unit-4

Pushdown Automata

→ A PDA consists of three components

(i) Input tape
(ii) Finite control
(iii) Stack structure

→ Input tape consist of a linear configurati of cells. Each of which contains a charac from the input alphabet. the tape can be moved one cell at a time to the left

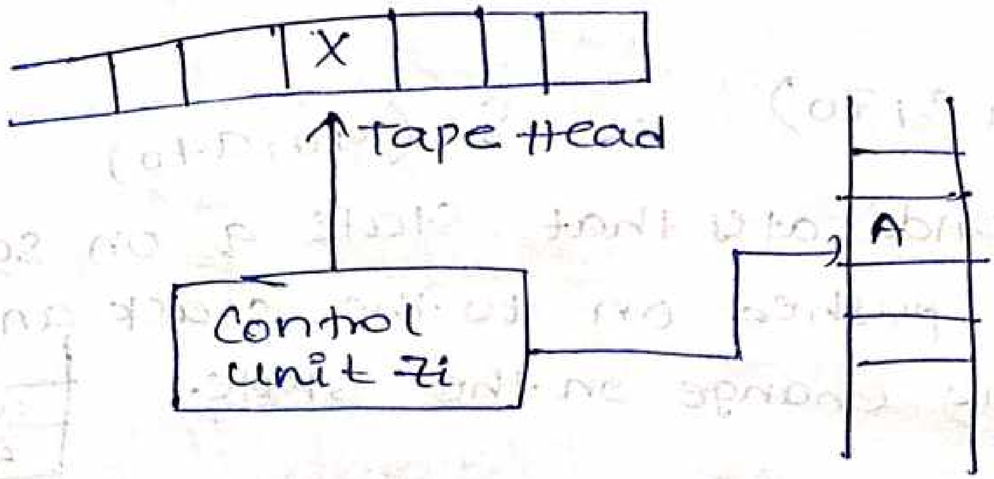
→ the stack is also a sequence structure tha as first element and grows on either directions from the other end.

→ Control unit has some pointer (head) whic points the current symbol that is to be read

→ The head position over the current stack element can read and write special stack characters from that position.

→ the current stack element is always the top element of the stack, hence the name stack

The control unit contains both tape head and stack head and finds itself at any movement on a particular state



⇒ A finite state PDA is 7 tuple machine where $M = \{Q, \Sigma, \delta, T, q_0, z_0, F\}$

Q = Finite set of states

Σ = A finite set of input alphabets.

T = A finite set of stack alphabets

q_0 = start | initial state

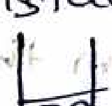
F = set of Final states

$$\boxed{\delta = Q \times (\Sigma \cup \{\epsilon\}) \times T \longrightarrow Q \times T^*}$$

z_0 = initial stack symbol

$z_0 \in T$.

z_0 will be be default in stack



⇒ A move on PDA indicates

(i) A Element may be added to the stack

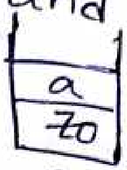
(ii) A Element may be deleted from the stack

(iii) there may (or) may not be change of stack (do nothing operation)

Operations

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

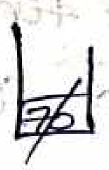
This indicates that state q on seeing a, a is pushed on to the stack and there is no change on the state.



only top two are declared

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

This indicates that on the state q on seeing a the current top symbol z_0 is deleted from the stack



→ if ϵ indicates z_0 (top of stack) will be popped.

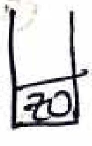
(iii) $\delta(q_0, a, z_0) = (q_1, az_0)$

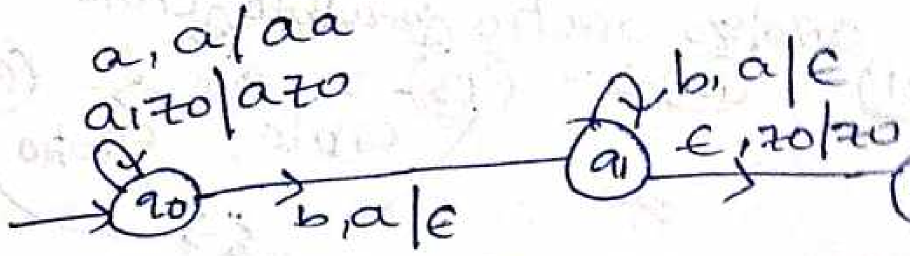
this indicates that on the state q on seeing a, a is pushed on to the stack and the state is changed to q_1.

