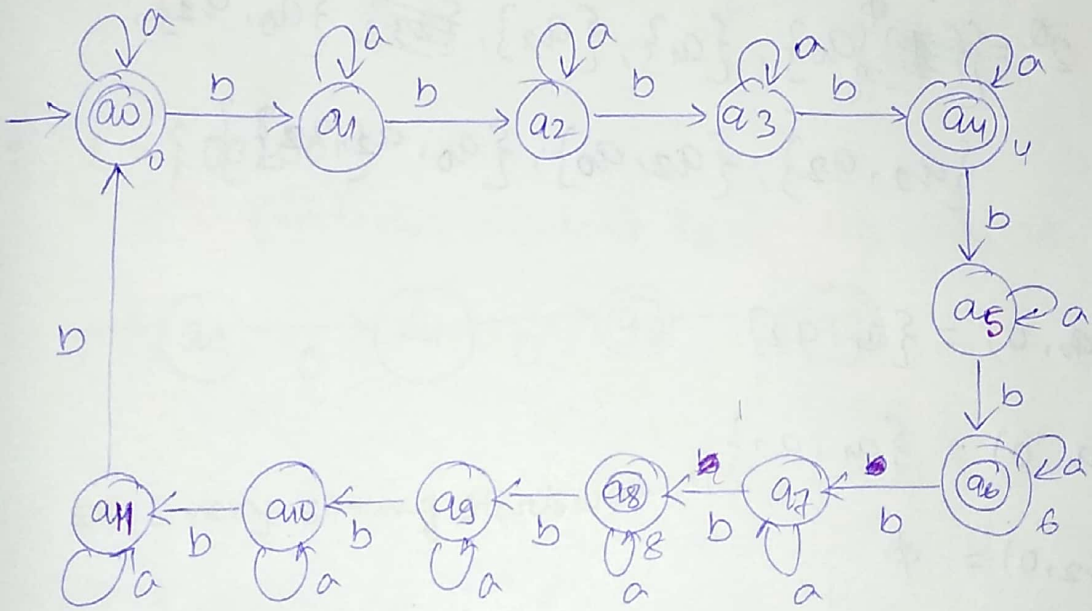


Ex: 2.1 Q: 1, 2, ~~3~~, 5, 6, 7.

16/01/20

Q: no. of b's divisible by 4 or 6.





NFA (Non-Deterministic Finite Automata):

The finite automata that can move to 0 or more no. of states for the input symbol is called NFA.

~~For an input symbol~~

$\delta: \Sigma \times Q \to 2^Q$ $\quad M = \{Q, \Sigma, \delta, q_0, F\}$

Difference between NFA & DFA:

(1) M is DFA iff $\delta(a_1, x)$ = state

(2) M is NFA iff $\delta(a_1, x)$ = Set of states

Ex. $\sum = \{0, 1\}$

$Q = \{a_0, a_1, a_2\}$.

$F = \{a_2\}$.

$2^Q = \{\phi, \{a_0\}, \{a_1\}, \{a_2\}, \{a_0, a_1\}, \{a_1, a_2\}, \{a_2, a_0\}, \{a_0, a_2, a_2\}\}$.

$\delta(a_0, 0) = \{a_1, a_2\}$.

$\delta(a_1, 0) = \{a_1, a_2\}$.

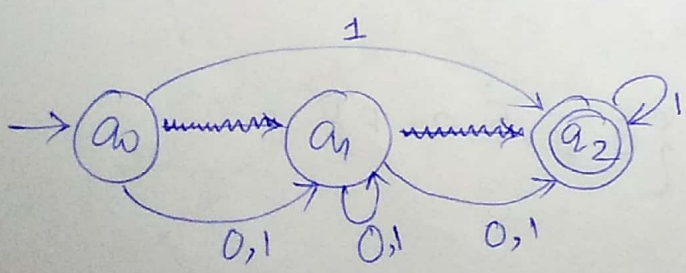
$\delta(a_2, 0) = \phi$

$\delta(a_0, 1) = \{a_1\}$.

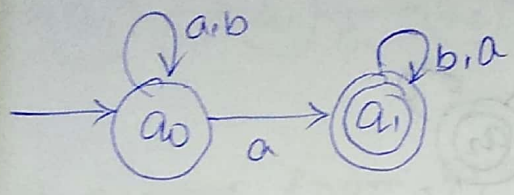
$\delta(a_1, 1) = \{a_2, a_1\}$.

$\delta(a_2, 1) = \{a_2\}$.

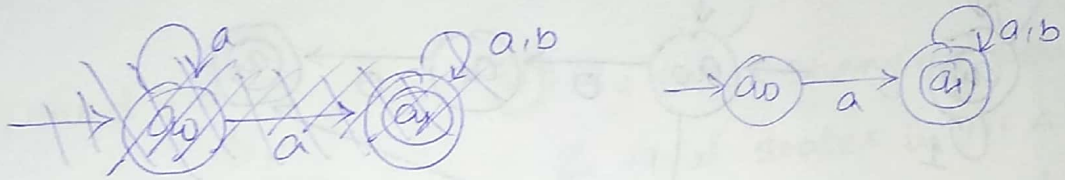
- ~~Every DFA can be converted into NFA & vice-versa.~~



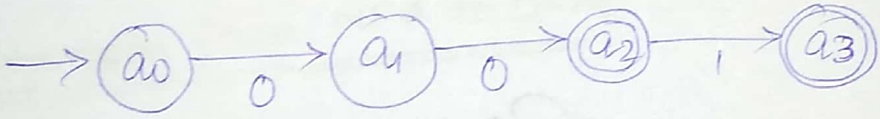
Q: NFA accepts all strings ~~end~~ contain 'a' as substring.



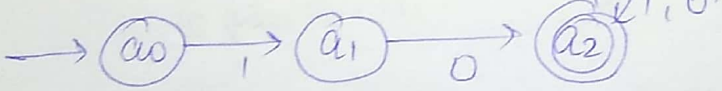
Q: Starts with 'a'



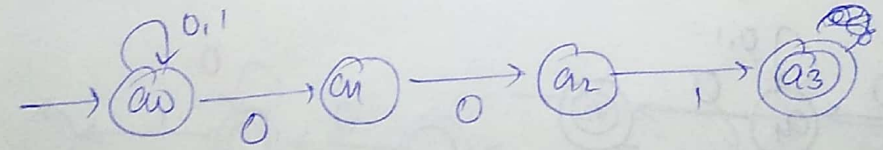
Q: $L = \{00, 001\}$



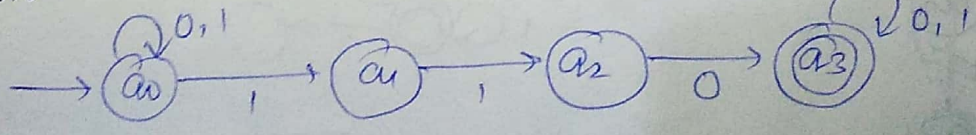
Q: ~~Ex~~ Every string starts with '10'



Q: ends with '001'

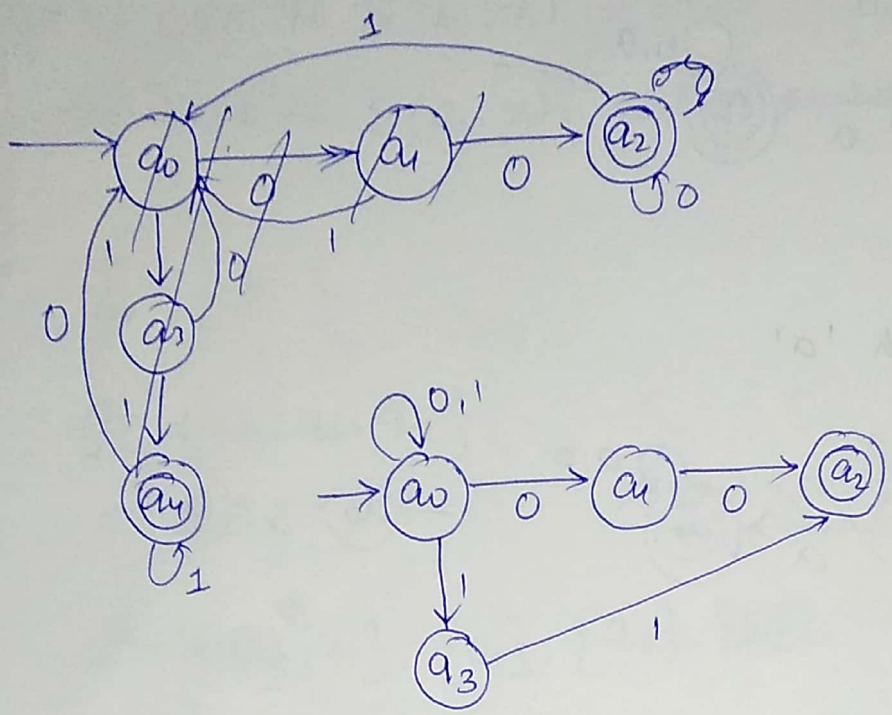


Q: Contains '110'

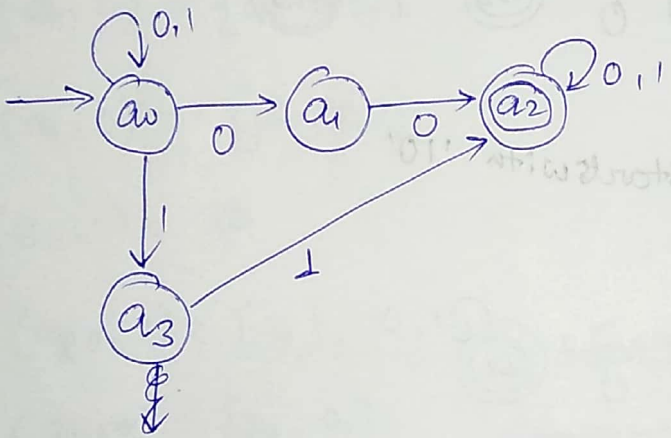


17/01/20

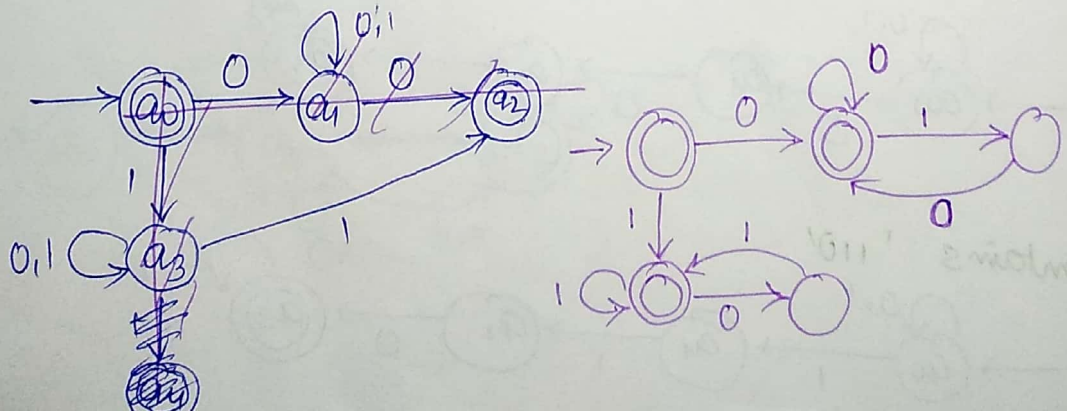
Q: NFA accepts all strings contains last 2 symbols are '00' or '11'



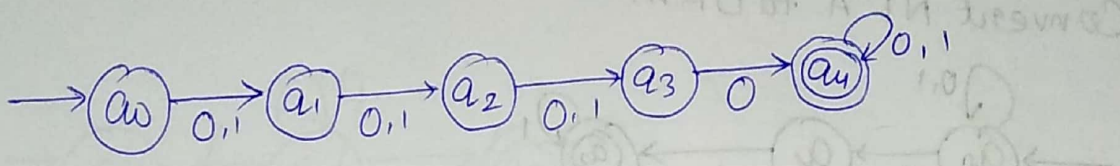
Q: NFA accepts all strings contains substring '00' or '11'.



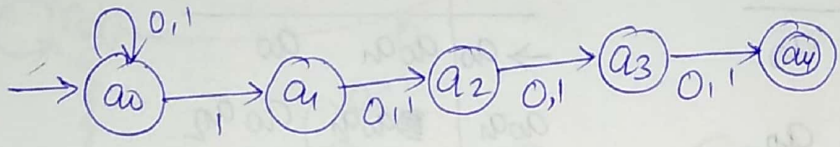
Q: Starting & ending with same symbol.



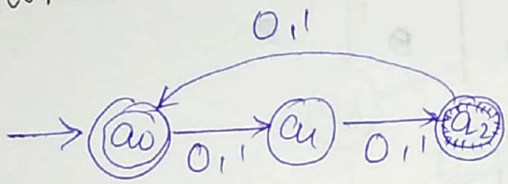
Q: 4th symbol from LHS is 0.



Q: 4th symbol from RHS is 1.



Q: $|w| \bmod 3 = 0$.



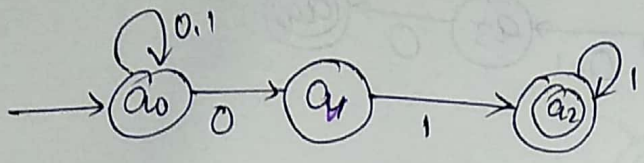
(*) By interchanging Non-final & final states in NFA, we will not get the complement of the language.

~~Q:~~

Conversion from NFA to DFA:

1. The initial state of NFA and its equivalent DFA is same.
2. Transition Function:
$$\delta'(q_0, q_1, q_2, \ldots q_n, n) = \delta(q_0, n) \cup \delta(q_1, n) \cup \delta(q_2, n), \ldots \delta(q_n, n)$$
3. Final State: Every subset which contains the final state of NFA is the final state in DFA.

Q: Convert NFA to DFA.

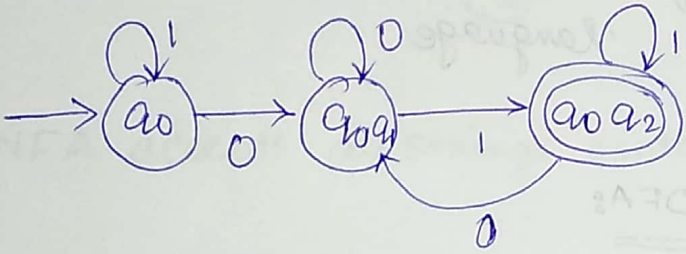


NFA:

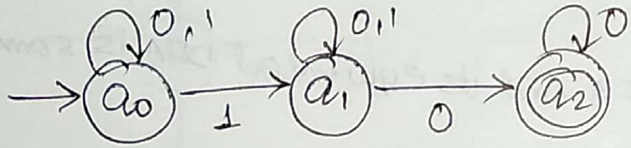
	0	1
→ q_0	q_0, q_1	q_0
q_1	ϕ	q_2
q_2	ϕ	q_2

DFA:

	0	1
→ q_0	q_0q_1	q_0
q_0q_1	q_0q_1	q_0q_2
q_0q_2	q_0q_1	q_0q_2



Q:

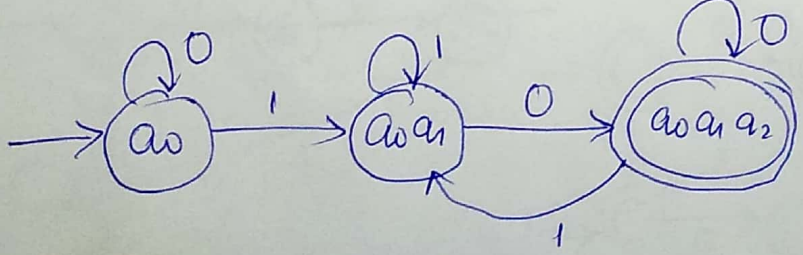


NFA

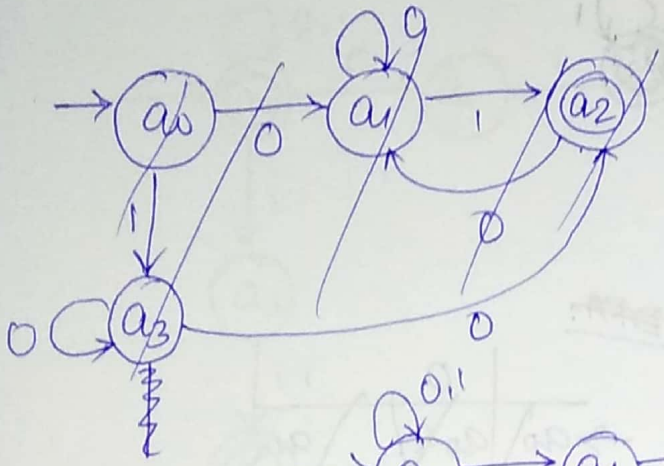
	0	1
→ q_0	q_0	q_0, q_1
q_1	q_1, q_2	q_1
q_2	q_2	ϕ

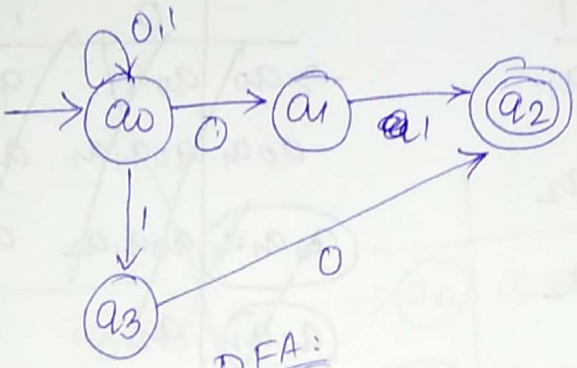
DFA

	0	1
→ q_0	q_0	q_0q_1
q_0q_1	$q_0q_1q_2$	q_0q_1
$q_0q_1q_2$	$q_0q_1q_2$	q_0q_1



Q: NFA.... last ~~1st and~~ last 2 symbols are different.



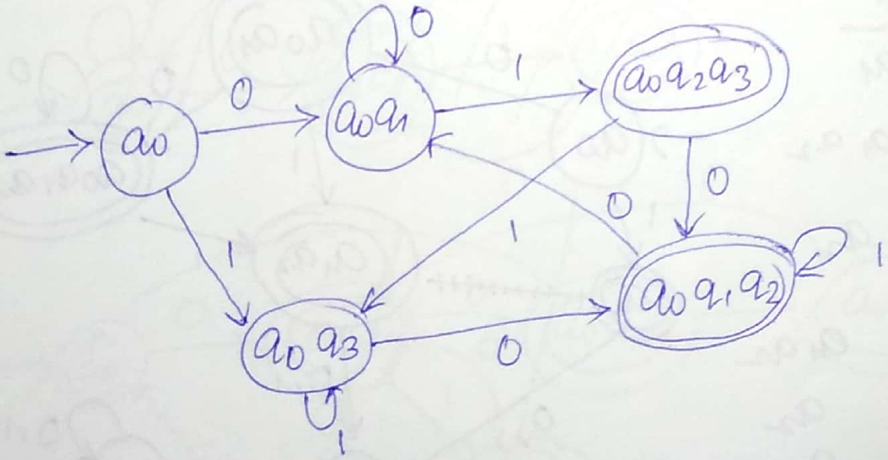


NFA:

	0	1
$\to q_0$	q_0, q_1	q_0, q_3
q_1	ϕ	q_2
(q_2)	ϕ	ϕ
q_3	q_2	ϕ

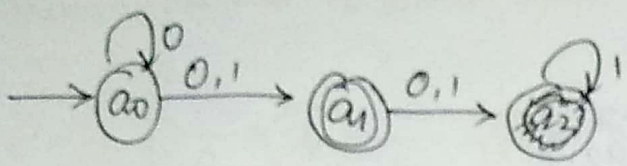
DFA:

	0	1
$\to q_0$	$q_0 q_1$	$q_0 q_3$
$q_0 q_1$	$q_0 q_1$	$q_0 q_2 q_3$
$q_0 q_3$	$q_0 q_1 q_2$	$q_0 q_3$
$(q_0 q_2 q_3)$	$q_0 q_1 q_2$	$q_0 q_3$
$(q_0 q_1 q_2)$	$q_0 q_1$	$q_0 q_2 q_3$



20/01/20

Q: Convert the NFA to DFA.

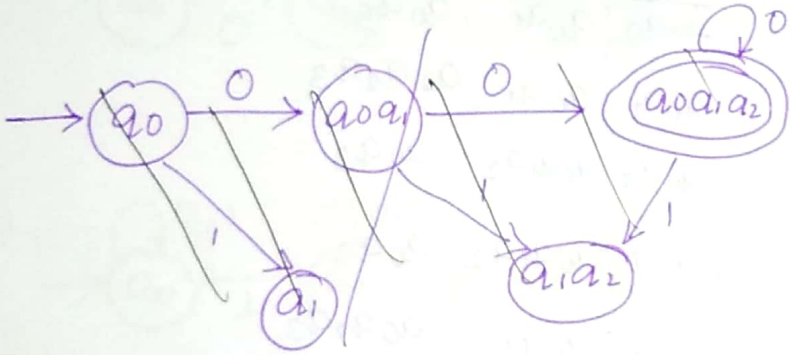


NFA:

	0	1
→ a_0	a_0, a_1	a_1
a_1	a_2	a_2
a_2	ϕ	a_2

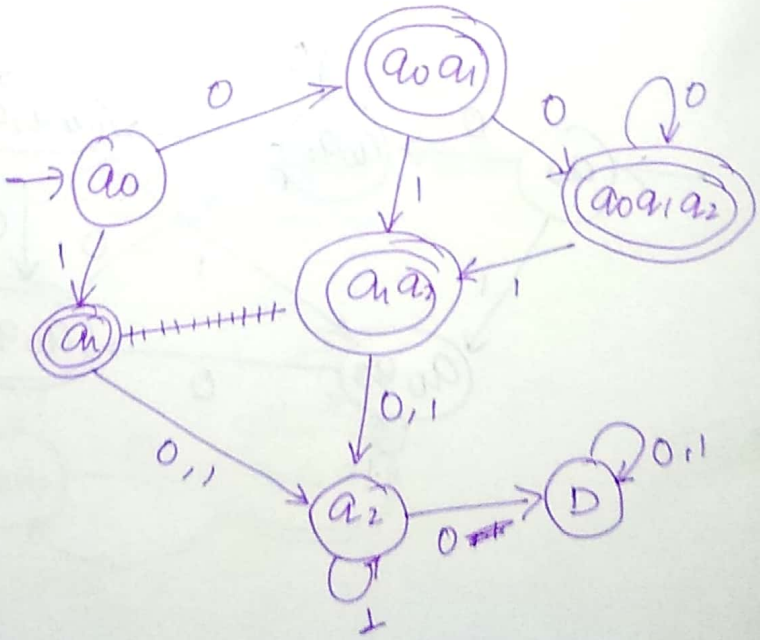
~~DFA:~~

	0	1
→ a_0	$a_0 a_1$	a_1
$a_0 a_1$	$a_0 a_1 a_2$	$a_1 a_2$
$a_0 a_1 a_2$	$a_0 a_1 a_2$	$a_1 a_2$
$a_1 a_2$	a_2	a_2
a_2		

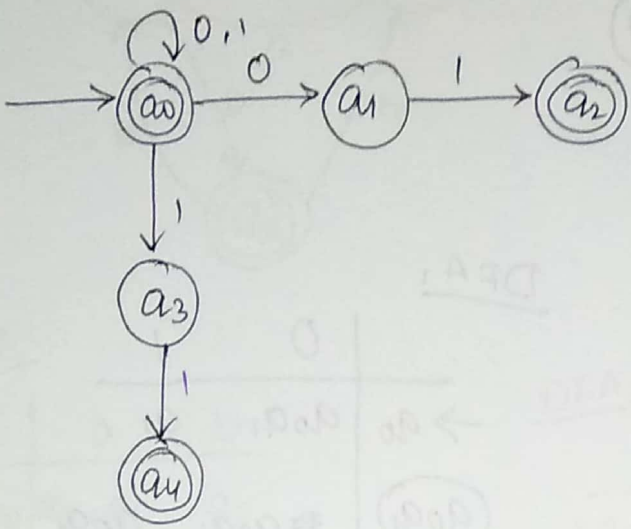


DFA:

	0	1
→ a_0	$a_0 a_1$	a_1
$a_0 a_1$	$a_0 a_1 a_2$	$a_1 a_2$
$a_1 a_2$	a_2	a_2
$a_0 a_1 a_2$	$a_0 a_1 a_2$	$a_1 a_2$
a_1	a_2	a_2
a_2	D	a_2
D	D	D

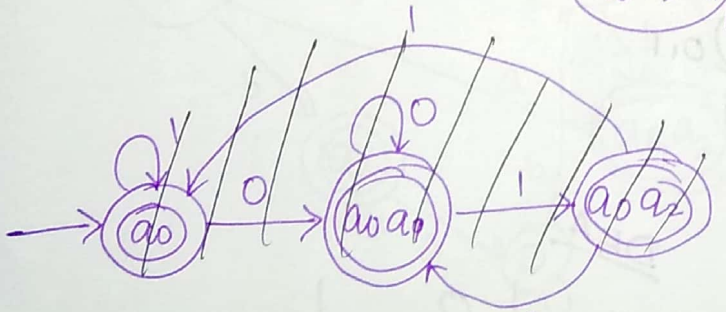


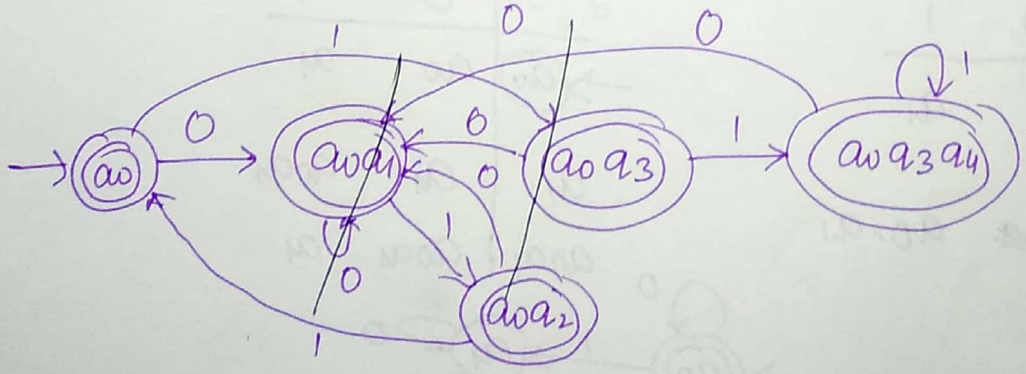
Q:



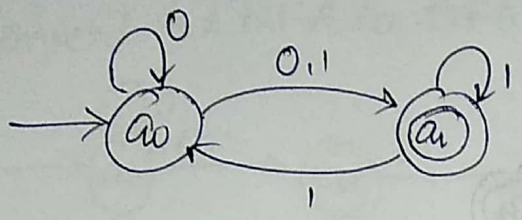
	0	1
→ a_0	a_0, a_1	a_0, a_3
a_1	ϕ	a_2
a_2	ϕ	ϕ
a_3	ϕ	q_4
a_4	ϕ	ϕ

	0	1
→ a_0	a_0a_1	a_0a_3
a_0a_1	a_0a_1	$a_0a_2a_3$
a_0a_2	a_0a_1	a_0 ← This is wrong. Correct Ownself
$a_2a_0a_3$	a_0a_1	$a_0a_3a_4$
a_4		
$a_0a_3a_4$	a_0a_1	$a_0a_3a_4$





Q:

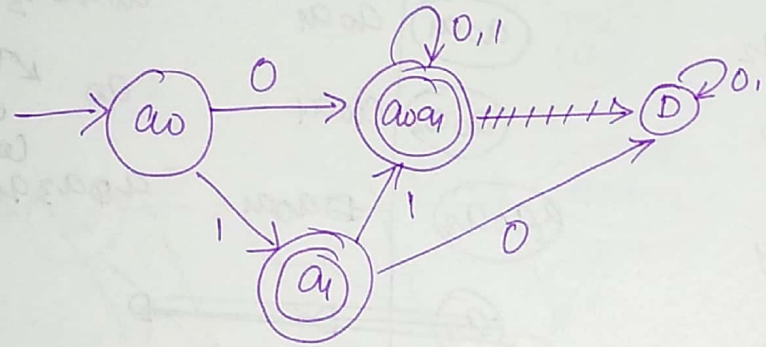


NFA:

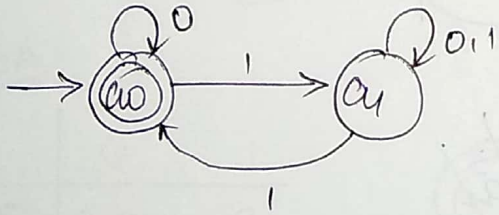
	0	1
→ q_0	q_0, q_1	q_1
(q_1)	ϕ	q_0, q_1

DFA:

	0	1
→ q_0	q_0q_1	q_1
(q_0q_1)	q_0q_1	q_0q_1
(q_1)	D	q_0q_1
D	D	D



Q:

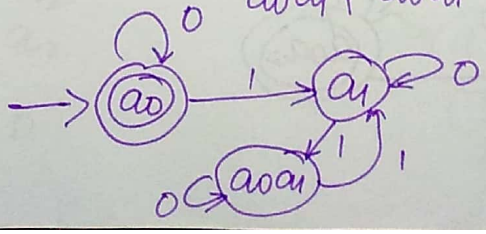


NEA

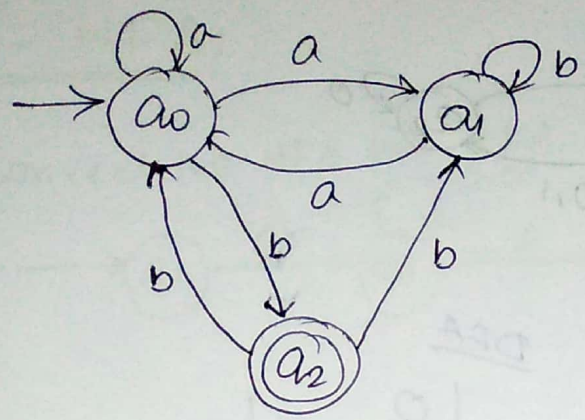
	0	1
→ (q_0)	q_0	q_1
q_1	q_1, ~~ϕ~~	q_0, q_1

DEA

	0	1
→ (q_0)	q_0	q_1
q_1	q_1	q_0q_1
q_0q_1	q_0q_1	q_1



Q:

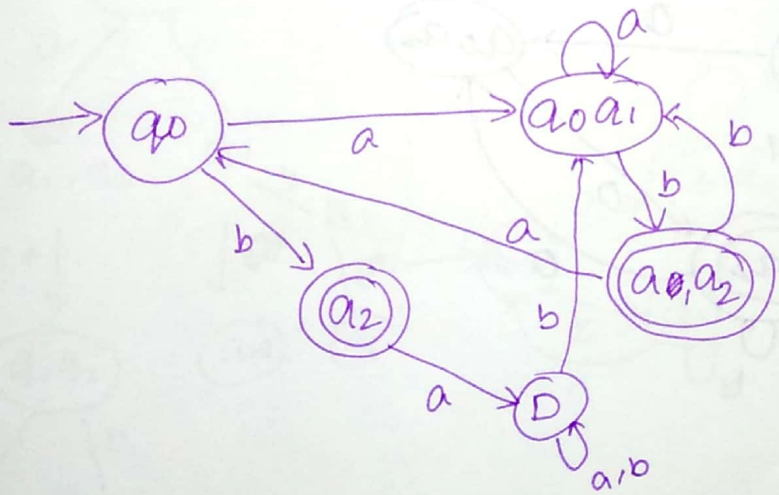


NFA:

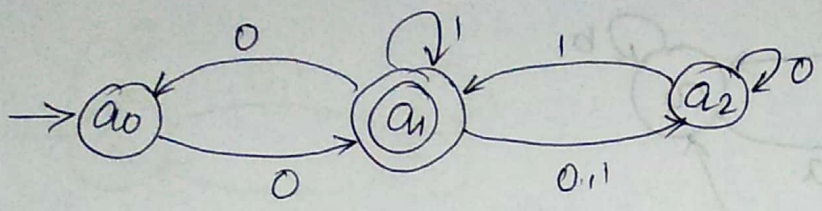
	a	b
$\rightarrow a_0$	a_0, a_1	a_2
~~$a_0 a_1$~~		
a_1	a_0	a_1
(a_2)	ϕ	a_0, a_1

DFA:

	a	b
$\rightarrow a_0$	$a_0 a_1$	a_2
$a_0 a_1$	$a_0 a_1$	$a_1 a_2$
$(a_1 a_2)$	a_0	$a_0 a_1$
(a_2)	D	$a_0 a_1$
D	D	D



Q:

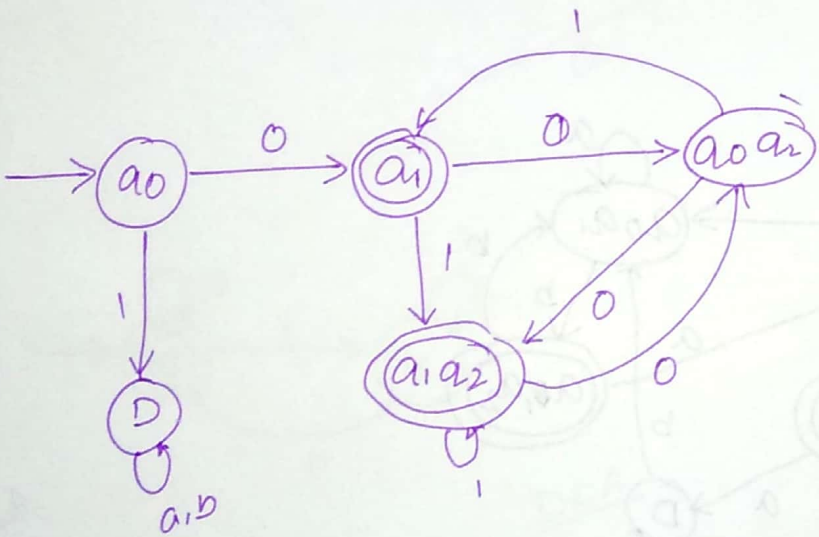


NFA:

	0	1
→ a_0	a_1	ϕ
a_1	a_0, a_2	a_1, a_2
a_2	a_2	a_1

DFA

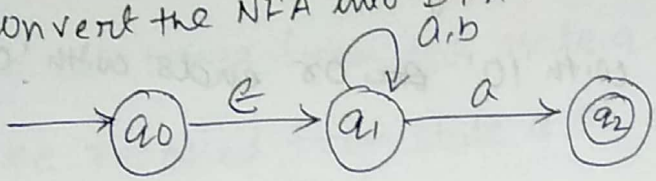
	0	1
→ a_0	a_1	D
a_1	$a_0 a_2$	$a_1 a_2$
$a_0 a_2$	$a_1 a_2$	a_1
$a_1 a_2$	$a_0 a_2$	$a_1 a_2$
D	D	D



21/01/20

ϵ - NFA:

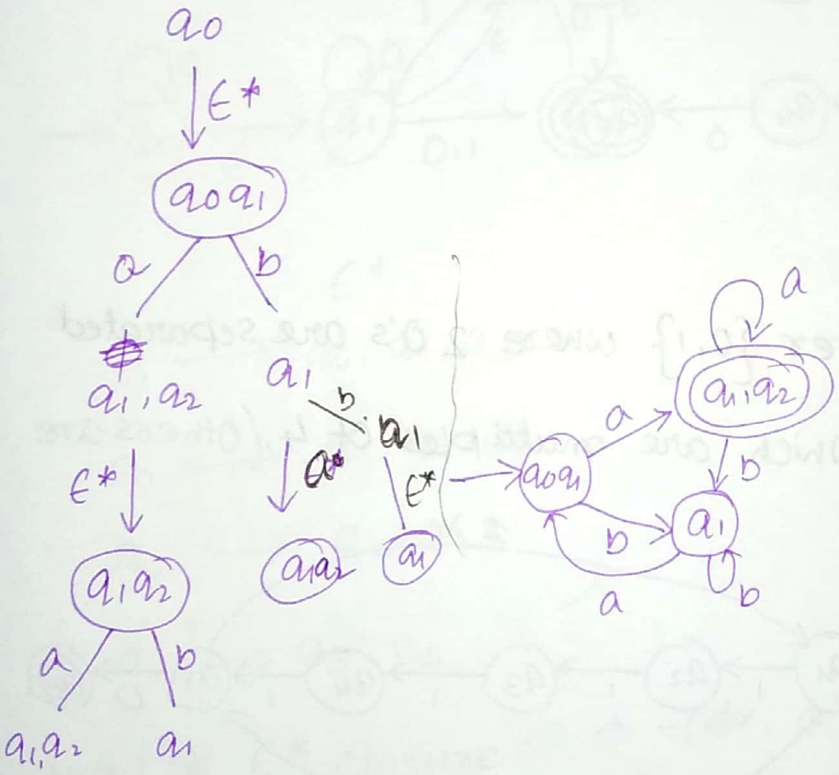
Q: Convert the NFA into DFA.



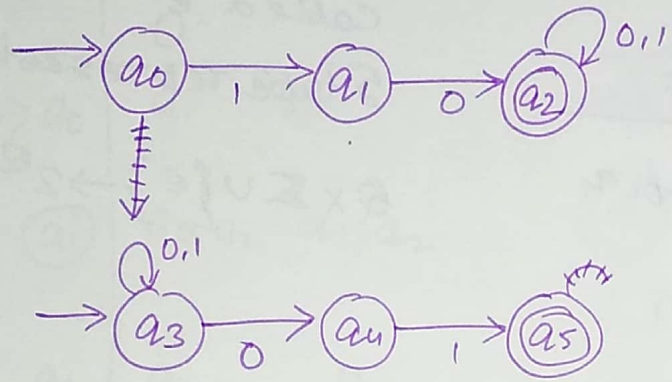
ϵ-NFA: An NFA that allows empty moves is called ϵ- NFA, is a 5 tuple representation:

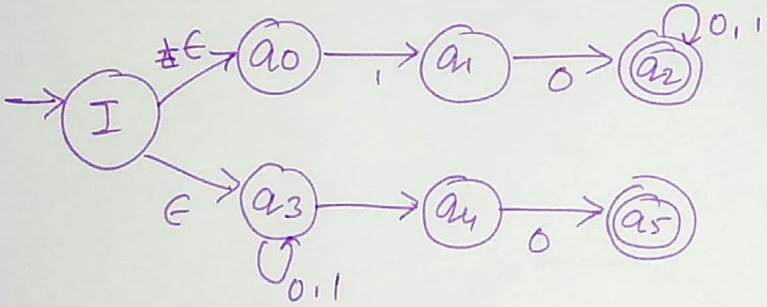
$Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

	a	b	ϵ
→ q_0	ϕ	ϕ	q_0, q_1
q_1	q_1, q_2	q_1	q_1
q_2	ϕ	ϕ	q_2

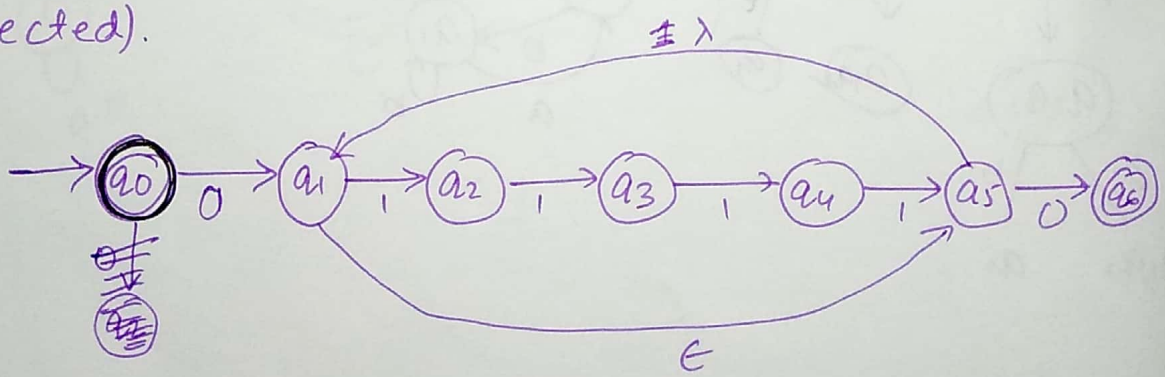


Q: ϵ-NFA.... set of all strings of 0's and 1's where string starts with '10' ~~and~~ or ends with '01'.



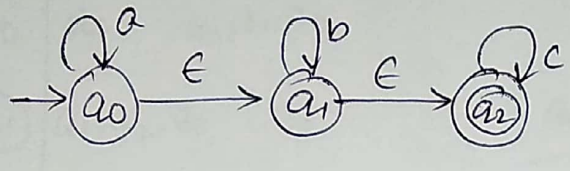


Q: ϵ-NFA ..... Over $\{0,1\}$ where 2 0's are separated by no. of 1's which are multiples of 4. (Others are rejected).



q_0 is not final.

ϵ^* (ϵ*) Closure: Set of all states which are in non-zero distance from the state q. Set of all the states that can be reached from state q on empty string (ϵ).

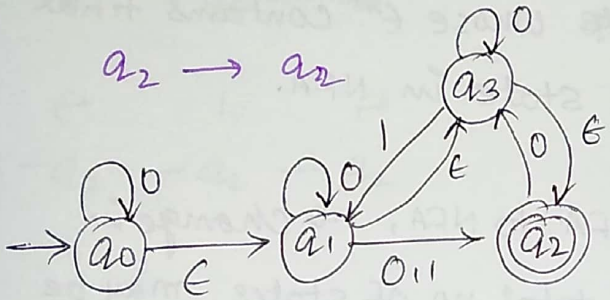


ϵ^*

$q_0 \longrightarrow q_0, q_1, q_2$

$q_1 \longrightarrow q_1, q_2$

$q_2 \longrightarrow q_2$



ϵ^*

$q_0 \longrightarrow q_0, q_1, q_3, q_2$

$q_1 \longrightarrow q_1, q_2, q_3.$

$q_2 \longrightarrow q_2$

$q_3 \longrightarrow q_3, q_2$

N.B: (1) ∄ ϵ^* closure of $\phi = \epsilon$, $\phi^* = \epsilon$

ϵ^* closure of $\epsilon = \epsilon$, $\epsilon^* = \epsilon$

(2) ∄ ϵ closure of $(x, y, z) = \epsilon^*(x) \cup \epsilon^*(y) \cup \epsilon^*(z)$

Conversion of ϵ-NFA to NFA:

(2) Transition Function:

$\epsilon^*()$

↓

$\delta()$

↓

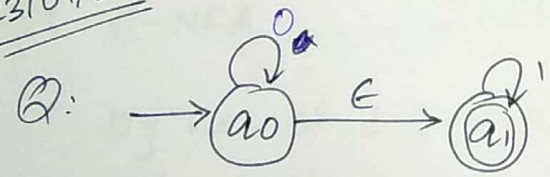
$\epsilon^*()$

(3) Final State: Every state whose ϵ^* contains final state of ϵ-NFA is a final state in NFA.

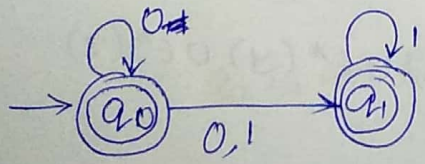
N.B: While converting ϵ-NFA to NFA, no change in initial state, no change in total no. of states, may be change in final states.