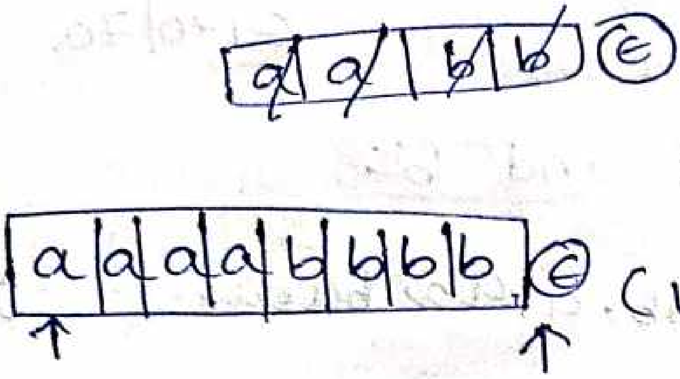
Example:-

Q) Design the PDA which accepts the Language $L = \{a^n b^n, n \geq 1\}$

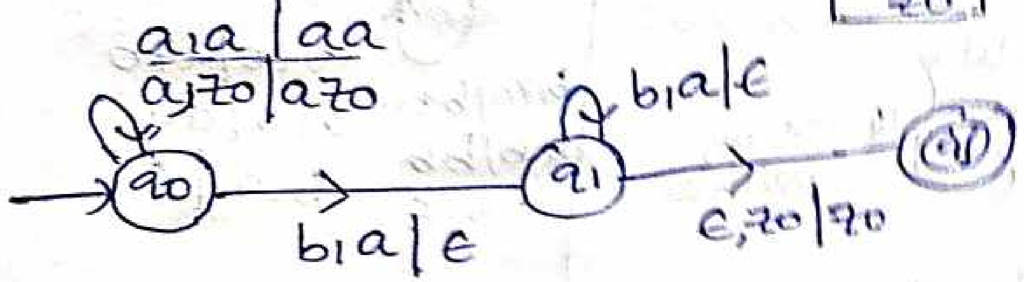
$L = \{ab, aabb, aaabbb \ldots\ldots\}$





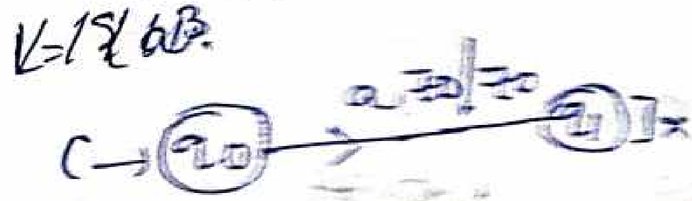


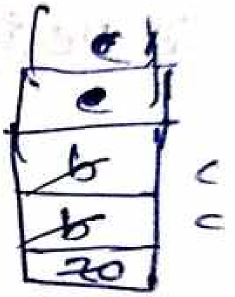
(L—R)

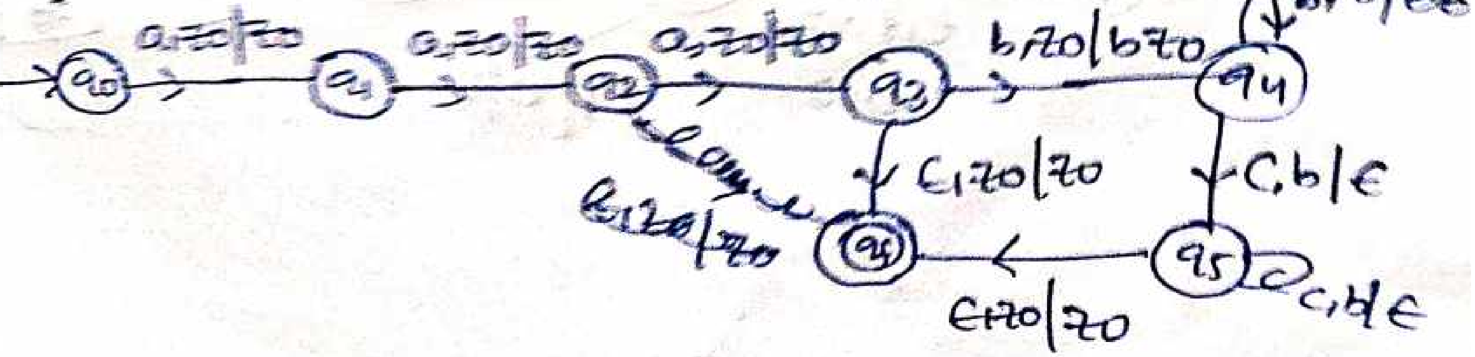


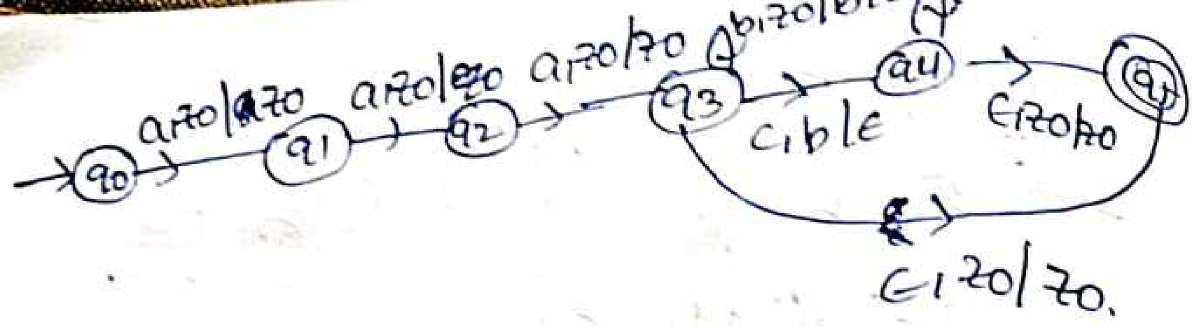
Q) Design the PDA which accepts the Language