23/01/20

Q:



	0	1
→ q_0	q_0, q_1	q_1
q_1	ϕ	q_1



ϵ^* 0 ϵ^*

q_0 — q_0 — q_0 — q_0, q_1

q_1 — ϕ — ϕ

ϵ^* 1 ϵ^*

q_0 — q_0 — ϕ — ϕ

q_1 — q_1 — q_1

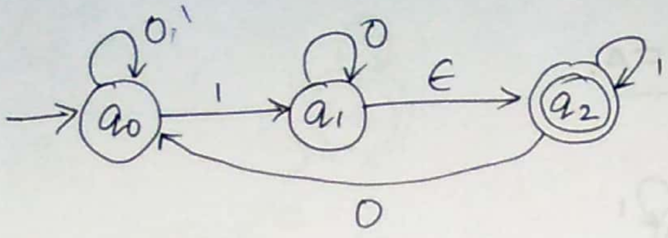
ϵ^* 0 ϵ^*

q_1 — q_1 — ϕ — ϕ

ϵ^* 1 ϵ^*

q_1 — q_1 — q_1 — q_1

Q:



	0	1
→ a_0	a_0	a_0, a_1, a_2
(a_1)	a_1, a_2, a_0	a_2
(a_2)	a_0	a_2

$a_0 - a_0 \xrightarrow{\epsilon^*} - a_0 \xrightarrow{0} - a_0 \xrightarrow{\epsilon^*}$

i.e. ϵ^*, 0, ϵ^*: $a_0 - a_0 — a_0 — a_0$

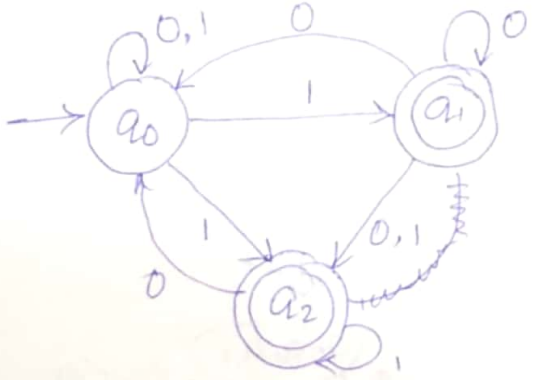
ϵ^*, 1, ϵ^*:
$a_0 - a_0 — a_1 — a_1, a_2$
$\quad \searrow a_0 — a_0 — a_0$

ϵ^*, 0, ϵ^*:
$a_1 - a_1 — a_1 — a_1, a_2$
$\quad \searrow a_2 — a_0 — a_0$

ϵ^*, 1, ϵ^*:
$a_1 - a_1 — \phi — \phi$
$\quad \searrow a_2 — a_2 — a_2$

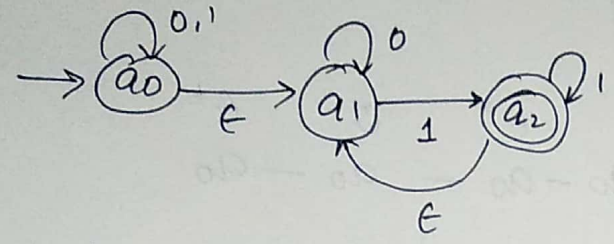
ϵ^*, 0, ϵ^*:
$a_2 - a_2 — a_0 — a_0$

ϵ^*, 1, ϵ^*:
$a_2 - a_2 — a_2 — a_2$

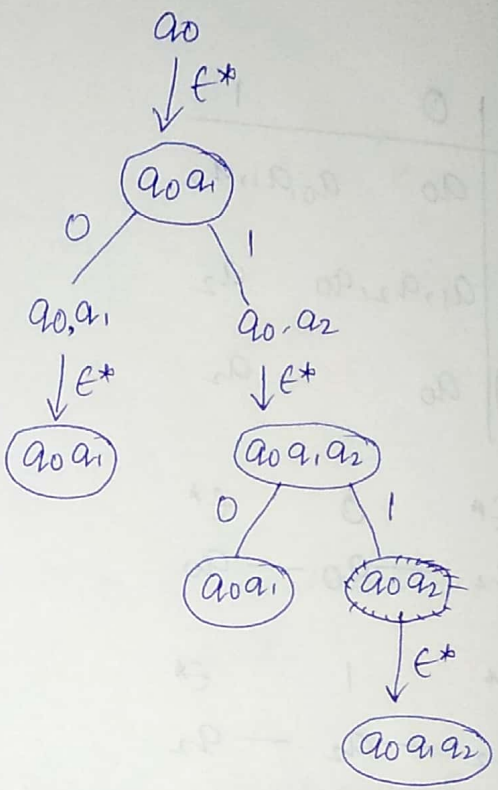


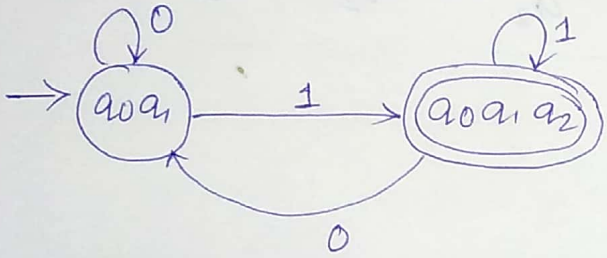
Conversion of ϵ NFA to DFA:

Q:

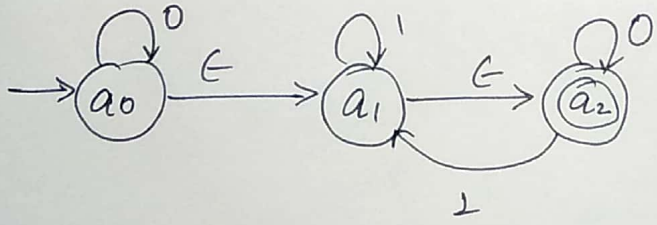


	0	1	ϵ*
→ q_0	q_0	q_0	q_0, q_1
q_1	q_1	q_2	q_1
q_2	ϕ	q_2	q_2, q_1

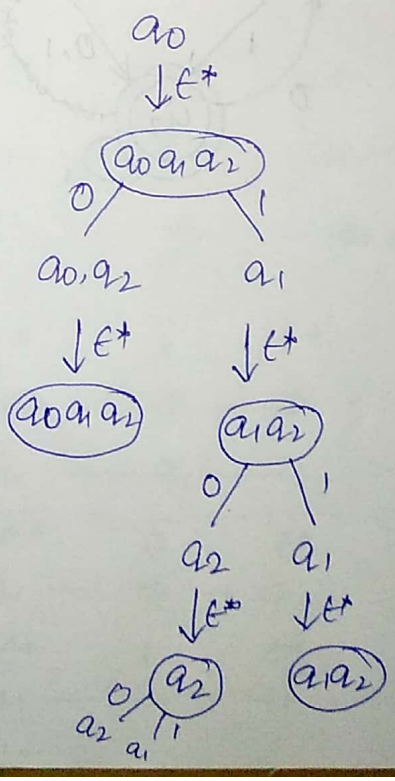


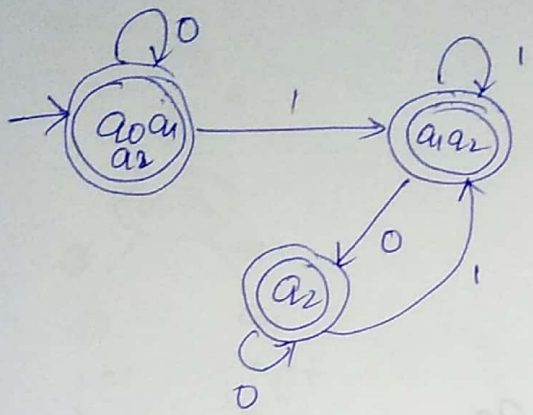


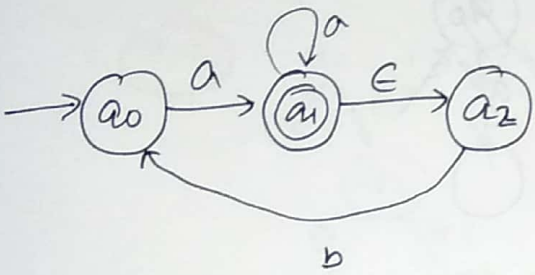
Q:



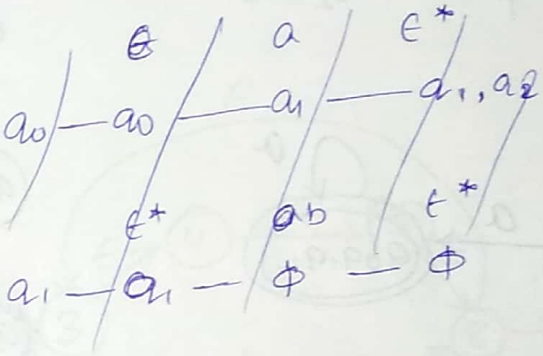
	0	1	ϵ*
→ q_0	q_0	ϕ	q_0, q_1, q_2
q_1	ϕ	q_1	q_1, q_2
q_2	q_2	q_1	q_2.

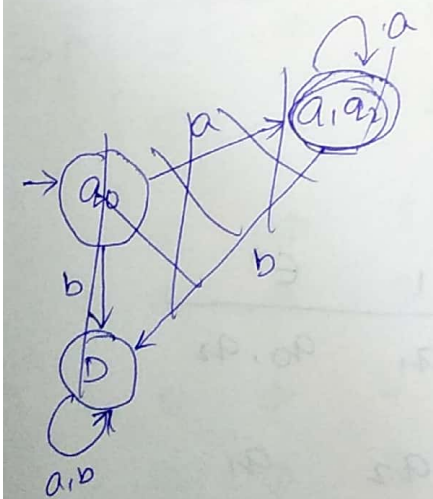


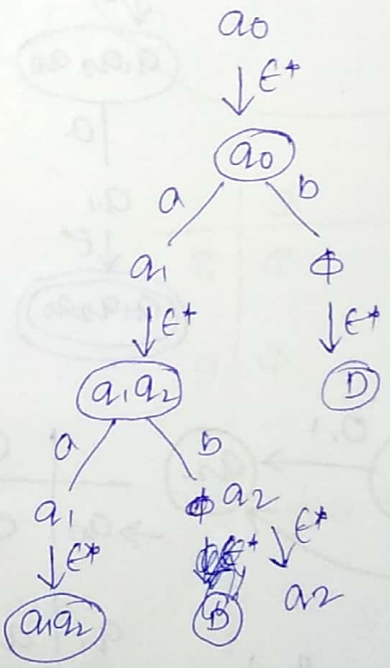




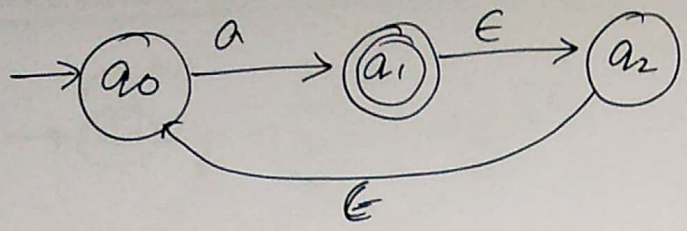
	a	b	ϵ^*
a_0	a_1	ϕ	a_0
a_1	a_1	ϕ	a_1, a_2
a_2	ϕ	a_0	a_2



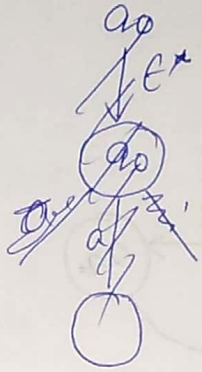


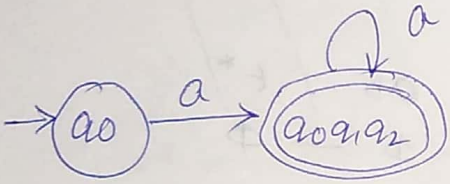


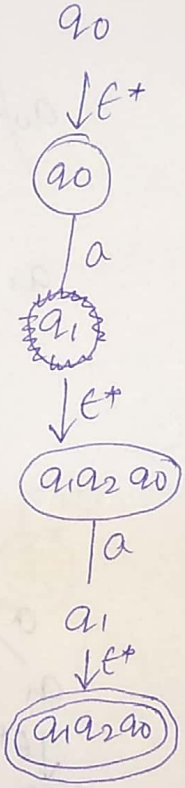
Q:



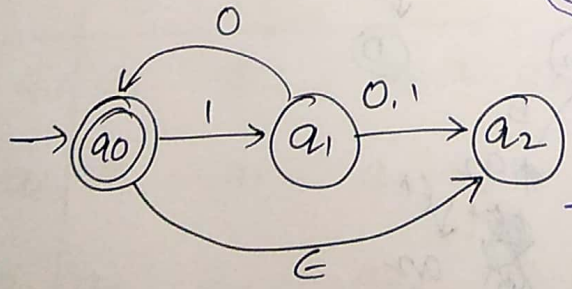
	a	ϵ^*
→ q_0	q_1	q_0
q_1	ϕ	q_1, q_2, q_0
q_2	ϕ	q_2, q_0







Q:



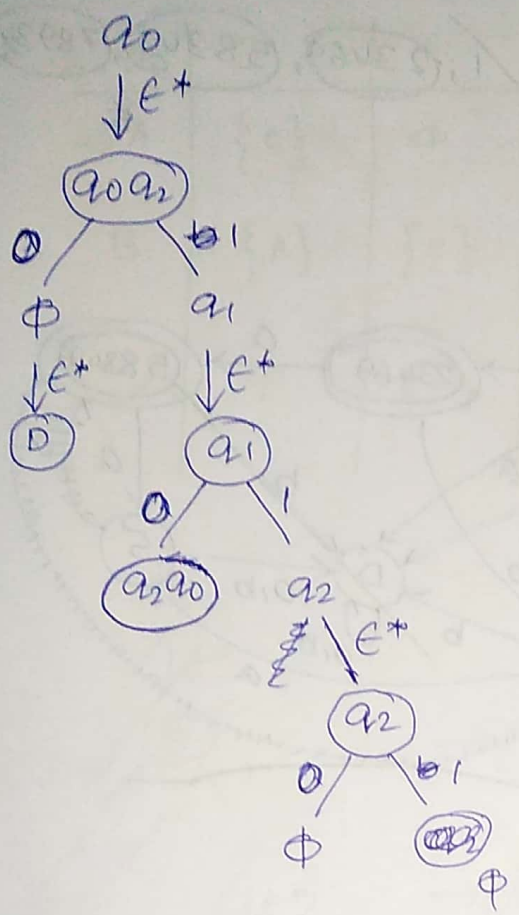
	0	1	ϵ^*
→ q_0	ϕ	q_1	q_0, q_2
q_1	q_2, q_0	q_2	q_1
q_2	ϕ	ϕ	q_2

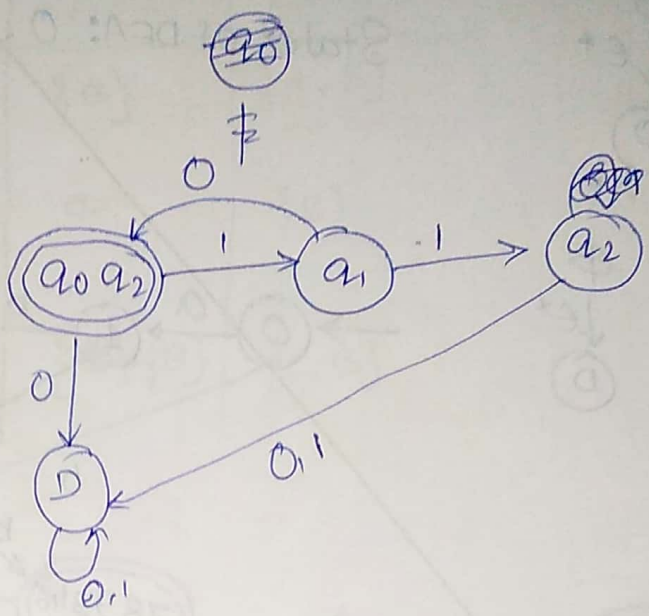
$q_0 q_2$

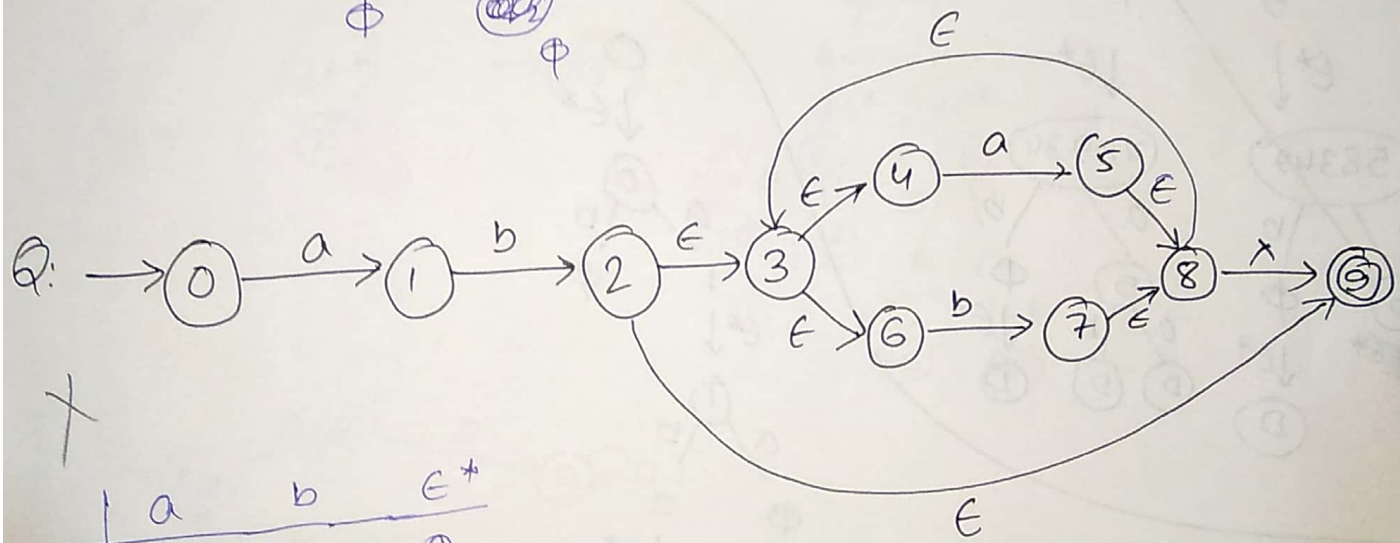
q_1

q_2

	0	1
$q_0 q_2$	ϕ	q_1
q_1	$q_0 q_2$	q_2
q_2	ϕ	ϕ





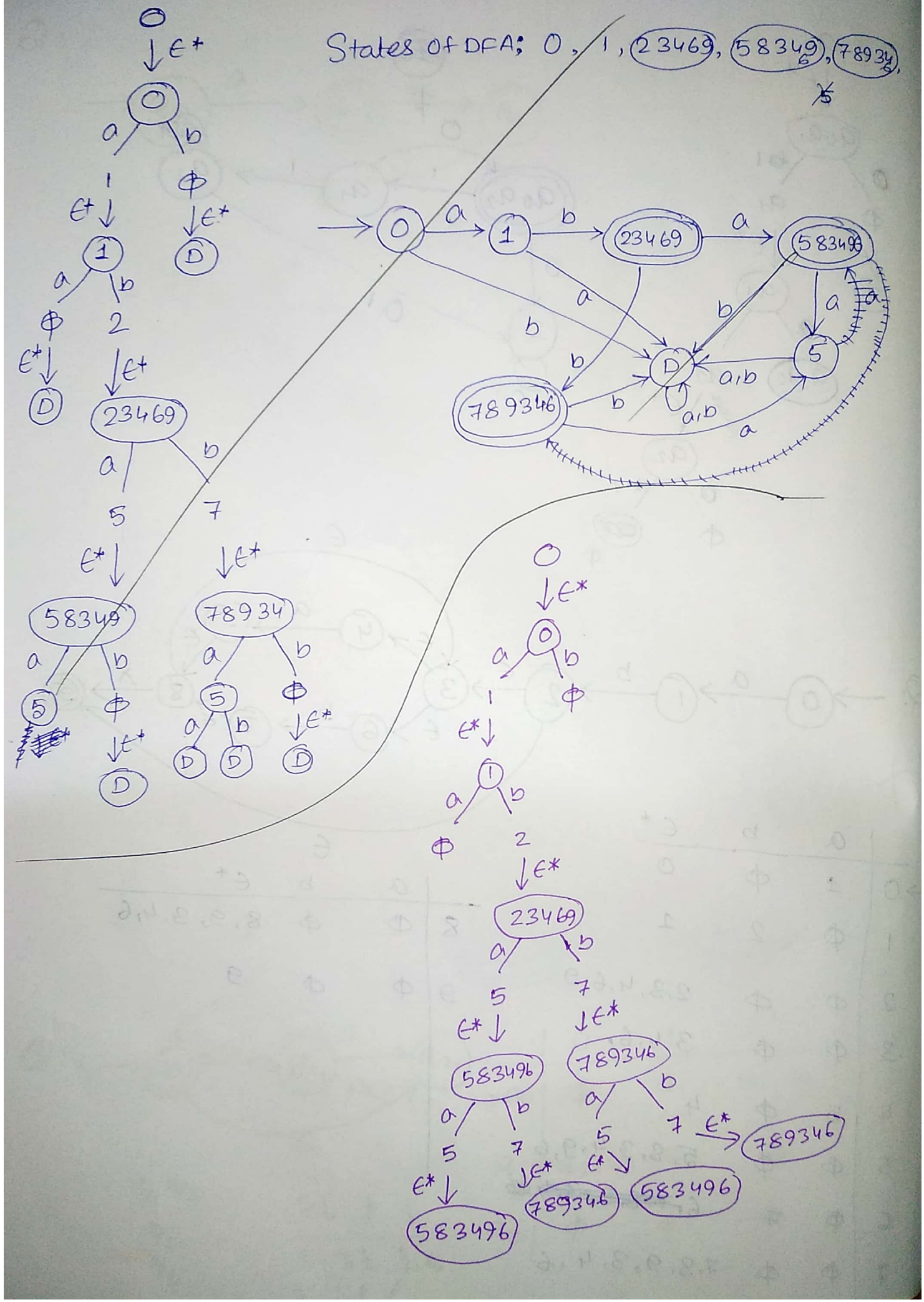


	a	b	ϵ^*
→ 0	1	ϕ	0
1	ϕ	2	1
2	ϕ	ϕ	2,3,4,6,9
3	ϕ	ϕ	3,4,6,
4	5	ϕ	4
5	ϕ	ϕ	5,8,3,4,9,6
6	ϕ	7	6, ~~7,8,9,3,4,6~~
7	ϕ	ϕ	7,8,9,3,4,6

	a	b	ϵ^*
8	ϕ	ϕ	8,9,3,4,6
9	ϕ	ϕ	9

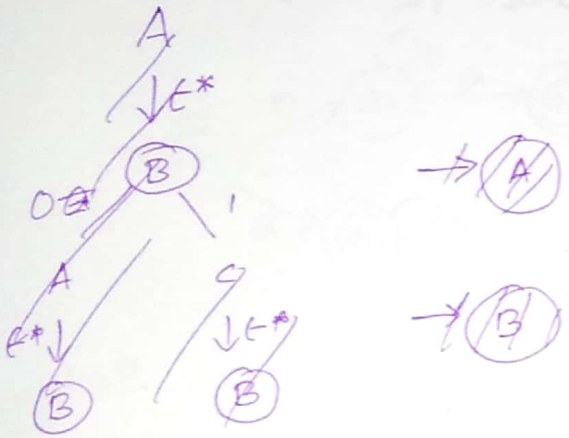
Do yourself

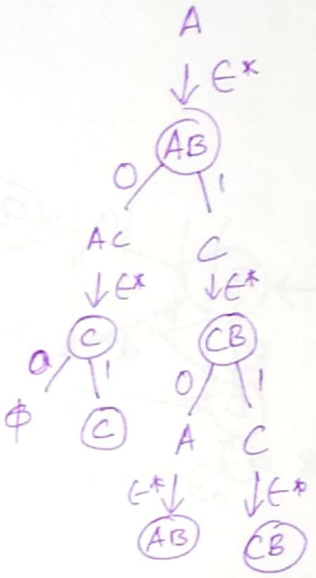
States of DFA; 0, 1, 23469, 583496, 789346,

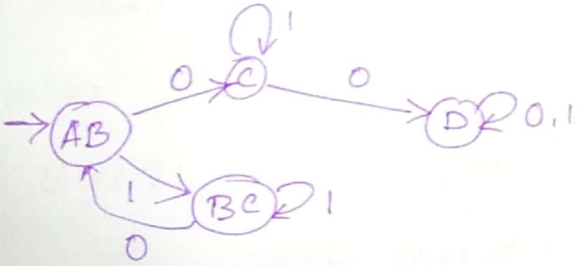


Q:

δ	0	1	ϵ	ϵ^*
$\rightarrow$A	{C}	ϕ	{B}	{A, B}
B	{A}	{C}	ϕ	{B}
(C)	ϕ	{C}	~~{ϕ}~~ {B}	{C, B}.

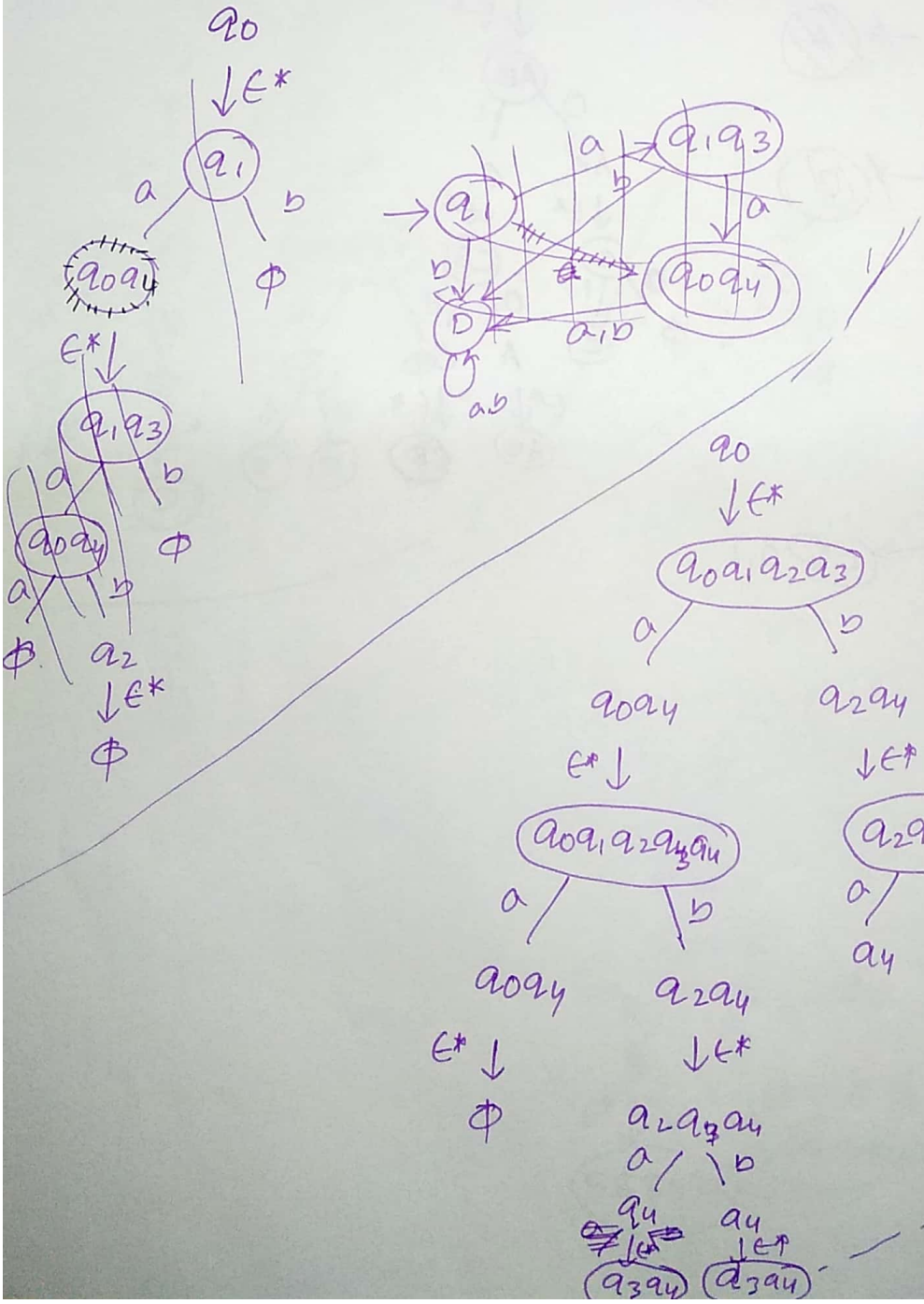






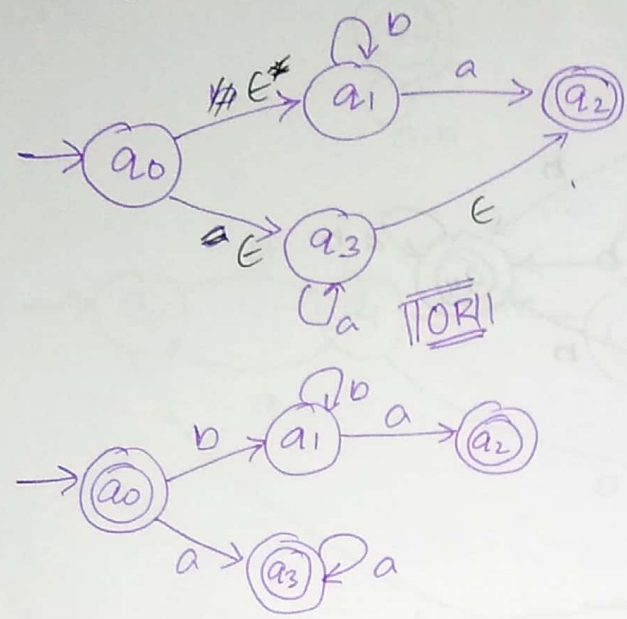
Q:

δ	a	b	ϵ	ϵ^*
$\to q_0$	ϕ	q_2	q_1	$\{q_0, q_1, q_2, q_3\}$
q_1	$\{q_0, q_4\}$	ϕ	$\{q_2, q_3\}$	$\{q_1, q_2, q_3\}$
q_2	ϕ	q_4	ϕ	$\{q_2\}$
q_3	q_4	ϕ	ϕ	$\{q_3\}$
(q_4)	ϕ	ϕ	q_3	$\{q_4, q_3\}$



Q: Design an NFA with 4 states for the language

$$L = \{b^n a \mid n \geq 0\} \cup \{a^n \mid n \geq 0\}$$



27/01/20

Minimization DFA:

Step1: Remove all the non-reachable state from starting state.

Step2: Construction of Π_{old}.

Obtain the indistinguisable states i.e. consisting of only final states & non-final states by partitioning method. Π_{old} consists of final states and other consists of non-final states.

Step 3: Construct Π_{new} from Π_{old}.

Step 4: If $\Pi_{new} != \Pi_{old}$

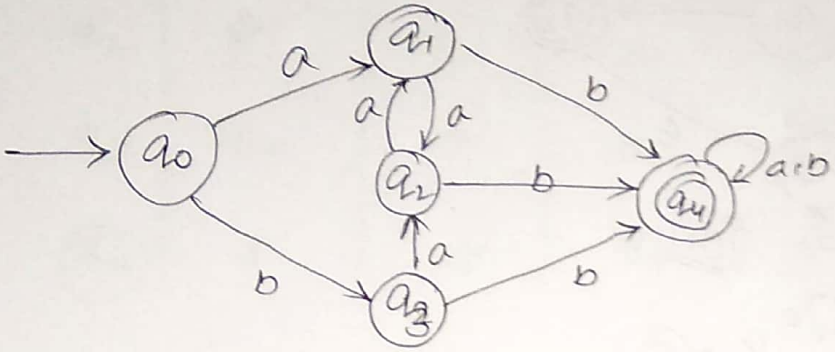
Copy Π_{new} to Π_{old}.

goto step 3

Step 5: Pick a representation from each group π.

Step 6: These representation will be the states of reduced DFA

Q:

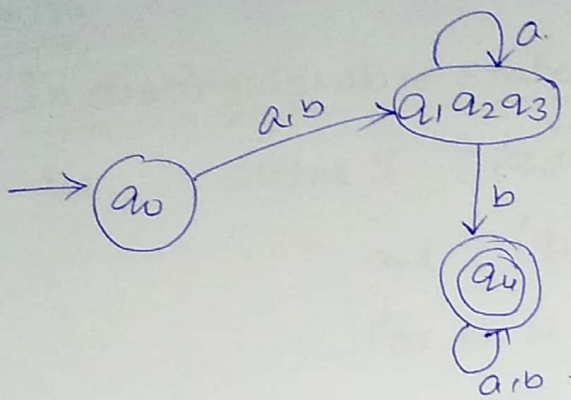


	a	b
→ q_0	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
(q_4)	q_4	q_4

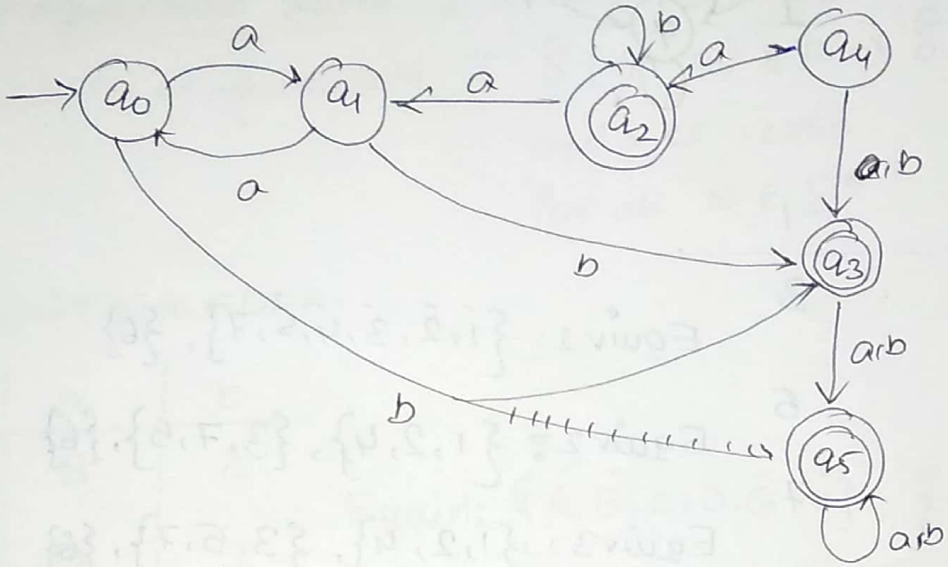
Equiv 1: $\{q_0, q_1, q_2, q_3\}, \{q_4\}$.

Equiv 2: $\{q_0\}, \{q_1, q_2, q_3\}, \{q_4\}$.

Equiv 3: $\{q_0\}, \{q_1, q_2, q_3\}, \{q_4\}$



Q:

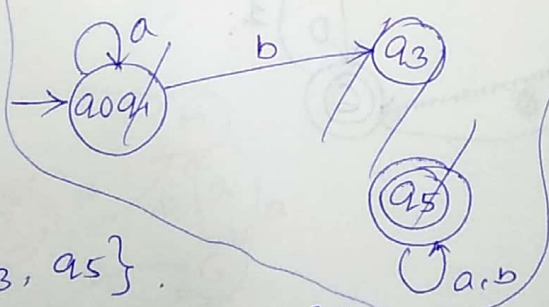


	a	b
→ q_0	q_1	q_3
q_1	q_0	q_3
~~q_2~~	q_1	q_2
q_3	q_5	q_5
~~q_4~~	q_2	q_3
q_5	q_5	q_5

Equiv 1 : $\{q_0, q_1, q_3, \cancel{q_4}\}, \{\cancel{q_2}, q_5\}$.

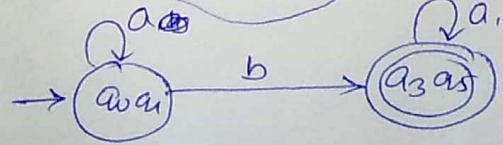
Equiv 2: $\{q_0, q_1\}$ $\{\cancel{q_3}, \cancel{q_4}, \cancel{q_2}, \cancel{q_5}\}$ $\{q_3, q_5\}$.

Equiv 3: $\{q_0, q_1\}, \{q_3\}, \{q_5\}$.

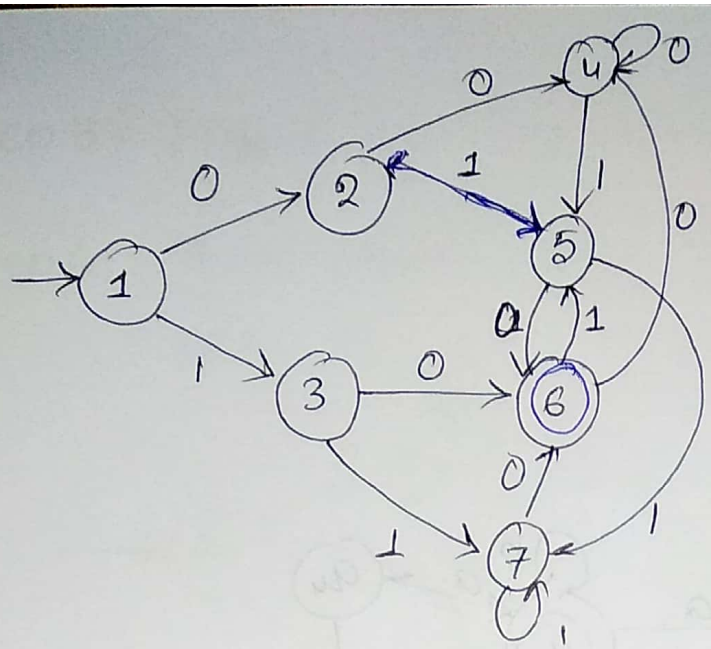


Equiv 1: $\{q_0, q_1\}, \{q_3, q_5\}$.

Equiv 2: $\{q_0, q_1\}, \{q_3, q_5\}$



Q:

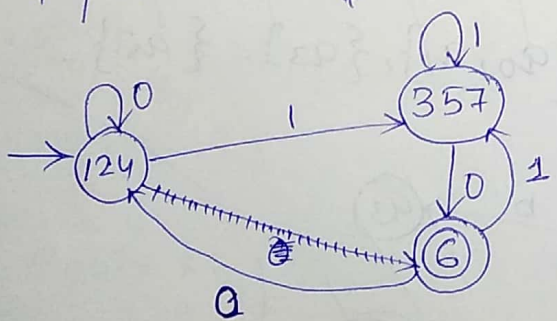


	0	1
→ 1	2	3
2	4	5
3	6	7
4	4	5
5	6	7
(6)	4	5
7	6	7

Equiv 1: {1, 2, 3, 4, 5, 7}, {6}.

Equiv 2: {1, 2, 4}, {3, 7, 5}, {6}

Equiv 3: {1, 2, 4}, {3, 5, 7}, {6}.



28/01/20

(*) Indistinguishable states. 2 states q_1 & q_2 are called indis.... states if $\hat{\delta}(q_1, w) \in F \Rightarrow \hat{\delta}(q_2, w) \in F$

and $\hat{\delta}(q_1, w) \notin F \Rightarrow \hat{\delta}(q_2, w) \notin F$

for all $w \in \Sigma^*$

(*) Distinguishable States: 2 states q_1 & q_2 of a DFA are called Distinguishable if $\hat{\delta}(q_1, w) \in F \Rightarrow \hat{\delta}(q_2, w) \notin F$

and vice-versa

for all $w \in \Sigma^*$

Minimize the DFA:

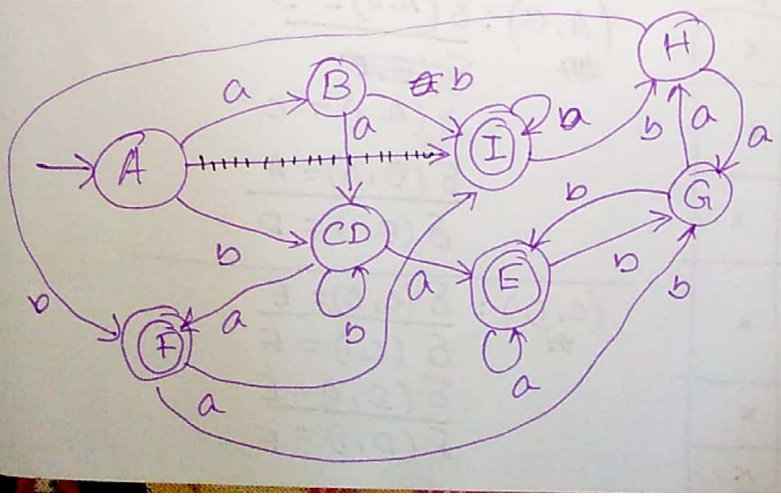
S	a	b
→A	B	C
B	D	I
C	E	D
D	F	G D
(E)	E	G
(F)	I	G
G	H	E
H	G	F
(I)	I	H

Equiv1: {A, B, C, D, G, H}, {E, F, I}.

Equiv2: {A}, {B, G, H}, {C, D}, ~~{I}~~ {E, F, I}.

Equiv3: {A}, {B} ~~{G}~~, {G, H}, {C, D}, {E, F, I}.

Equiv4: {A}, {B}, {G, H}, {C, D}, {E, F, I}.



MYHILL-NERODE THEOREM: (Table Filling Method)

To minimize DFA

Step 1: Draw a table for all pairs of states (P, Q)

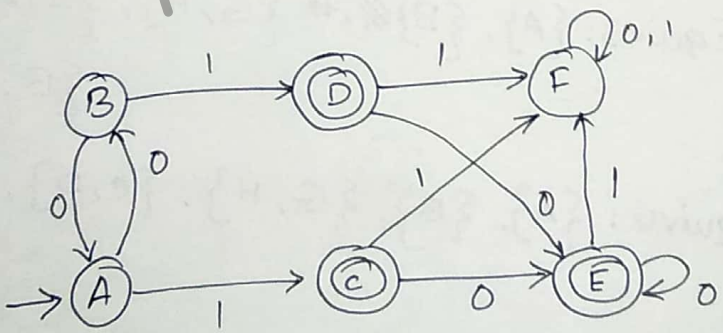
Step 2: Mark all pairs where P ∈ Final and Q ∉ Final or vice-versa.

Step 3: If any unmarked pairs are there (P, Q) such that $[\delta(P, x), \delta(Q, x)]$ is marked, then mark [P, Q].

Repeat this procedure till no more marking possible.

Step 4: Combine all unmarked pairs and make them a single state in the minimized DFA.

Q:



	A	B	C	D	E	F
A	×	×	×	×	×	×
B	✓	×	×	×	×	×
C	✓	✓	×	×	×	×
D	✓	✓	✓	×	×	×
E	✓	✓	✓	✓	×	×
F	✓	✓	✓	✓	✓	×

(A, B): $\delta(A,0) = B$

~~$\delta(B,0)$~~

$\delta(A,1) = C$

$\delta(B,0) = A$

$\delta(B,1) = D$

(C, D): $\delta(C,0) = E$

$\delta(C,1) = F$

$\delta(D,0) = E$

$\delta(D,1) = F$

(C, E): $\delta(C,0) = E$; $\delta(C,1) = F$; $\delta(E,0) = E$; $\delta(E,1) = F$

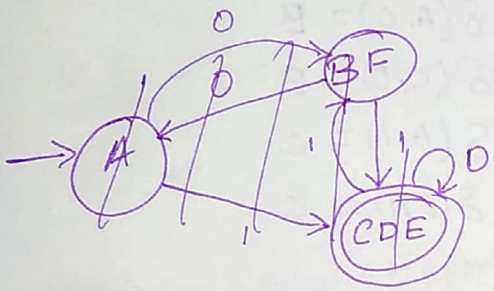
(A, F): $\delta(A,0) = B$; $\delta(A,1) = C$; $\delta(F,0) = F$; $\delta(F,1) = F$

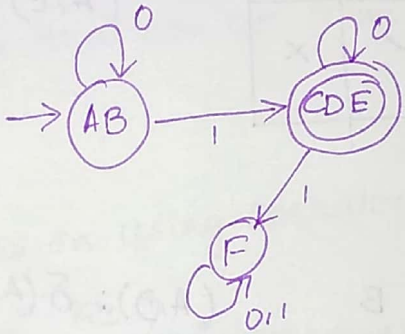
(B, F): $\delta(B,0) = A$; $\delta(B,1) = D$; $\delta(F,0) = F$; $\delta(F,1) = F$

(E, D): $\delta(E,0) = E$; $\delta(E,1) = F$; $\delta(D,0) = E$; $\delta(D,1) = F$

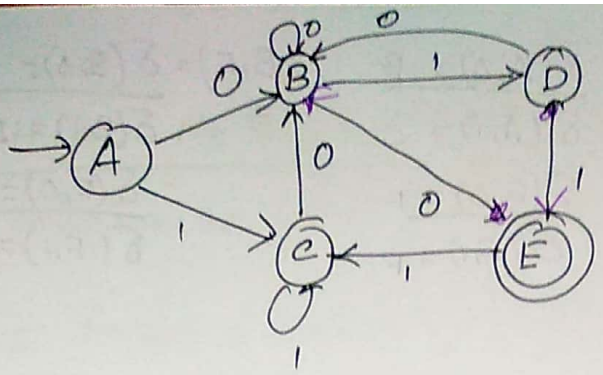
States: ~~(C,E), (B,F), (CDE),~~ A.