$L = \{a^3 b^n c^n \mid n \geq 0\}$



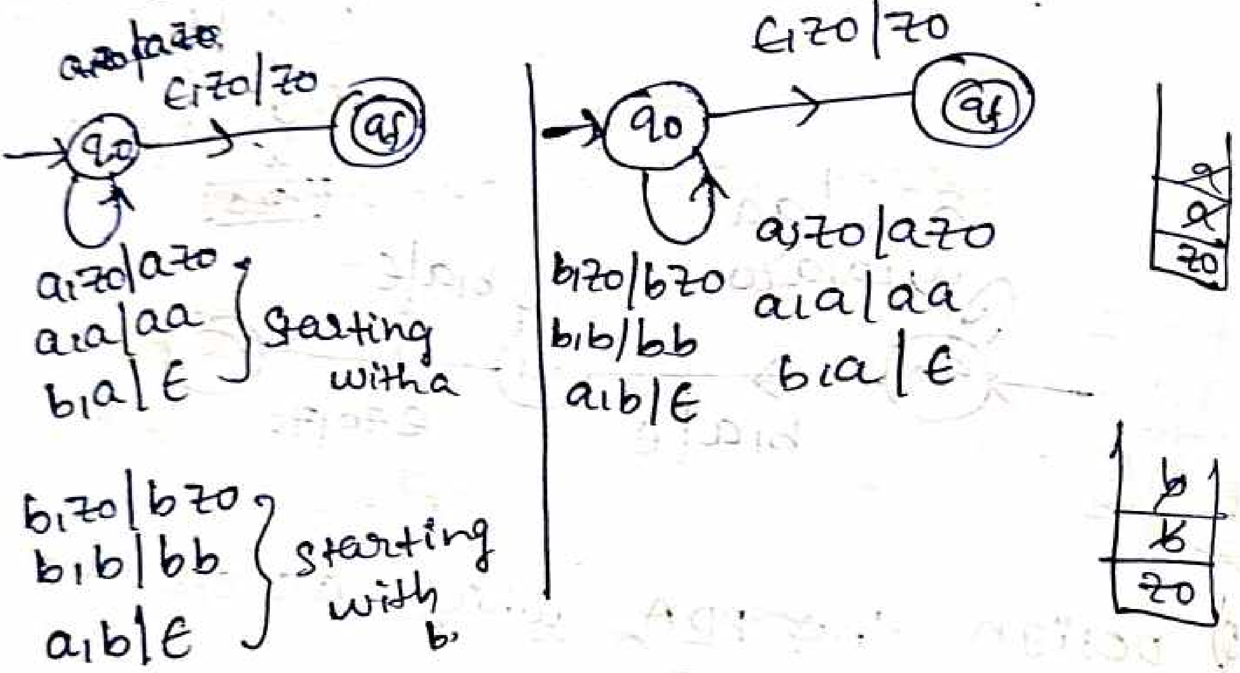






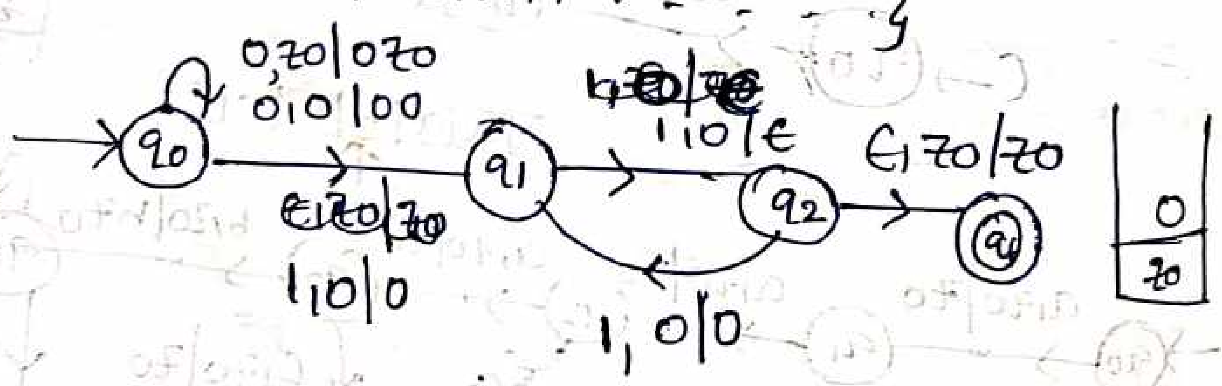
Q) Equal No. of a's and b's

$L = \{\epsilon, ab, aa, bb, abab, baba, \ldots\}$

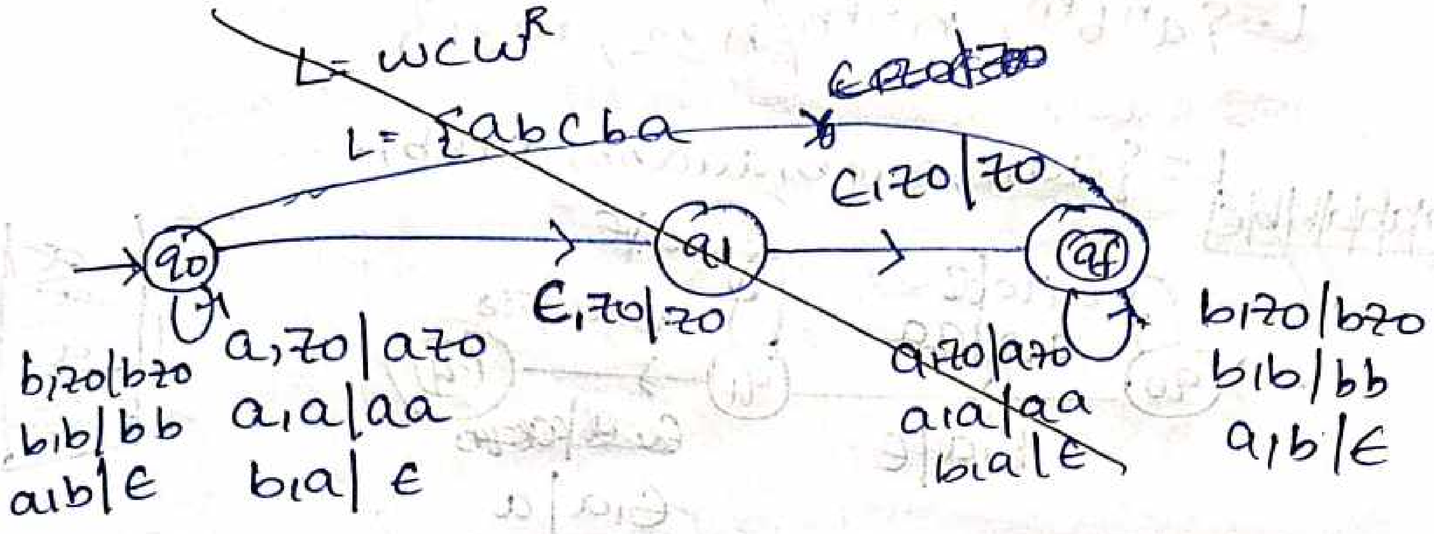


Q) Design a PDA that accepts $L = \{0^n 1^{2n}, n \geq 1\}$

$L = \{011, 001111, \ldots\}$



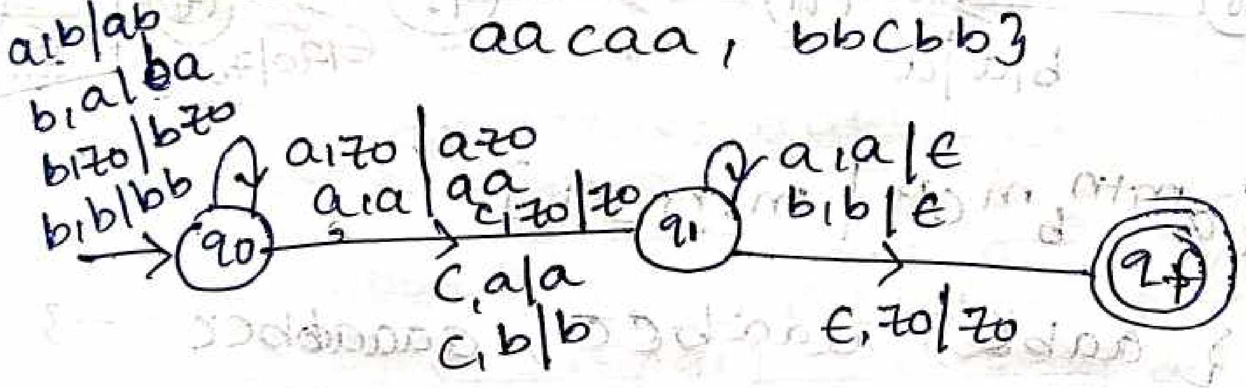