(A, B), (CDE), F





Q:



	A	B	C	D	E
A	X	x	x	x	x
B	~~x~~	x	x	x	x
C	·	~~x~~	x	x	x
D	~~x~~	~~x~~	~~x~~	x	x
E	✓	✓	✓	✓	x

(A, B): $\delta(A,0) = B$
$\delta(B,0) = B$
$\delta(A,1) = C$
$\delta(B,1) = D$

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

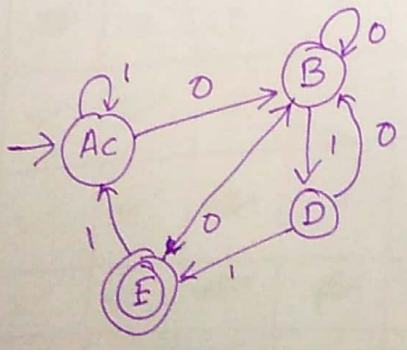
(B, C): $\delta(B,0) = B$
$\delta(C,0) = B$
$\delta(B,1) = D$
$\delta(C,1) = C$

(A, D): $\delta(A,0) = B$
$\delta(D,0) = B$
$\delta(A,1) = C$
$\delta(D,1) = E$

(B, D): $\delta(B,0) = B$
$\delta(D,0) = B$
$\delta(B,1) = D$
$\delta(D,1) = E$

(C, D): $\delta(C,0) = B$
$\delta(D,0) = B$
$\delta(C,1) = C$
$\delta(D,1) = E$

States: (A, C), B, D, E.



03/02/20

Regular Expression:

Pattern to recognise a string.

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

P_1 : starts with 0.

P_2 : ends with 001.

This pattern is implemented by Regular Expressions.

R.E : The expression that can be formed with operators '+', '*', '.'

Mathematical Expression using operators is called Regular Expression.

RL
FA
RE
} inter-convertable.

Regular Operators : * → Kleen's Closure
. → Concatenation
+ → Union

Operator Precendence : * > . > +

N.B: Every regular expression generates one regular language. The lang. generated by r.e. should be accepted by FA.

Construction of R.E from R.L:

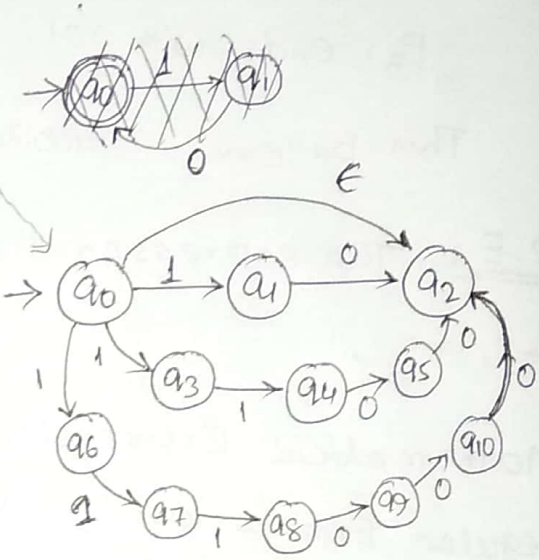
Q:1 $L = \{\lambda, 0, 11\}$.

R.E = $(\emptyset + 1 + 11)^*$ $\lambda + 0 + 11$

Q:2 $L = \{1^n 0^n \mid 0 \le n \le 3\}$

~~R.E = $\lambda + 10 + (10)^2 + (10)^3$~~

R.E = $\lambda + 10 + 1100 + 111000$.



Q:3 $L = \{1^n 0^m / m+n = 2\}$

R.E = $11 + 10 + 00$

Q:4 $L = \{1^n \mid n \ge 0\}$.

R.E = 1^*

Q:5 $L = \{1^{2n} \mid n \ge 0\}$

R.E = $(11)^*$

Q:6 $L = \{1^{2n} 0^{3m+1} / m, n \ge 0\}$

R.E = ~~$(11(000)^* 0)^*$~~ $(11)^* (000)^* 0$

Q:7 $L = \{01^{2n} 0^{m+1}\}$, $m, n \ge 0$

R.E = $0(11)^* 0^* 0 = 0(11)^* 0^+$

Q:8 $L = \{1^{2n+1} / n \ge 0\}$

R.E = $(11)^* 1$ = ~~$(11)^+$~~

Q:9 $L = \{1^m 0^n , m,n \geq 0\}$

R.E = $1^* 0^*$

Q:10 $L = \{1^m 0^n / m \geq 0, n \geq 1\}$

R.E = $1^* 0^+$

Q:11 $L = \{0^n 1^n / m \equiv 2 \pmod 3 , n \equiv 1 \pmod 4\}$

R.E = ~~$000^* 0$~~ $00(000)^* 1(1111)^*$

~~$0(000)^* 1(1111)^* + (000)^* 1(1111)^*$~~

Q:12 Construct a R.E over $\{0,1\}$.

(i) starts with 01

(ii) ends with 10

(iii) containing ~~00~~ 101.

$(\lambda + 0 + 1)^* = (0+1)^*$

(i) RE = ~~0X X X 01~~ $01(0+1)^*$

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

(iii) RE = $(1+0)^* 101 (1+0)^*$

04102120

Q:1 Construct R.E over $\{a,b\}$ where length of string —

(i) at least 3 (ii) at most 3, (iii) exactly 3.

(i) RE = $(a+b)^3 (a+b)^*$

(ii) RE = $\epsilon + (a+b) + (a+b)^2 + (a+b)^3 = (\epsilon + a + b)^3$

(iii) RE = $(a+b)^3$ ~~$+ (a+b)$~~

Q: length at most: 2 : $RE = (a+b+\lambda)^2$

$= \lambda + (a+b) + (a+b)^2$

Q:2 (i) Starts and ends with same symbols.

(ii) Starts and ends with diff. symbols.

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

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

Q:3 (i) starts with '00' or '11'

(ii) ends with '00' or '11'

(iii) Containing '00' or '11'

(i) $RE = 00(0+1)^* + 11(0+1)^* = (00+11)(0+1)^*$

(ii) $RE = (0+1)^* 00 + (0+1)^* 11 = (0+1)^* (00+11)$

(iii) $RE = (0+1)^* 00 (0+1)^* + (0+1)^* 11 (0+1)^*$

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

Q:4 (i) 5th bit from left end is 0.

(i) $RE = (0+1)^4 0 (0+1)^*$

Q: 5 4th bit from right end is 1.

$RE = (0+1)^* 1 (0+1)^3$

Q: 6 ~~All strin~~ 2nd & 4th bit from left end is 0 or 1st & 5th bit from right end is 1

$RE = (0+1)\,0\,(0+1)\,0\,(0+1)^* + (0+1)^* 1 (0+1)^3 1$ ~~(0+1)~~.

Q: 7 length of string divisible by 3 / $|w| \bmod 3 = 0$

$RE = ((a+b)^3)^*$ ~~(0+1)~~

Q: 8 No. of 1's is even.

RE : ~~0*(11)*0*~~ $0^* 1^* (0^* 1 0^* 1 0^*)^*$

Q: 9 length at least 2, begins and ends with 'a' & in between any words using 'b'.

$RE = a b^* a$.

Q: 10 $L = \{a^n b^m / m \le 2, n \ge 3\}$

$RE = aaaa^* + aaaa^* b + aaaa^* b^2$

$= a^3 a^* (b + b^1 + \epsilon)$

Q: 11 $L = \{a^n b^m, (n+m) \text{ is even}\}$

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

m	n	m+n
o	o	e
e	o	o
o	e	o
e	e	e.

06/02/20

Q:1 $L = \{a^m b^n : m+n \text{ is even and } m*n \text{ is odd}\}$

$R.E = a(aa)^* b(bb)^*$

m	n	m+n	m*n
0	0	e	o
o	e	o	e
e	o	o	e
e	e	e	e

Q:2 Every string starts with '101' & length should be even

$R.E = 101((0+1)^2)^*(0+1)$

Q:3 Every string doesn't contain consecutive '0' ~~or~~ and consecutive '1'.

ϵ ✓ 0 ✓ 1 ✓ 00 × 01 ✓ 10 ✓ 11 ×

~~$R.E = 0(1)^* + 1(0)^* + \epsilon$~~

~~$\epsilon + 0 + 1 + (10)^* + (01)^*$~~

$RE = (\lambda+1)(01)^*(0+\lambda)$ or $(\lambda+0)(10)^*(\lambda+1)$.

Q:4 All strings of 0's & 1's where each string doesn't contain '10'. $L = \{\epsilon, 0, 1, 00, 01, 11, 0011, 000111, \ldots\}$

~~$RE = (\lambda+1)(01)^* + \epsilon/1*(0^*+1^*+0^*0^*1^*)$~~

~~$(\epsilon + 0/+1)^* + (01)1^* +$~~

$RE = 0^*1^*$

Q:5 Every string starts with 0 & doesn't contain consecutive 1's. $L = \{0, 00, 000, \ldots 01, 001, 011, \ldots\}$

$R.E = 0^*/101$ ~~$0^* + i(01)^* + 0(10)^*$~~ $(0+01)^+$

~~$= 0^+(01+10)^*$~~

Q:6 At least one a & one b. over {a,b,c}

~~RE = (b+c)* c* a+ c* b+ c* + c* b+ c+ a+ c*~~

RE = (a+b+c)* a (a+b+c)* b (a+b+c)* +

(a+b+c)* b (a+b+c)* a (a+b+c)*

Q:7 Ending with b & no substring 'aa'.

R.E = ~~ε~~ b* ~~(ab+bb+ba)*~~ ~~b~~ a b aa ab ba bb

✗ ✓ ✗ ✓ ✗ ✓

= (ab+b)+

(b+ab+bb)+

Q:8 2 consecutive symbols contains at least 2 a's.

R.E = (b* a+ b* a+ b*)* +

b* a+ b* a+ b* b* a+ b* a+ b*

abaab aabba

RE = [aa(a+b)* + a(a+b)a(a+b)* + a(a+b)*a + (a+b)a*(a+b)

+ (a+b)a(a+b)a + (a+b)*a*]+

~~RE = (a+b)+ a (a+b)+ a (a+b)*~~

~~aabb will not be accepted~~

abbaaa

aabb

~~RE = (a+b)+ ((a+b)* a (a+b)* a (a+b)*)* (a+b)*~~

RE = [aa(a+b)* + a(a+b)a(a+b) + a(a+b)*a + (a+b)a*(a+b)

+ (a+b)a(a+b)a + (a+b)*a*]+

~~(a+b)* a (a+b)* + (a+b)~~

07/02/2020

Q: Strings not ending with '01'.

$L = \{\epsilon, 0, 1, 00, 10, 11, \ldots\}$

$RE = \epsilon + 0 + 1 + 00 + 10 + 11$

$= (0+1)^* (00 + 10 + 11) + (\epsilon + 0 + 1)$

Q: Strings not ending with 'bba'

$RE = (a+b)^* (aaa + aab + aba + abb + bab + bbb + baa) + (\epsilon + a + b)^2$

($(\epsilon + a + b)^2$: at most 2 a or b)

$= (\epsilon + a + b) + (a+b)^2 + (a+b)^* \{aaa + aab + abb + aba + baa + bab + bbb\}$.

Q: $L = \{vwv \mid w, v \in (a+b)^+ \text{ and } |v| = 2 \text{ and } |w| \geq 2\}$

$w, v \in a, b$.

$RE = \underbrace{(a+b)^2}_{v} \underbrace{(a+b)^* (a+b)^*}_{w} \underbrace{(a+b)^2}_{v}$

Q: $L = \{a^n b^m : n \geq 4, m \leq 3\}$

$RE = a^4 a^* (b + b^2 + b^3 + \epsilon)$.

$= a^4 a^* \underbrace{(\epsilon + b)^3}_{\text{at most 3 b's}}$

Q: Construct a RE over $\{a, b\}$ length of string is at most 10.

$RE = \underbrace{(a + b + \epsilon)^{10}}_{\text{at most 10 a or b.}}$

Q: $L = \{a^n b^m : n \geq 40, m \leq 300\}$

$RE = a^{40} a^* (\lambda + b)^{300}$

(*) Standard Regular Expressions:

MSIES Σ is an alphabet set. Standard Regular Exp. are.

(i) λ or ϵ

(ii) For each $a \in \Sigma$ is the symbol is a regular expression.

(iii) If R is a regular exp. then R^* is also be RE.

(iv) If R_1 & R_2 are 2 two REs, then $R_1 + R_2$ & $R_1 . R_2$ will be RE.

Primitive Regular Expression:

(i) $\phi, \lambda, a \in \Sigma$ are primitive RE.

(ii) If R_1 & R_2 are 2 RE, $R_1 + R_2$, $R_1^* R_2$, R_1^*, R_1 all are RE.

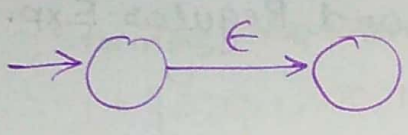
(iii) A string is a RE iff it is derived from primitive RE.

$(a+bc)(c+\phi)$ ✓

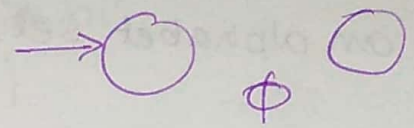
$(a+b+)$ ✗

Identities of Regular Expressions:

1. $\phi + R = R$
2. $\phi R = R \phi = \phi$
3. $\lambda R = R \lambda = R$
4. $\lambda^* = \lambda$, $\phi^* = \lambda$
5. $R + R = R$
6. $RR^* = R^* R$
7. $R^* R^* = R^*$
8. $(R^*)^* = R^*$
9. $\lambda + RR^* = \lambda + R^* R = R^*$
10. $(PQ)^* P = P(QP)^*$
11. $(P+Q)^* = (P^* . Q^*)^* = (P^* + Q^*)^*$
12. $(P+Q)R = PR + QR$
13. $R(P+Q) = RP + RQ$



ϵ-transition



ϕ-transition

10/02/20

Equality of Regular Expressions:

Two REs are said to be equal/equivalent iff

$L(r_1) = L(r_2)$.

e.g: $0(10)^*$ & $(10)^*0$ are equivalent

e.g. $r_1 = (0+10+1+11)^*$ $L_1 = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001 \ldots\}$

$r_2 = (0+1)^*$ $L_2 = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001 \ldots\}$

Both are equivalent.

Q: $R = \underset{\lambda +}{\lambda} + a^* \underset{R}{(abb)^*} \underset{R^*}{(a^*(abb)^*)^*}$

$= \lambda + (a^* (abb)^*)^*$

$= [(a+abb)^*]^*$ $[\because P^*Q^* \neq (P^*Q^*)^*]$

$= (a+abb)^*$

~~$(P^*Q^*)^*$~~

$= (P^* + Q^*)^*$

$= ((P+Q)^*)^*$

$= (P+Q)^*$

Q: $(1+00^*1) + (1+00^*1)(0+10^*1)(0+10^*1) = 0^*1(0+10^*1)^*$

LHS: $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$

$= (\lambda+00^*1)+$

$= (1+00^*1)\left[\underset{\epsilon+}{\epsilon} + \underset{R}{(0+10^*1)^*}\cdot\underset{R^*}{(0+10^*1)}\right]$

$= (1+00^*1).(0+10^*1)^*$ $\quad [\epsilon + RR^* = \epsilon + R^*R = R^*]$

$= \epsilon(\lambda+00^*)1\,(0+10^*1)^*$

$= 0^*1\,(0+10^*1)^*$ [Proved]

Construction of RE from FA:

State Elimination Method: This method is for DFA, NFA, ϵ-NFA. It reduces the FA of n states into transition graph of 2 states.

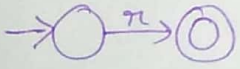
~~Steps~~:

FA

↓

SAM

↓



(Transition Graph)

Step:

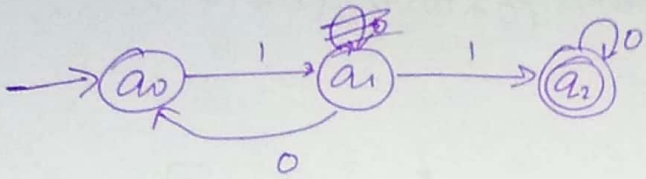
~~Step~~ Step 1: Simplify the FA to have only one initial & one final state.

Step 2: Simplify FA to have distinct initial & final state.

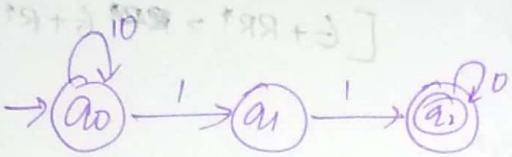
Step 3: Simplify the parallel edges.

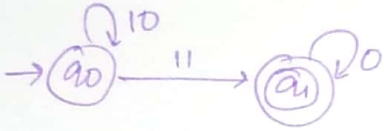
4. Eliminate cycle/loop/ states with concatenation.

Q:



$1(01)^* 1 0^*$

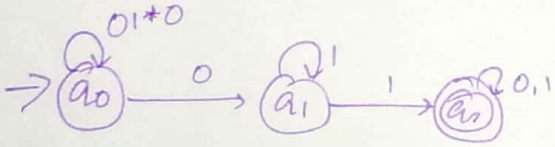


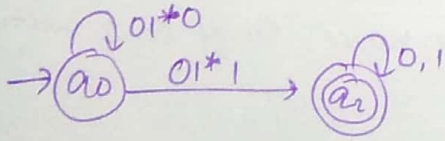


$(10)^* 11 0^*$

Q:

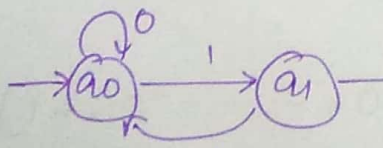
~~$0(00)^* 1^* 1 (0+1)^*$~~



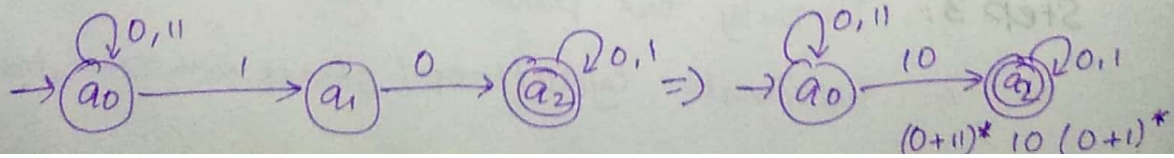


$(01^*0)^* 01^* 1 (0+1)^*$

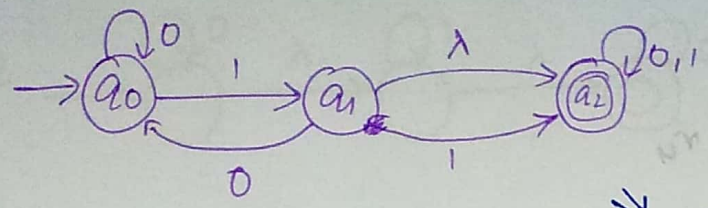
Q:



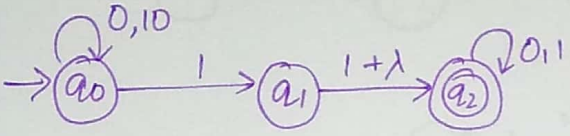
$0^* 1 (1 0^* 1)^* 0 (0+1)^*$

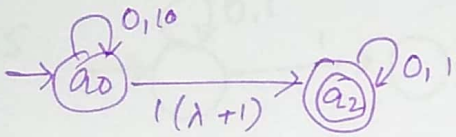


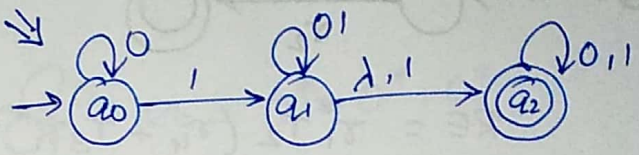
Q:

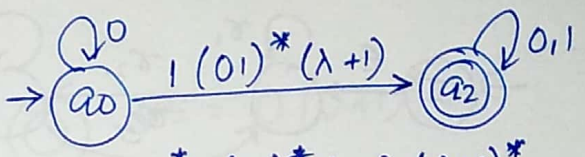


~~$0^*1(00^*1)^*1(0+1)^*$~~







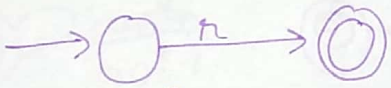


$0^*1(01)^*(\lambda+1)(0+1)^*$

$(0+10)^*1(1+\lambda)(0+1)^*$

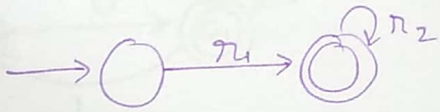
Step 4: Obtain the RE from Trans. Graph if it's any of the following forms:

(1)



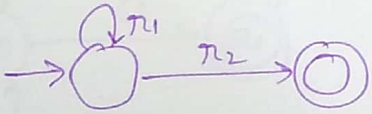
$RE = r$

(2)



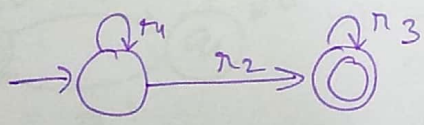
$RE = r_1 . r_2^*$

(3)



$RE = r_1^* . r_2$

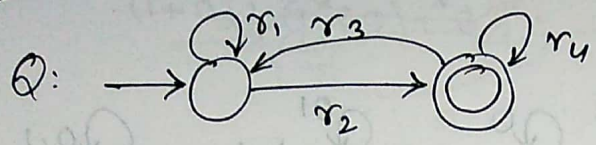
(4)



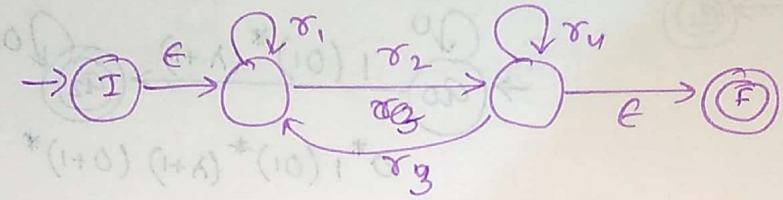
$RE = r_1^* r_2 r_3^*$

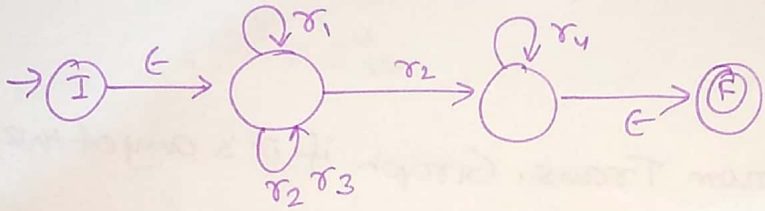
11/02/20

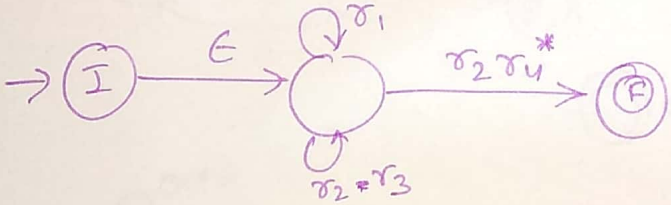
Q:

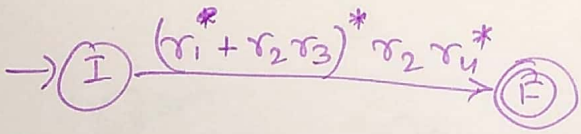


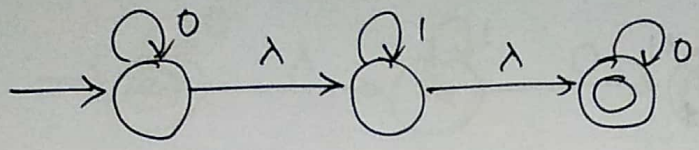
$RE = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$



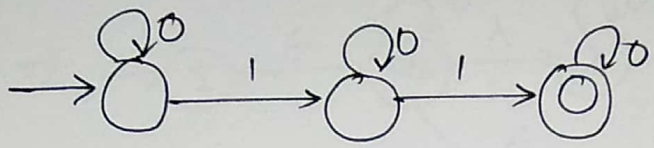




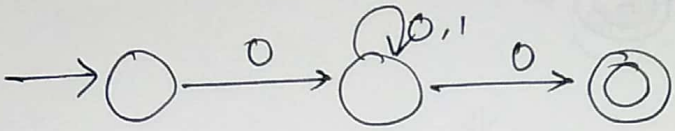


Q:2 

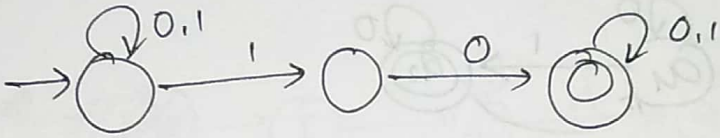
RE = $0^*1^*0^*$

Q:3 

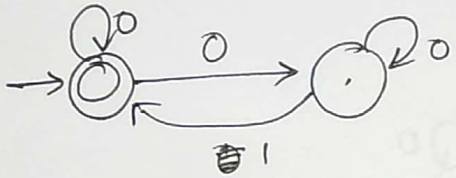
RE = $0^*10^*10^*$

Q:4 

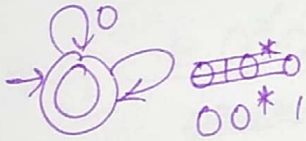
RE = $0(0+1)^*0$

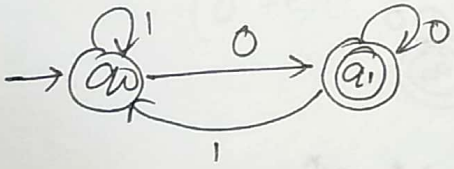
Q:5 

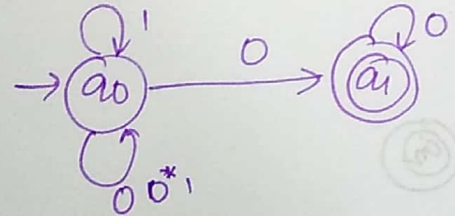
RE = $(0+1)^*10(0+1)^*$

Q:6 

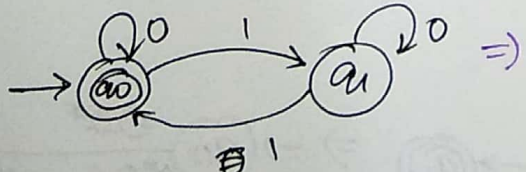
RE = ~~0^*0~~ $(0+00^*1)^*$

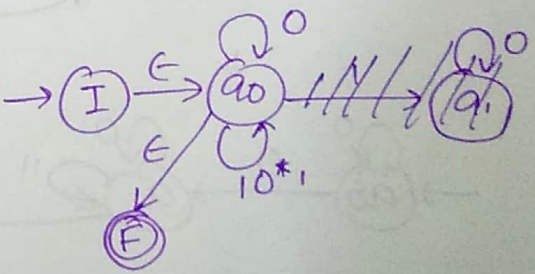


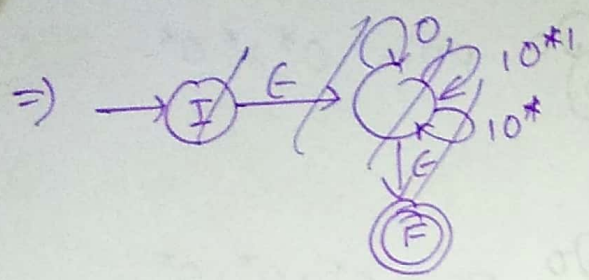
Q:7 

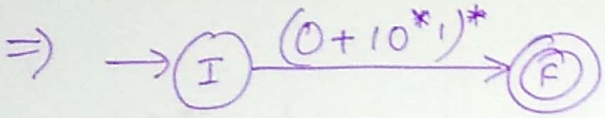


RE = $(1+00^*1)^*00^*$

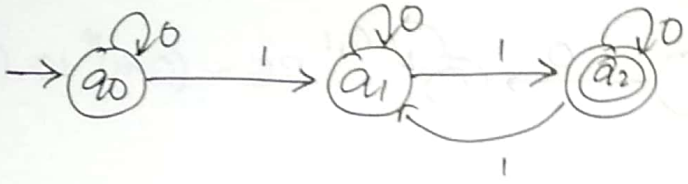
Q:8 

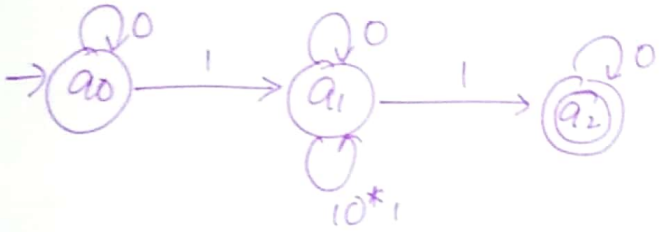
⇒ 

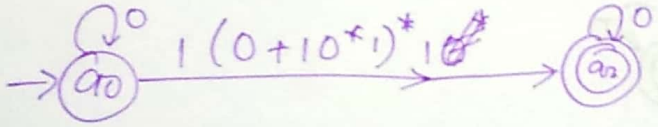




Q: 9

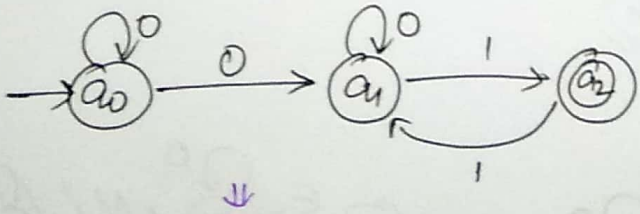


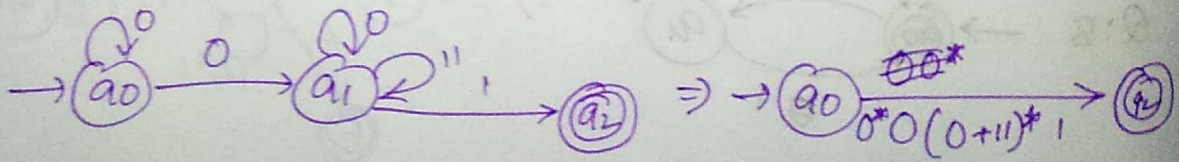




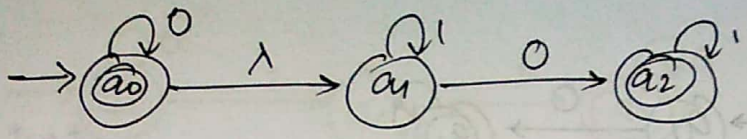
RE = $0^*1(0+10^*1)^*10^*$

Q: 10

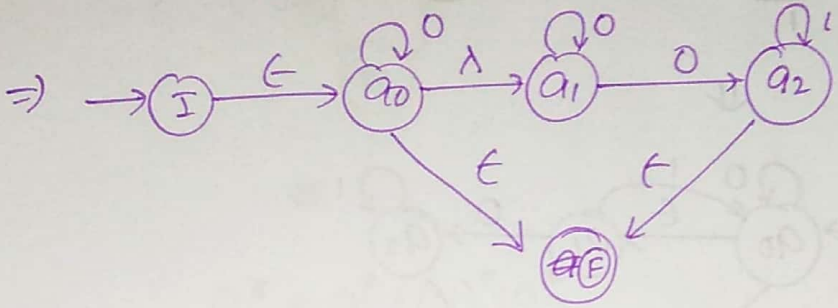


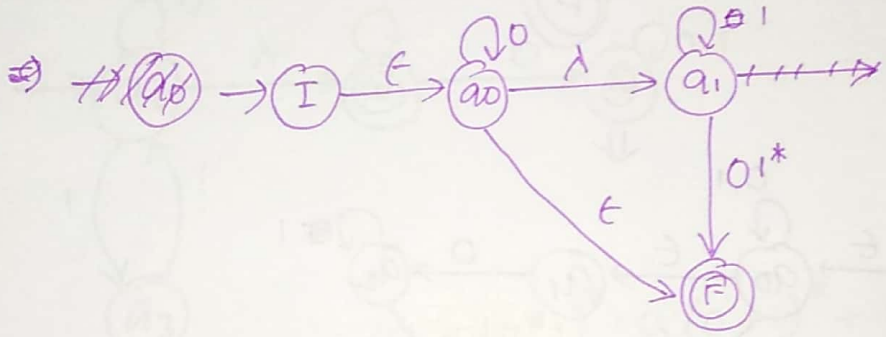


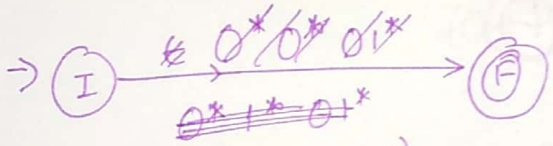
Q: 11



$0^*+0^*1^*+01^*$





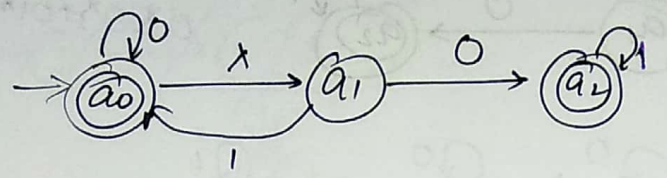


$0^*+0^*0^*01^*$

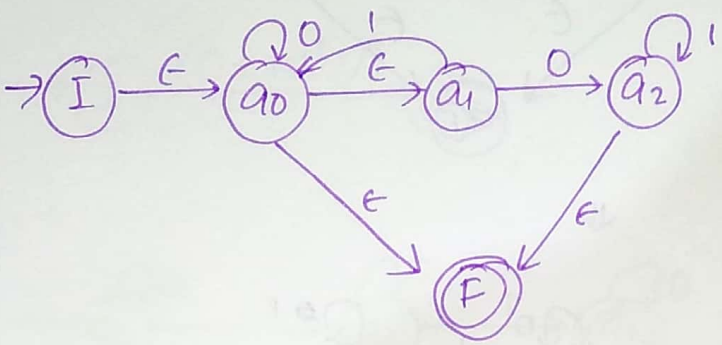
$0^*+1^*01^*$

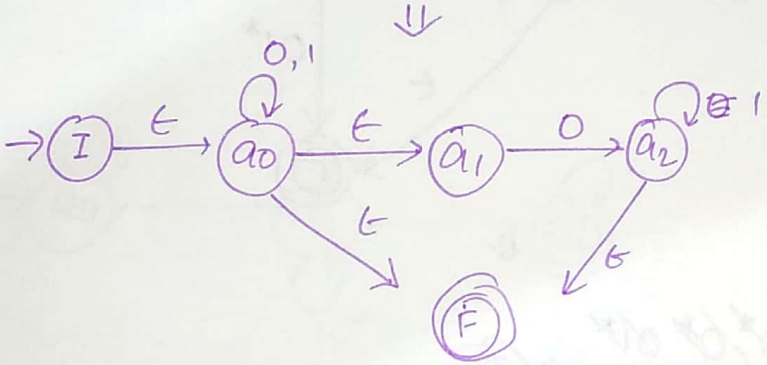
$(0^*+\epsilon)(1^*01^*)$

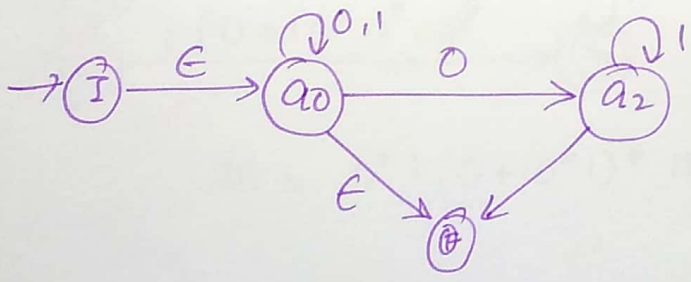
Q: 12



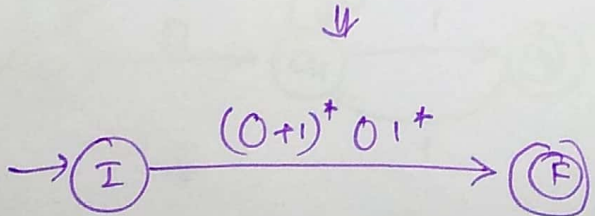
$(0+1)^* + (0+1)^* 01^*$



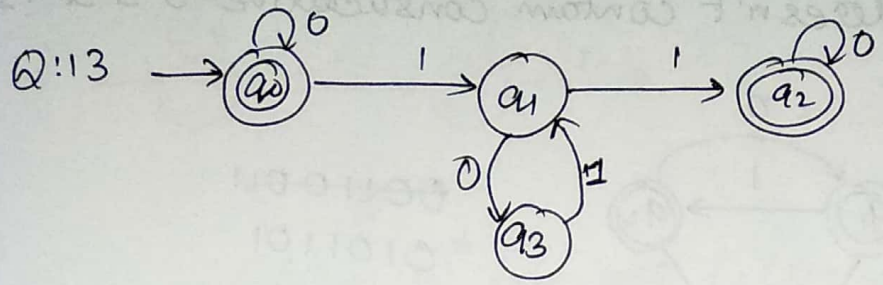




$(0+1)^* 01^*$

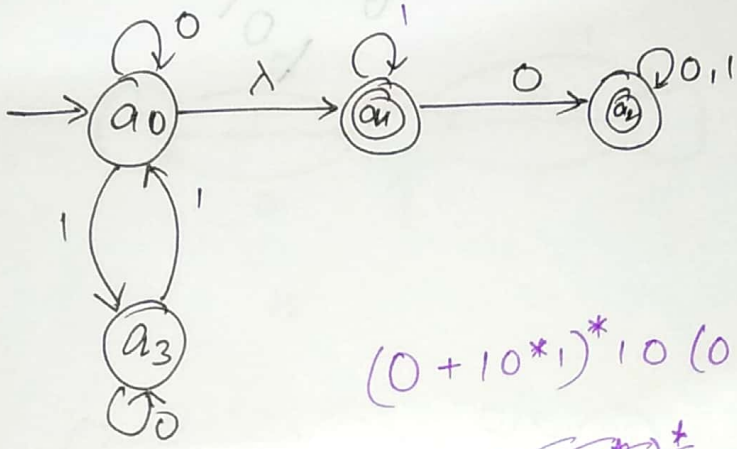


Q:13



$0^* + 0^*1(01)^*10^*$

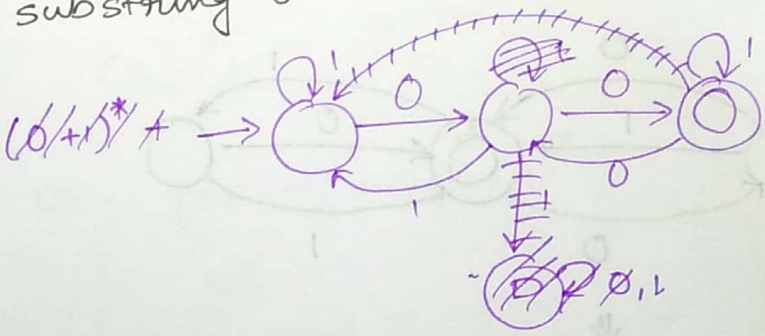
Q:14

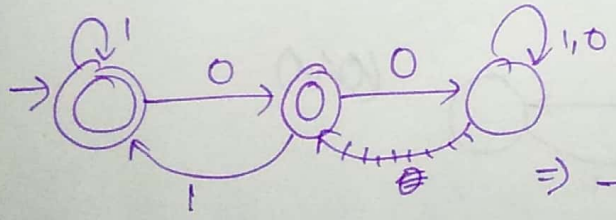


$(0+10^*1)^*10(0+1)^* + (0+10^*1)^*1^*$

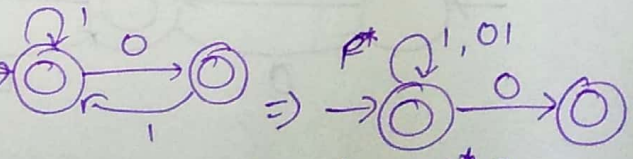
~~$(0+10^*1)^*$~~

Q:15 Construct RE that accepts every string doesn't contain substring '00'.