Q) *Imp $L = wcw^R$ inputs $= \{a, b\}$

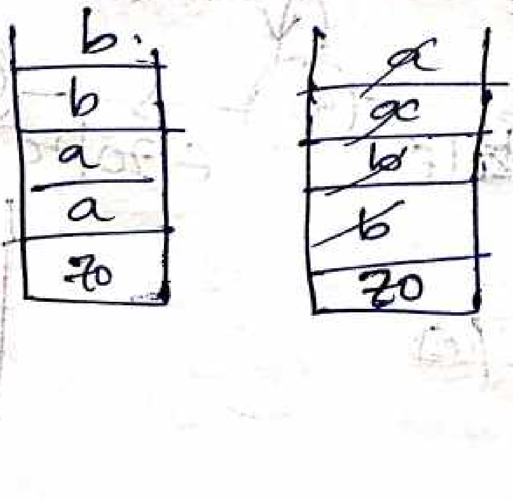


Q) ***Imp $w \in (a+b)^*$

$L = wcw^R$ {a,b}

$L = \{\epsilon, aca, bcb, abcba, aabcbaa, bbacabb$
$aacaa, bbcbb\}$

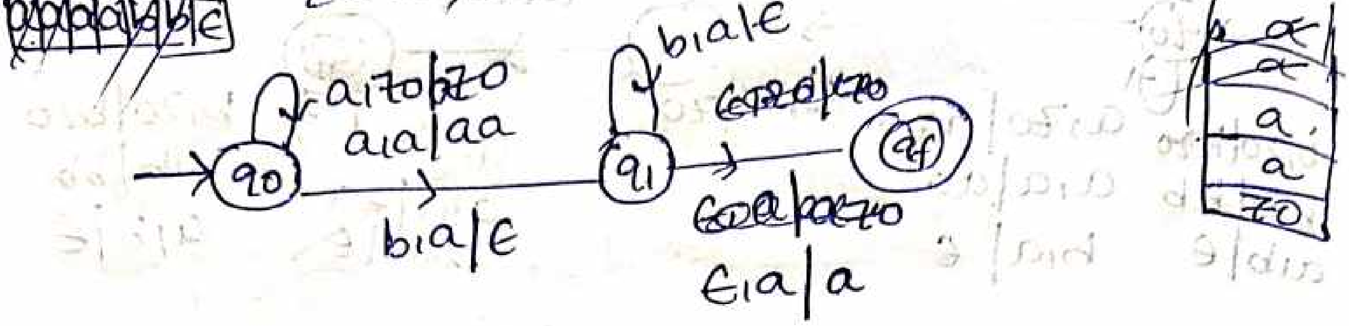




Q) $\Sigma = \{a^n b^n\}$,

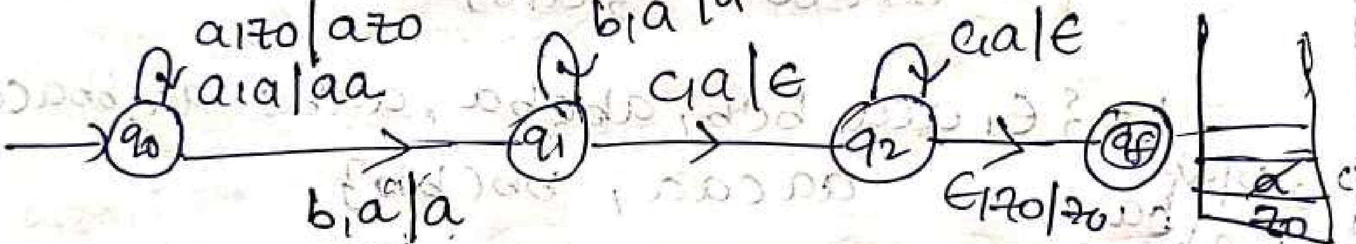
$L = \{a^n b^m, \; n > m \;/\; n \geq 2, \; m \geq 1\}$