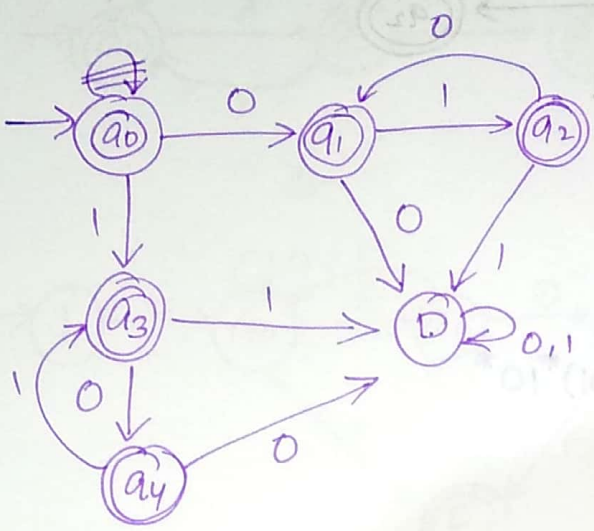


$1^*(01)^* + 1^*00(1+0)^*$



$\therefore RE = (1+01)^*0$

Q:16 Every String doesn't contain consecutive 0's & 1's



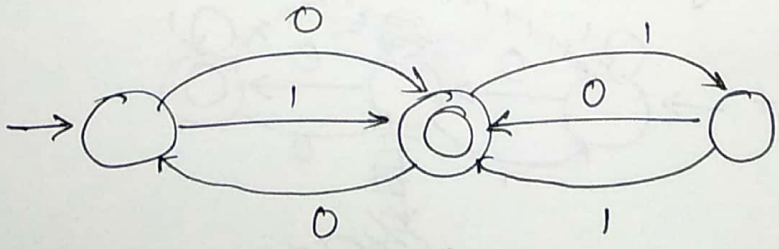
~~00~~1~~1~~0 ~~01~~1

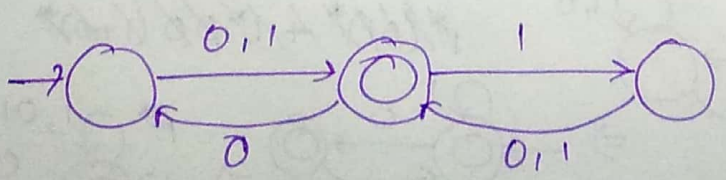
0101101

~~0~~1~~0~~

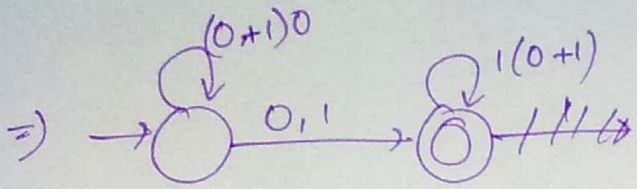
101

Q:17



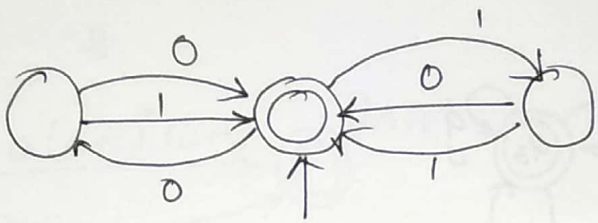


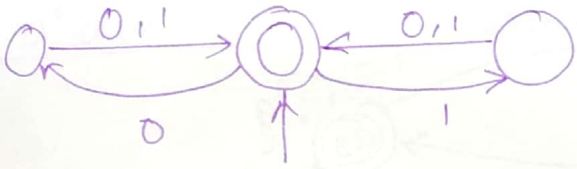
(0/+1)

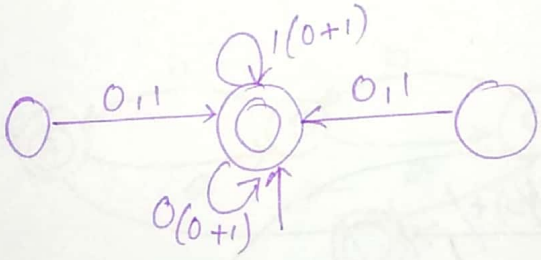


$$((0+1)0)^*(0+1)(1(0+1))^*$$

Q:18



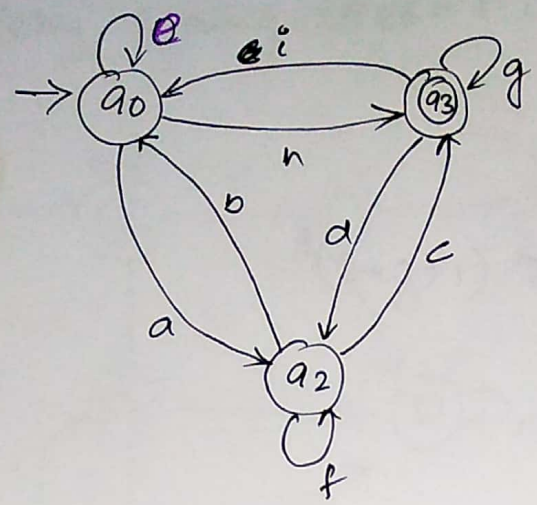


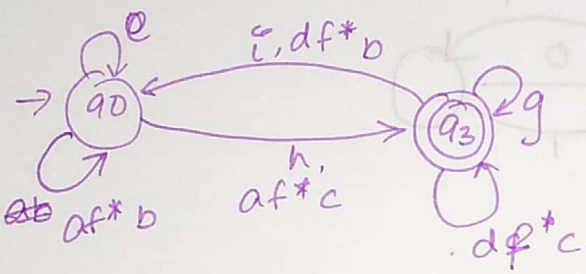


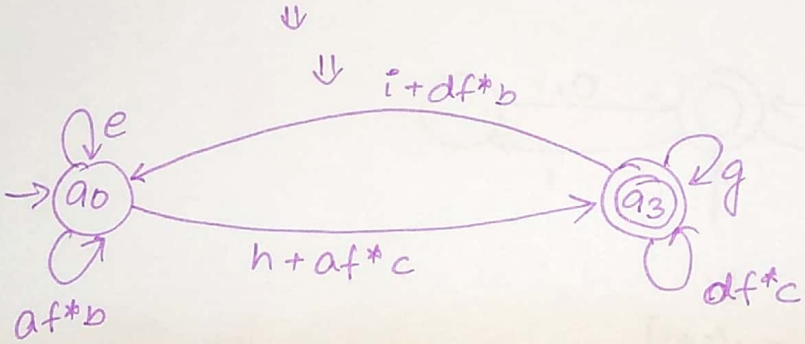
$$RE = (1(0+1) + 0(0+1))^*$$

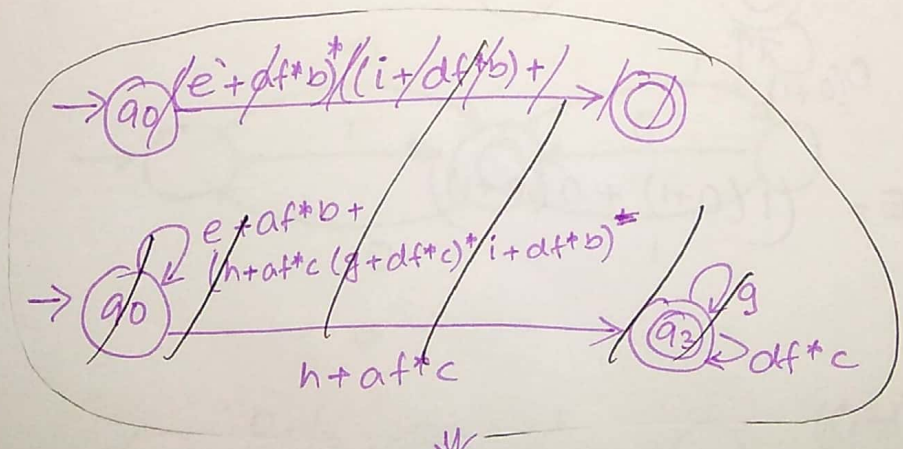
13/02/20

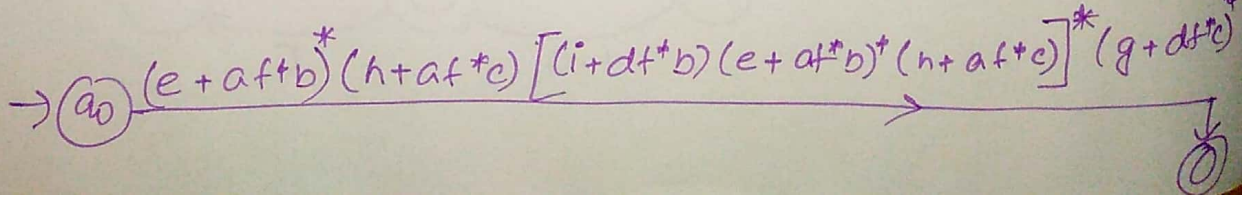
Q:19



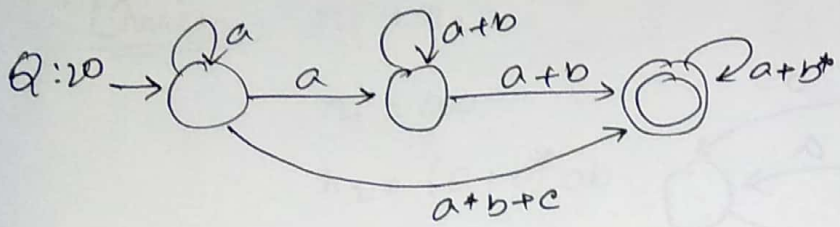


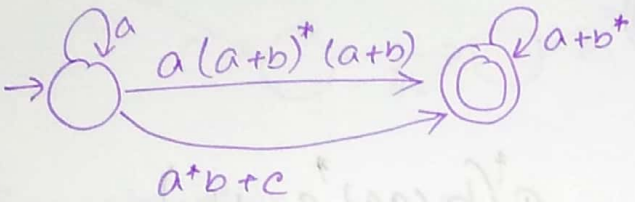


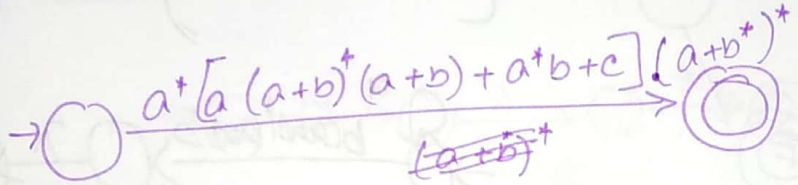




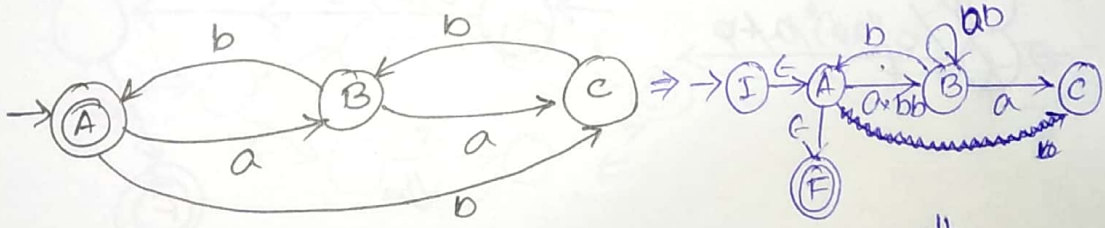
Q:20

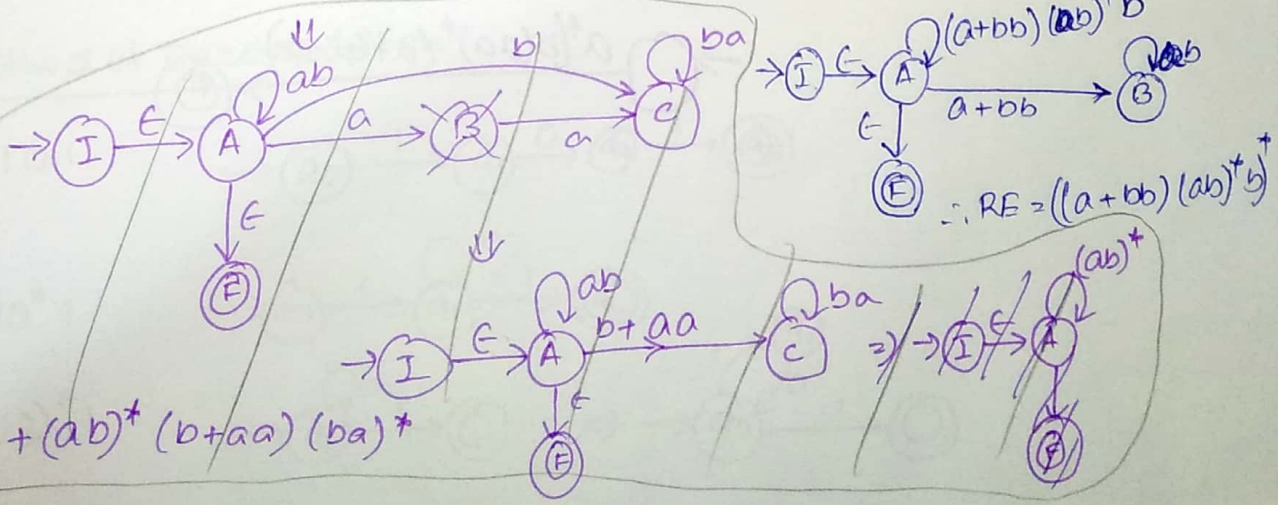




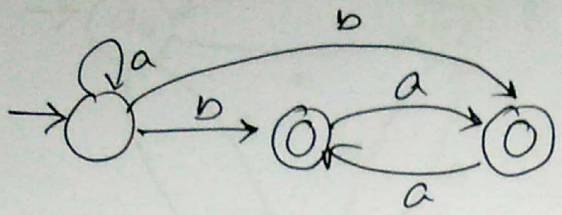


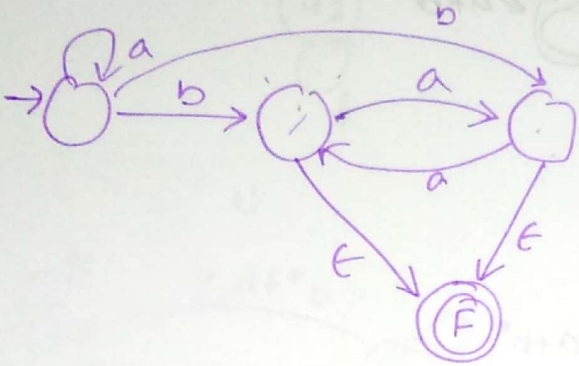
Q: 21

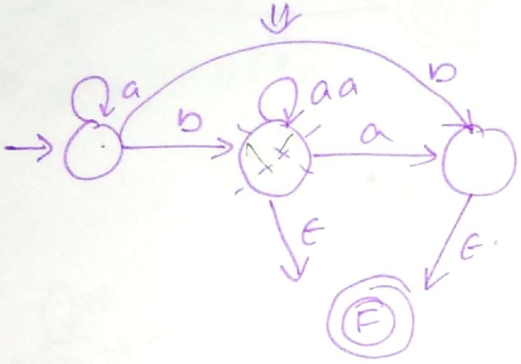




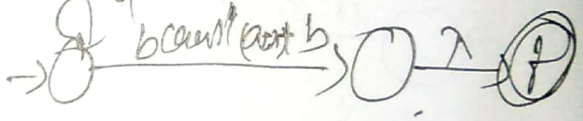
Q:22

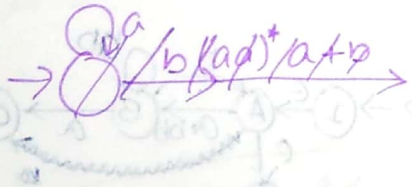


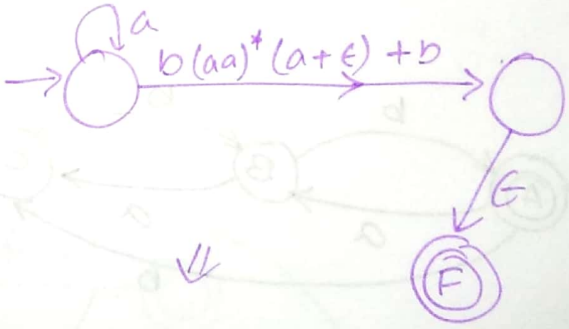


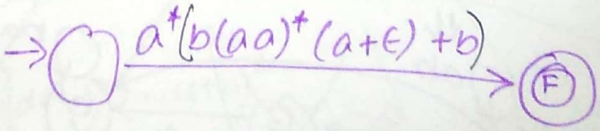


$a^*(b(aa)^*a + b)$









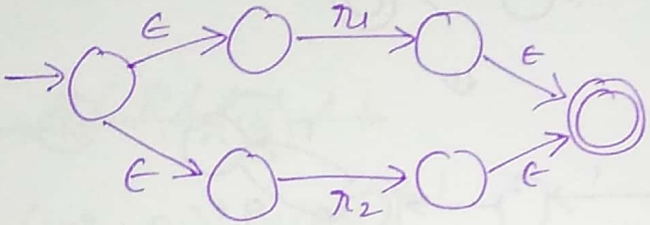
Construction of FA from RE :

Let, R is any RE.

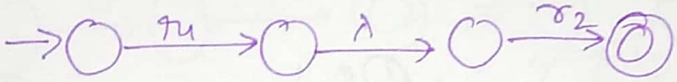
Union: $r_1 + r_2$

$r_1 = ab$

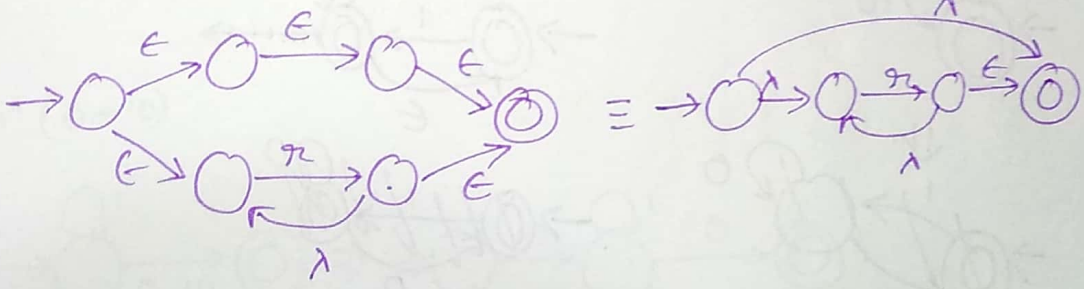
$r_2 = (a+b)^*ab$



Concatenation: $r_1 . r_2$

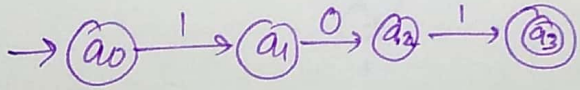


Kleen's Closure: $r^* = \lambda + r_1^+$

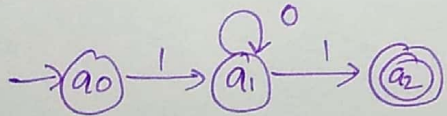


Method of Decomposition :

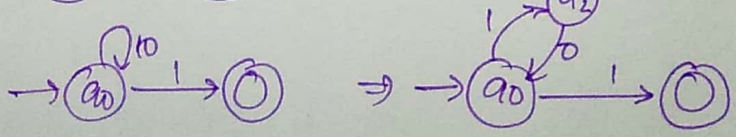
① 101



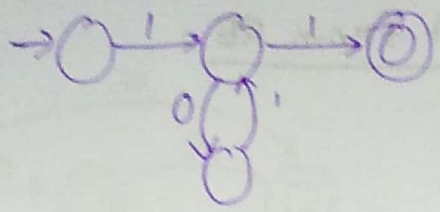
② 10^*1



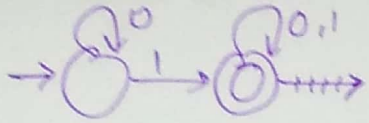
③ $(10)^*1$



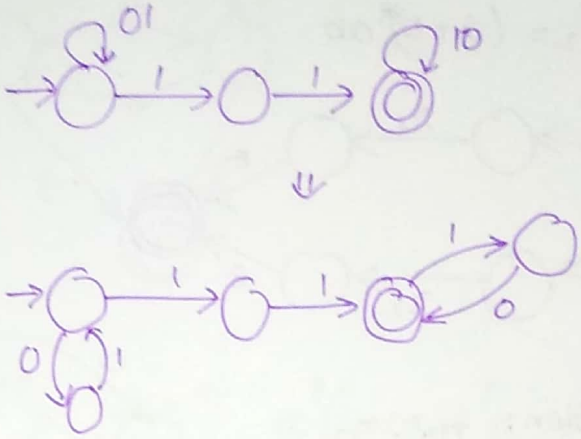
④ $1(01)^*1$



⑤ $(0^*1)(0+1)^*$

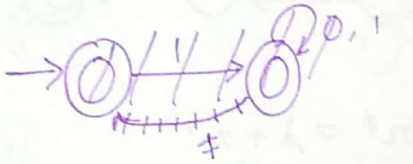


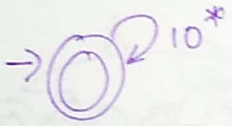
⑥ $(01)^*11(10)^*$

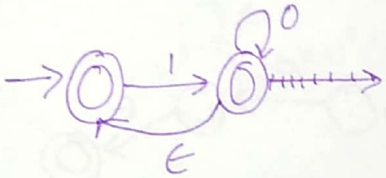


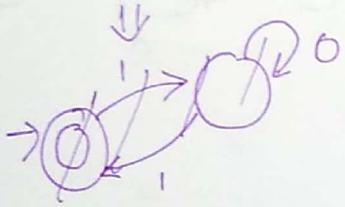
14102120

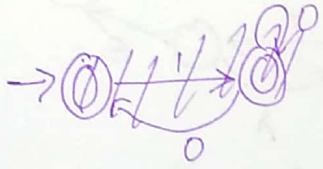
⑦ $(10^*)^*$

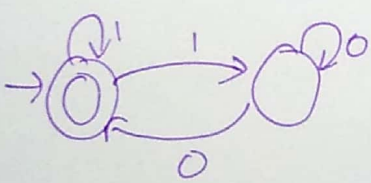




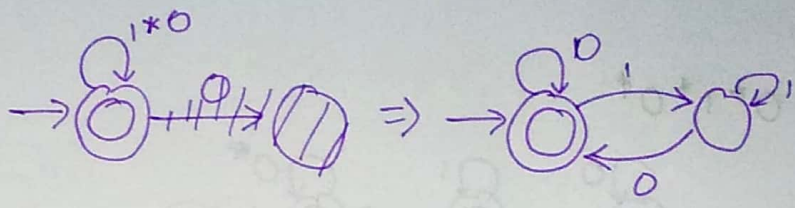




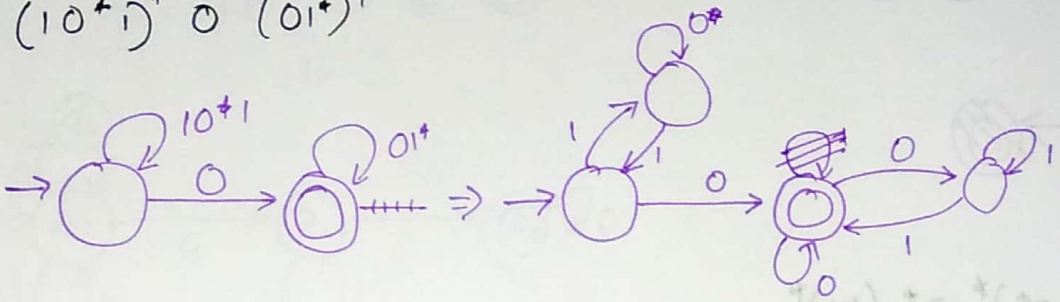




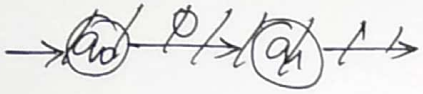
(8) $(1^*0)^*$



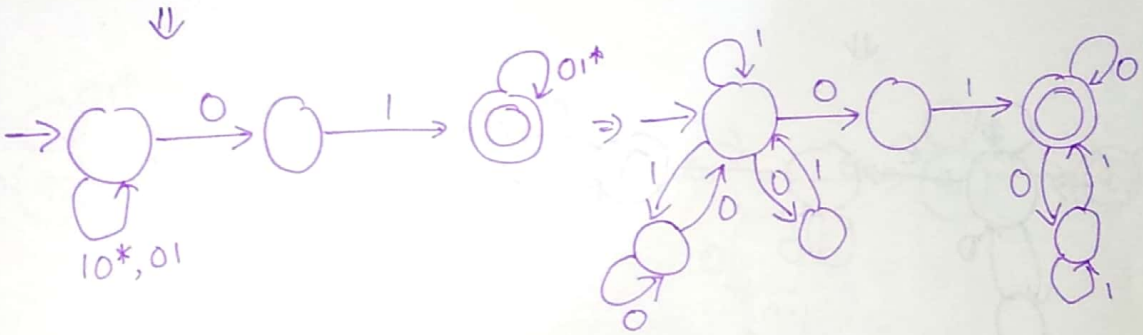
(9) $(10^*1)^*\,0\,(01^*)^*$



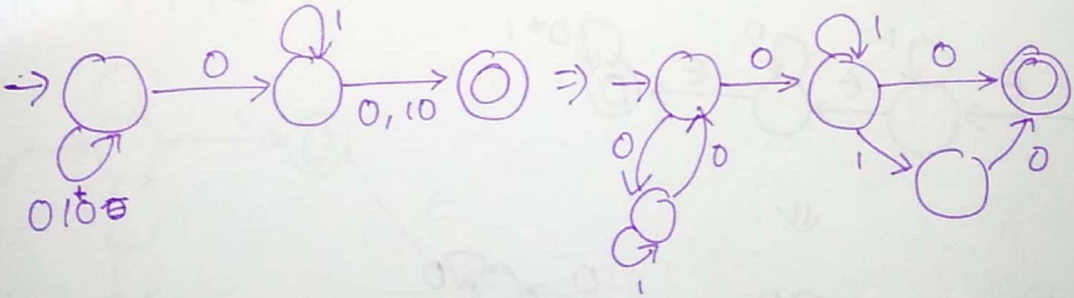
(10)