$L = \{aab, aaab, aaabb, aabb, \dots$



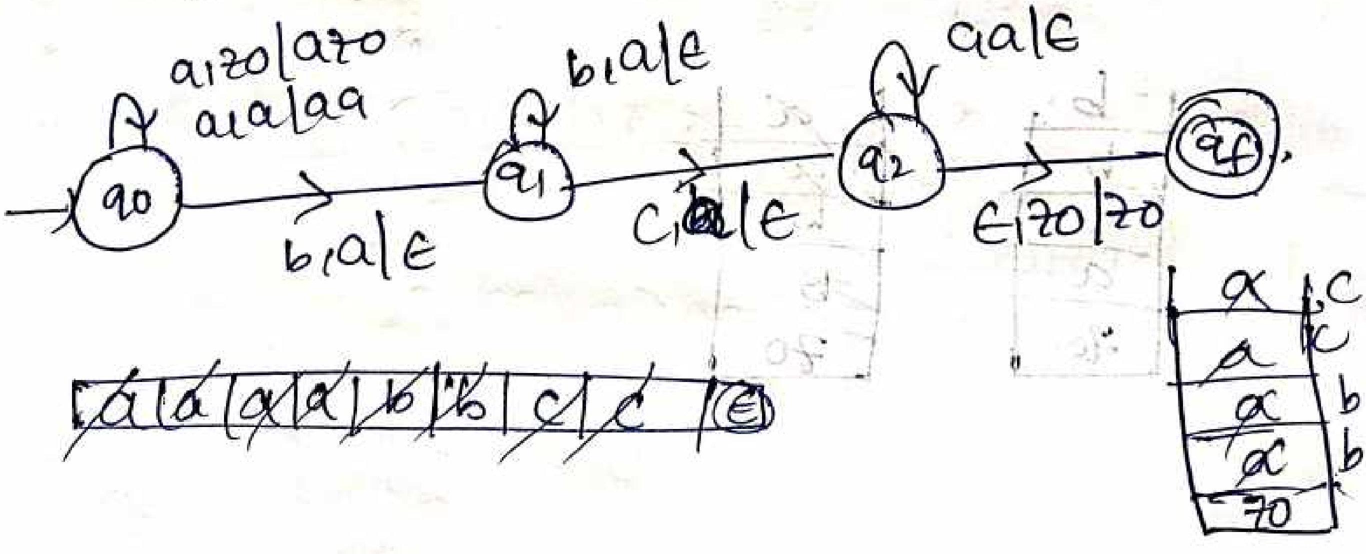
Q) $L = \{a^n b^m c^n \mid n, m \geq 1\}$

$L = \{$ ~~acb~~ $abc, abbc, aabbbc, \dots\}$



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

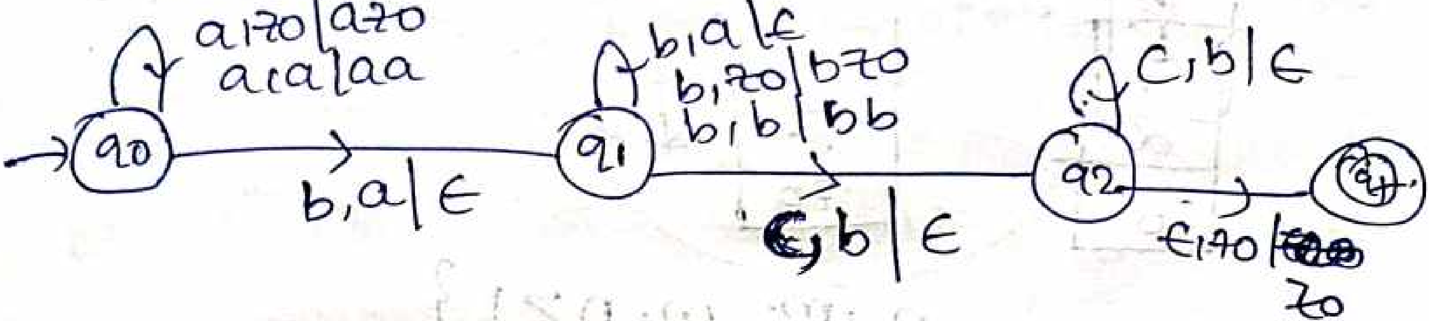
$L = \{aabc, aaabcc, aaaabbcc \dots\}$



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

$= \{ abbc,\ aabbbbcc,\ abbbcc, \ldots \}$

~~a~~	~~a~~	~~b~~	~~b~~	~~b~~	~~b~~	~~c~~	~~c~~	ϵ

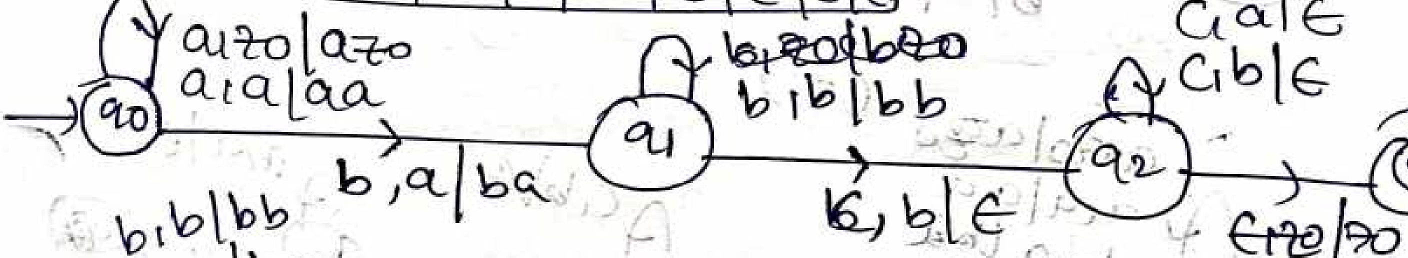


(or)

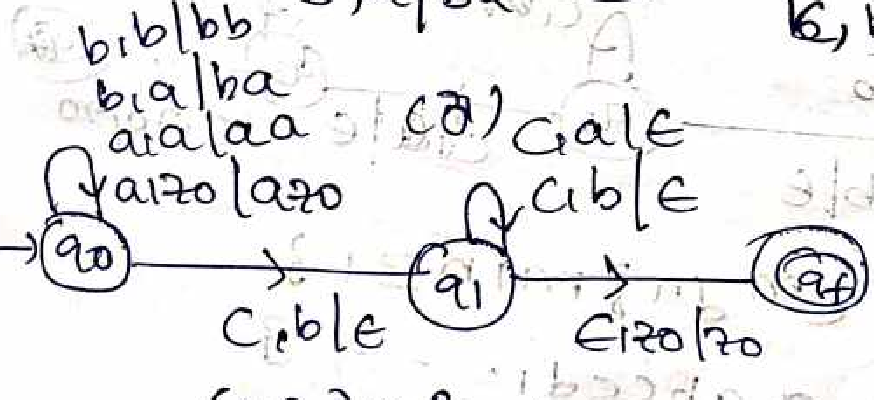
Q) $L = \{a^n b^m c^{m+n} \mid n, m \geq 1\}$

$\{abcc,\ aabccc,\ aabbcccc, \ldots \}$

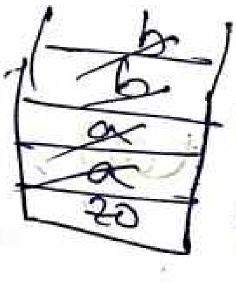
a	a	b	b	c	c	c	c



(or)

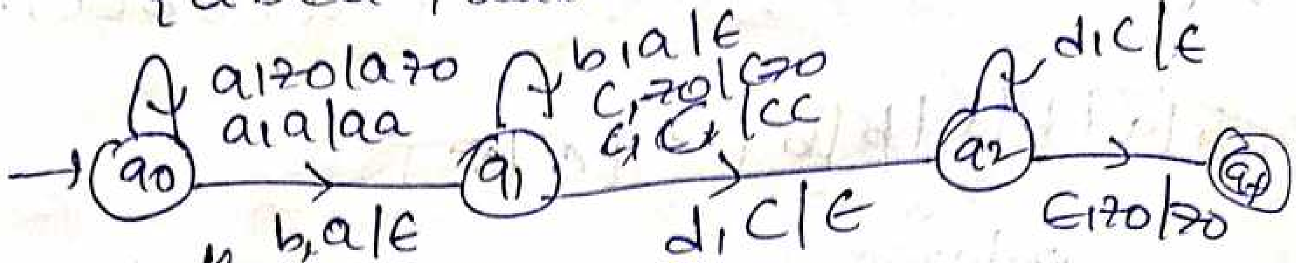


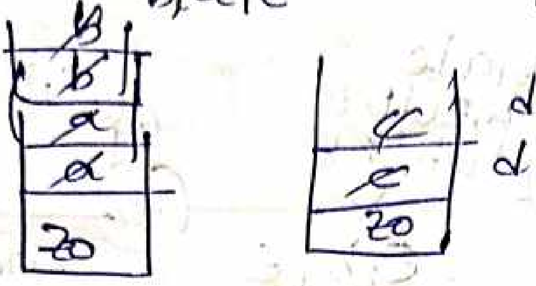
(Minimized Automata)