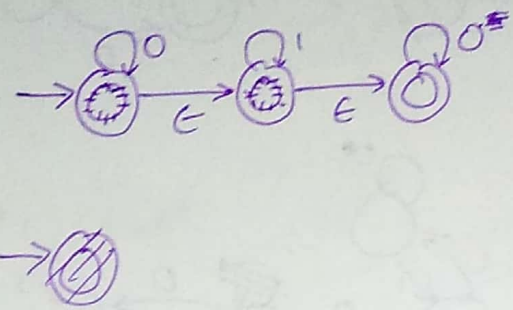
$(10^* + 01)^*\,01\,(01^*)^*$



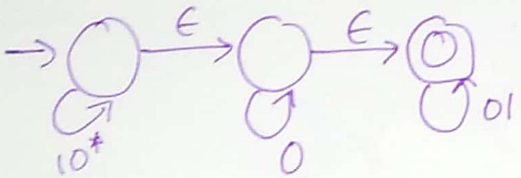
(11) $(01^*0)^*\,01^*\,(0+10)$

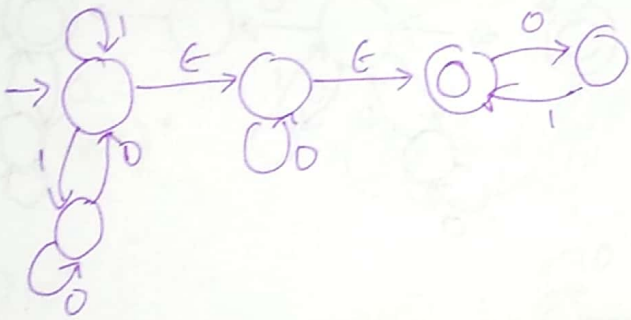


(12) $0^*1^*0^*$

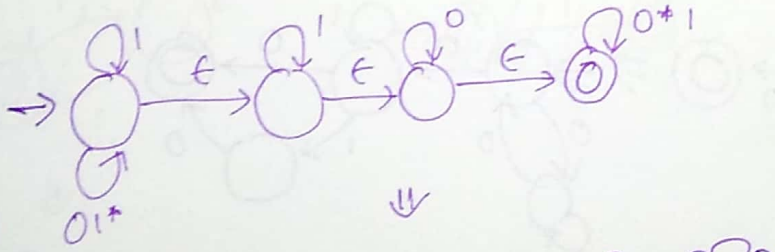


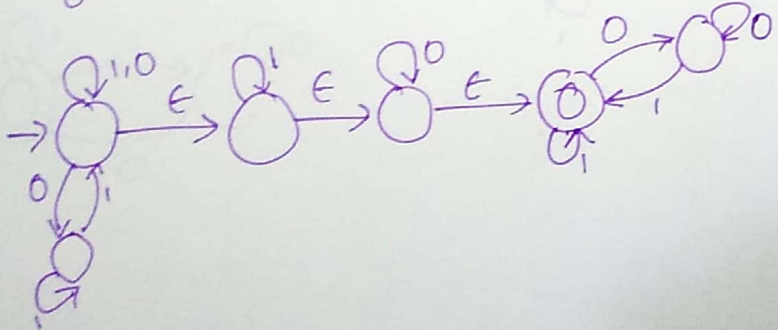
(13) $(10^*)^*\,0^*\,(01)^*$



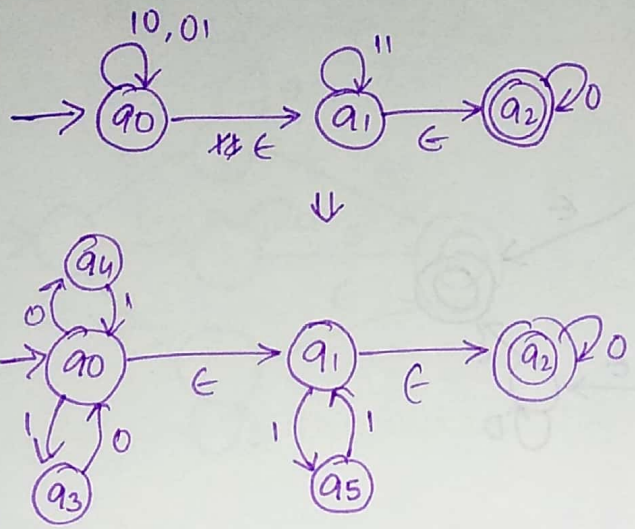


(3) $(01^* + 1)^*\,1^*0^*\,(0^*1)^*$

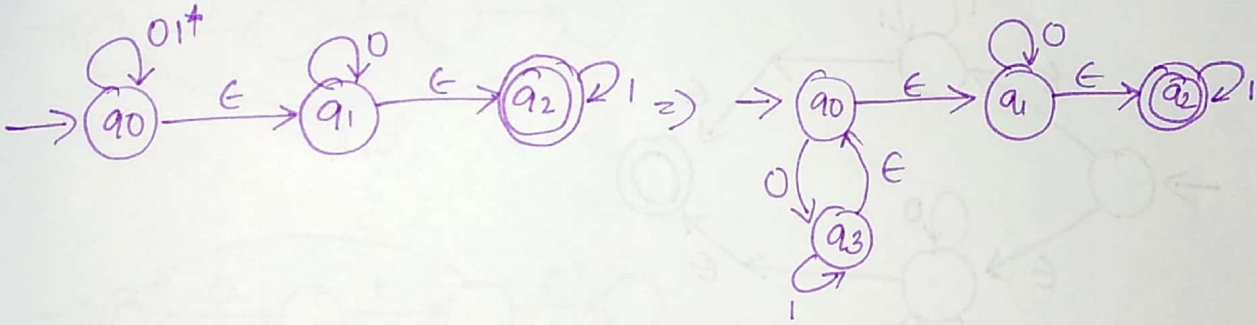




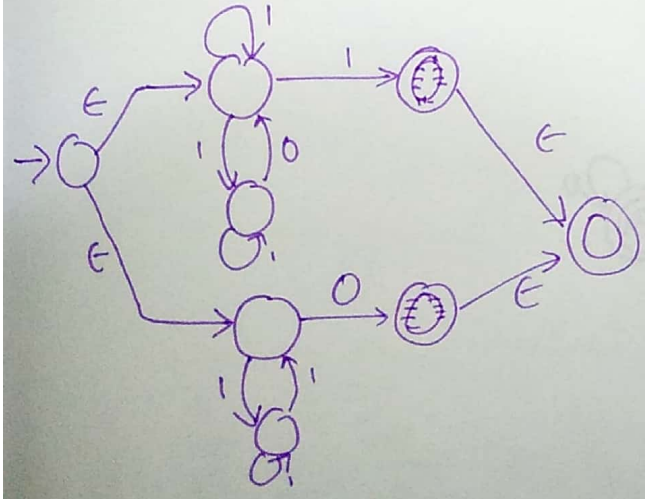
(14) $(10+01)^*(11)^*0^*$



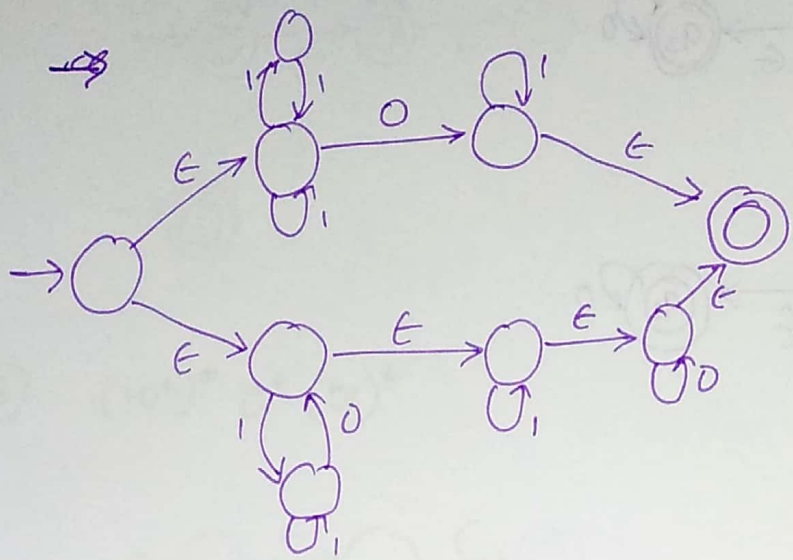
(15) $(01^*)^*\,0^*\,1^*$



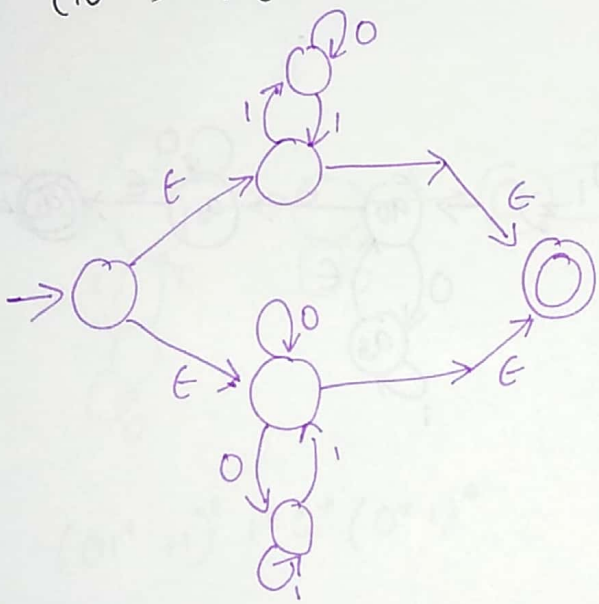
(16) $(10^*)^*1 + (10^*1)^*0$



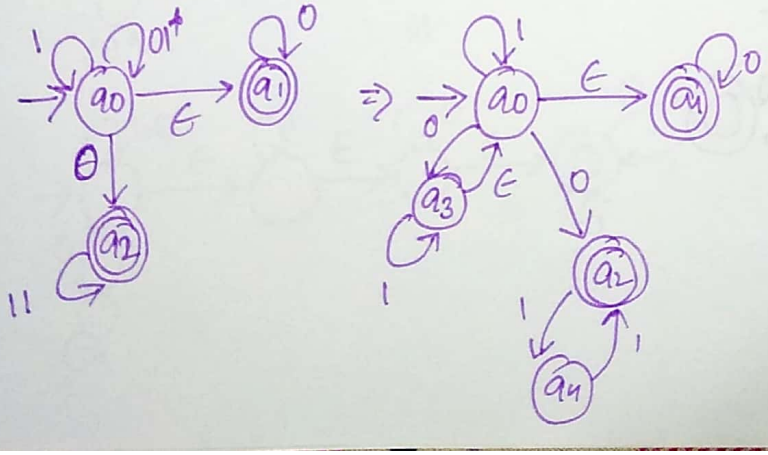
(17) $(10+1)^* 01^* + (11^*0)^* 1^* 0^+$



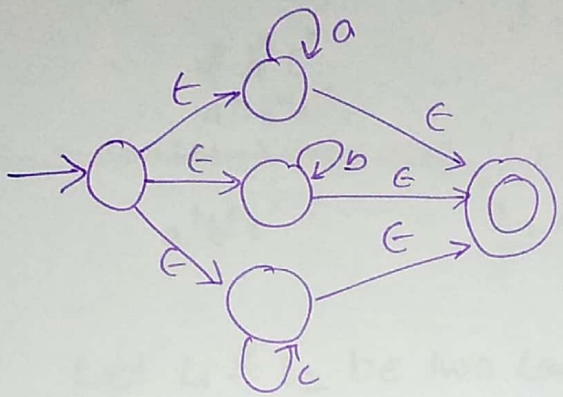
(18) $(10^*1)^* + (01^*)^*$



(19) $(01^*)^* 0^* + 1^* 0 (11)^*$

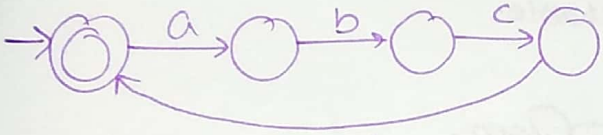


(20) $a^* + b^* + c^*$

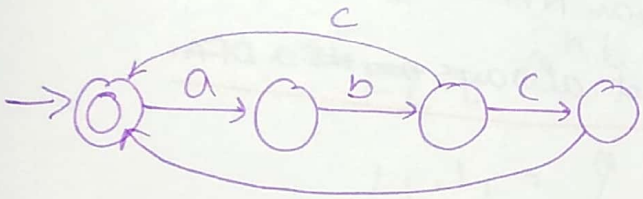


Construct NFA:

$(abc + abcd)^*$



λ, d



d

17/02/20

Closure Properties of Regular Languages:

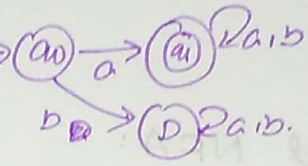
Complementation:

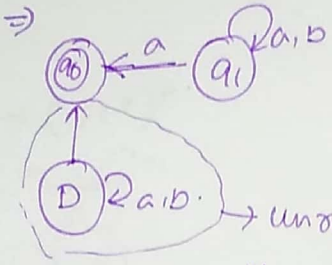
Intersection:

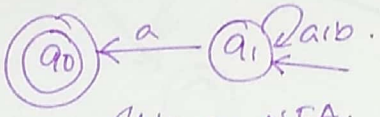
Union:

Difference:

Reverse:

DFA accepts strings start with 'a' → 

Reverse ⇒ 

⇒ 

It's an NFA.

Reverse of a DFA is not always ~~an~~ ~~NFA~~ a DFA.

Homomorphism:

Definition 4.1

Example 4.2

Example 4.3.

Q: Suppose $L_1 \cup L_2$ is regular & L_1 is finite, can we conclude that L_2 is regular? Justify your answer.

$a^n \vee a^n b^n$

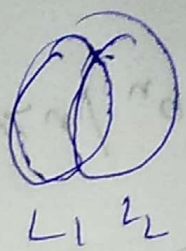
$L_2 = a^n$

$L_2 = a^n b^n$ ~~$a^n b^n$~~ a^n. $a^n b^n$

$L_1 \cup L_2 =$ ~~$a^n b^n$~~ $a^n b^n$

$L_1 \cup L_2 = L_1 + L_2 - L_1 \cap L_2$

$\Rightarrow L_2 = (L_1 \cup L_2) - L_1 + (L_1 \cap L_2)$



~~$L_2 = (L_1 \cup L_2) - (L_1 - L_2)$~~

$L_1 = (a+b)^*$

$L_2 = a^n b^n$

$L_1 \cup L_2 = (a+b)^*$

Q: Let L_1 & L_2 be two languages over the same alphabet Σ. Given that L_1 & $L_1 L_2$ both are regular. Prove or disprove L_2 must be regular

Let, $L_1 = \{$

L_2 Let L_2 is not

$L_1 = a^n b^n$

$L_1 = \phi$

$L_2 = \phi$

$L_1 \cdot L_2 = \phi$

$L_1 = \phi$

$L_2 = a^n b^n$ (NR)

$L_1 \cdot L_2 = \phi$ - R

18/02/20

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

~~Pumping~~

Pumping Lemma for Regular Languages

Let, $M = (Q, \Sigma, q_0, \delta, F)$ be a finite automata having n no. of states.

Let, L be any language and $x \in L$ where x is a string.

If the string x can be decomposed into no. of strings. $x = uvw$ such that

$|uv| \leq m$ & $|v| \geq 1$, m = pumping length

then $uv^iw \in L$

for $i = 0, 1, 2, 3 - - - -$

Q: $\{a^n b^n \mid n \geq 0\}$.

$L = \{\lambda, ab, aabb, aaabbb, - - - -\}$.

Let, $x = aaabbb$ $\quad 1 < m <$ length of x

Let, pumping length $m = 4$

Case 1: $\underbrace{aa}_{u}\ \underbrace{ab}_{v}\ \underbrace{bb}_{w}$ $\quad uv^iw$: $i=0$. $aabb \in L$

$i=2$. $aaababbb \notin L$

Case 2: $\underbrace{aaa}_{u}\ \underbrace{b}_{v}\ \underbrace{bb}_{w}$ $\quad uv^iw$: $i=0$: $aaabb \notin L$

$i=1$: $aaabbb \in L$ $\quad i=2$: $\notin L$

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

Q: $L = \{ww^R, w \in \{a,b\}^*\}$

Let $L = \{\overset{\in}{aa}, abba, abbbba, \cdots\}$.

Let, $m = 5$ (pumping length)

Case 1: $\underbrace{a}_{u}\underbrace{bbbb}_{v}\underbrace{a}_{w}$

$i=0$: $\in L$
$i=1$: $\in L$
$i=2$ $\in L$
⋮

Case 2: $\underbrace{a}_{u}\underbrace{bbbb}_{v}\underbrace{a}_{w}$

$i=0$: $aba \notin L$
$i=1$: $abbbba \in L$
$i=2$: $abbbbbba \notin L$.

Q: $L = \{a^n b^{n+1} / n \geq 0\}$ $L = \{\overset{b}{a}b, abb, aabbb, aaabbbb, \cdots\}$.

Let, $m = 5$

Case 1: $\underbrace{aa}_{u}\underbrace{abbb}_{v}\underbrace{b}_{w}$

$i=0$: $aabb \notin L$
$i=1$: $aaabbbb \in L$
$i=2$: $aaabbabbbb \notin L$

Case 2: $\underbrace{aaa}_{u}\underbrace{bb}_{v}\underbrace{b}_{w}$

$i=0$: $aaabb \notin L$
$i=1$: $aaabbbb \in L$
$i=2$: $aaabbbbb \notin L$.

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

Let,

Q: $\delta^*(1, ba) = ?$

δ	a	b	ϵ	ϵ^*
1	5	—	4	1,4,3
2	1	—	—	2
3	—	2	—	3
4	—	7	3	4,3
5	—	—	1	5,4,1,3
6	—	5	4	6,4,3
7	6	—	—	7

$\delta(1, ba)$

$\delta(\{1,4,3\}, ba)$.

$\delta(\{7,2\}, a)$

$= \{6,4,3,1\}$.

ϵ^* b ϵ^*

$1 - 1,4,3 - \phi,7,2 - 7,2$

ϵ^* a ϵ^*

$2,7 - 2,7 - 1,6 - 1,4,3,6.$

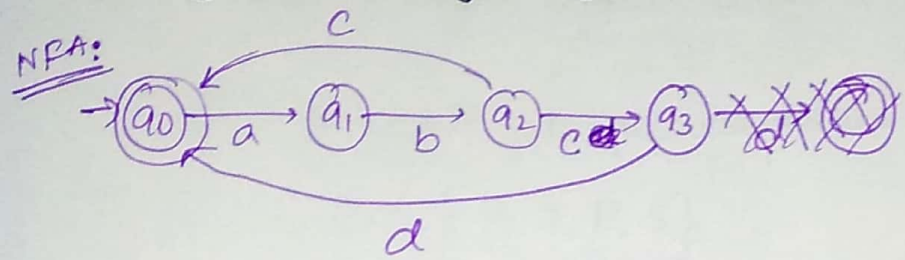
$\therefore \delta^*(1, ba) = \{6,4,3,1\}$.

19/02/20

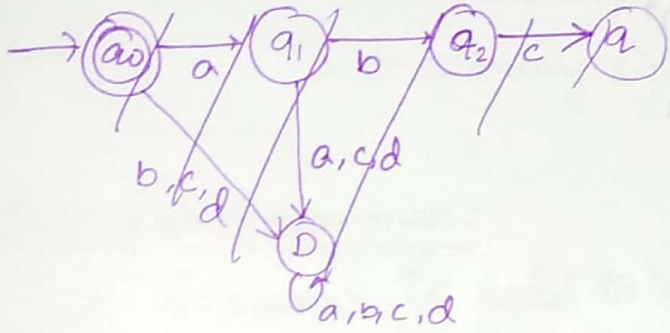
~~NFA & DFA:~~

Q: $\{abc, abcd\}^*$ NFA

NFA:

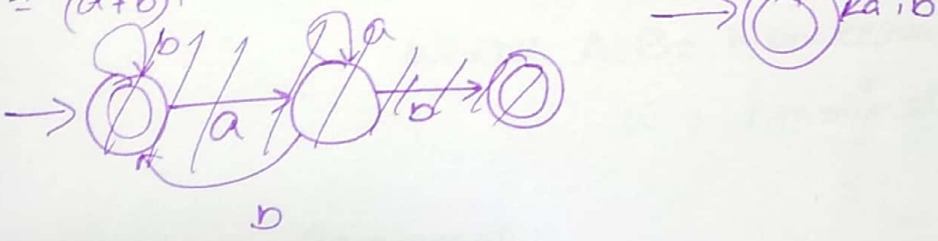


~~DFA:~~



: $r = (b^+a^+ + ab + \lambda)^*$. DFA

$= (a+b)^*$



$L_1 = \{ uwu \mid v, w \in \Sigma^* \; |u| \leq 2 \}$.

~~$(a+b)^2$~~

~~$(aa)^* \; b(bb)^*$~~

$(b^*a^* + bba + \lambda)^*$

$= ((b^*a^*)^* \cdot (bba)^*)^*$ $\quad (P^*Q^*)^* = (P^* + Q^*)^*$

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

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

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

$= (a+b)^*$