Q) $L = \{a^m b^m c^n d^n, m, n \geq 1\}$

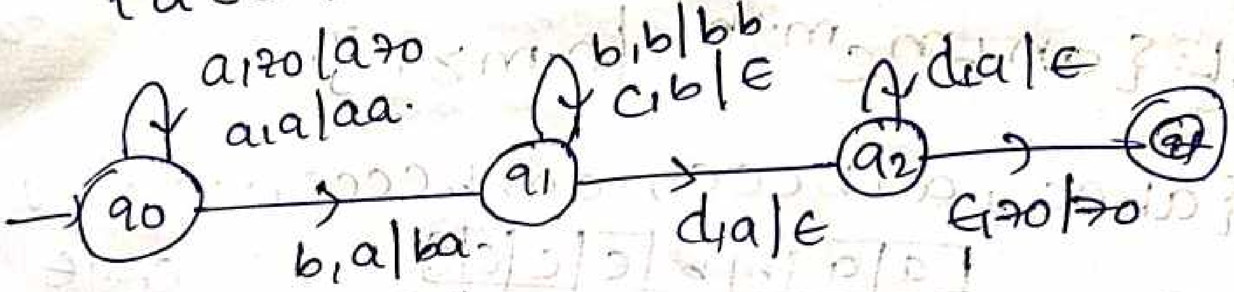
$\{abcd, aabbccdd, \ldots\}$



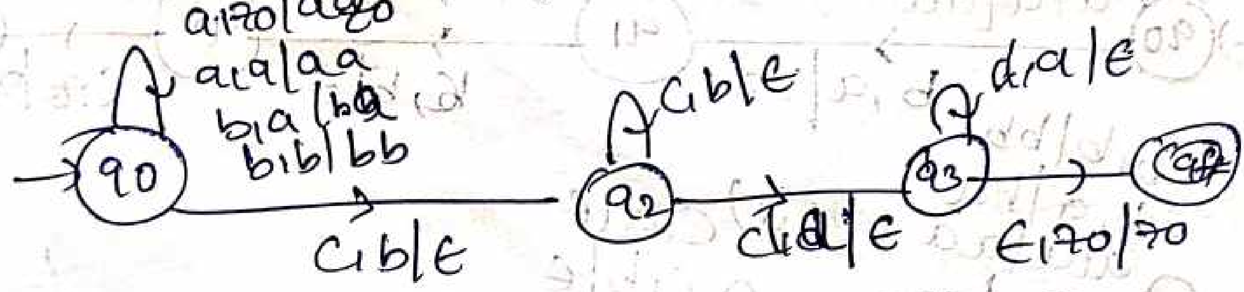


Q) $L = \{a^m b^n c^n d^m, m, n \geq 1\}$

$\{abcd, aabbccdd, aabcdd, \ldots\}$

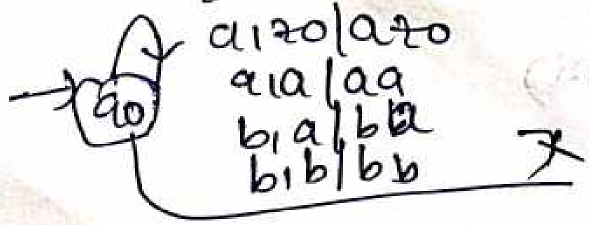


(or)



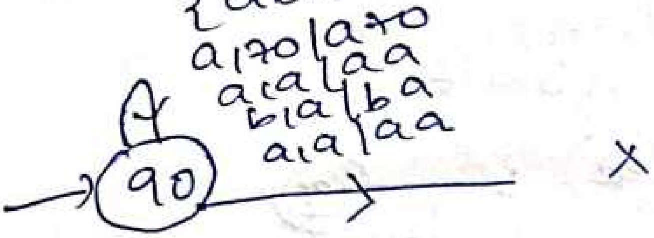
Q) $L = \{a^m b^n c^m d^n, m, n \geq 1\}$

$\{abcd, aabccd, \ldots\}$



Q) $L = \{a^m b^n c^m d^n, \ m, n \geq 1\}$

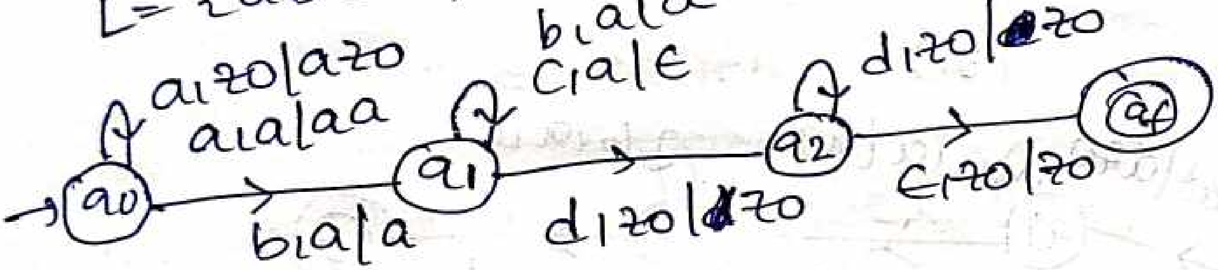
$\{abcd, aabccd, \ldots\}$.



PDA cannot be drawn because comparison is Not possible & No matching for push and pop.

Q) $L = \{a^m b^i c^m d^k, \ i, m, k \geq 1\}$

$L = \{abcd, aabccd, \ldots\}$

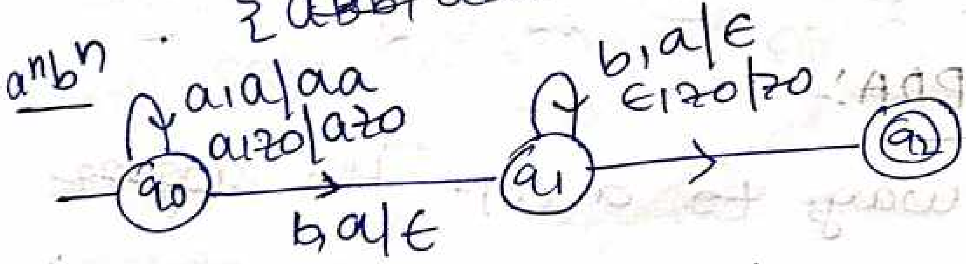


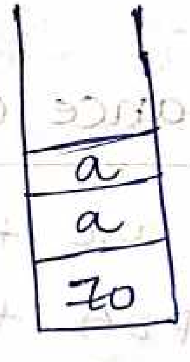
Q) $L = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$

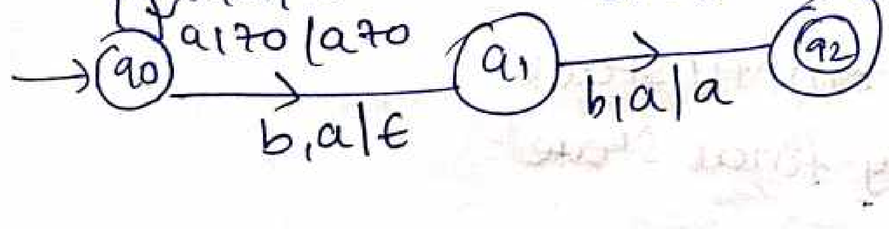
$L = \{ab, aabb, aaabbb, \ldots\} \cup$

$\{$~~abb, aabbbb, aba~~ ... $\}$.

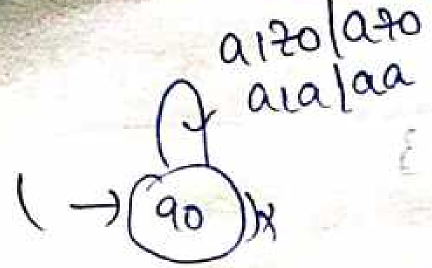
$\{$ ~~abb, aabbbb~~

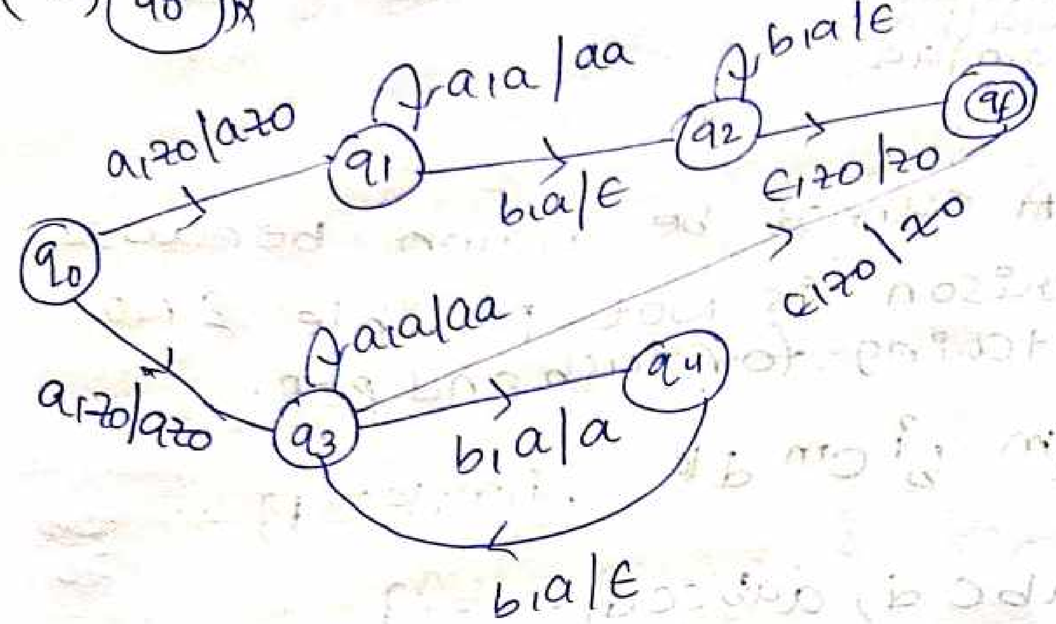




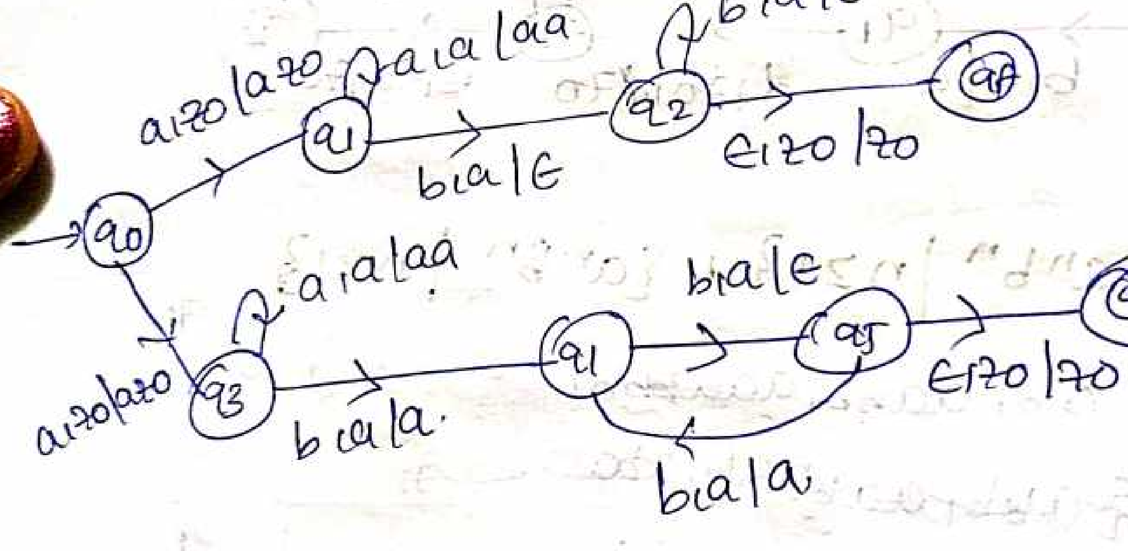


21





(ग)



Acceptance of PDA:-

there are two ways to accept the language by PDA they are

(i) Accepted by empty stack

(ii) Accepted by final state

Accepted by empty stack:-

the given language accepted by empty stack to be defined as $L(M)=\{W / \delta(q_0,w,z_0) \vdash (P,\epsilon,\epsilon)$ for some p in $Q\}$ that is if the stack becomes empty after scanning Entirestring then it is accepted by PDA otherwise not accepted.

Ex:-

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

Accepted by final state:-

the given language accepted by final state defined as

$$L(M) = \{W \mid \delta(q_0,w,z_0) \vdash (P,\epsilon,\lambda) \text{ for some } p \in F \text{ and } \lambda \in \Gamma\}$$

that is even though stack is Not empty after scanning input string If the finite control reaches to final state then it is accepted otherwise not accepted.

Ex:-

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

Types of PDA:-

(i) DPDA (Deterministic push down Automata)
(ii) NPDA (Non-Deterministic push down Automata)

⇒ with PDA that has atmost one chance of move in any state is called a DPDA

⇒ NPDA provides Non deterministic in the move Defined

⇒ DPDA are useful in programming languages

Ex:- parsers are used in yet another compiler compiler (YACC) are determined in PDA's

DPDA:-

A DPDA is seven tuple Machine

$$M=\{Q, \Sigma, q_0, \Gamma, Z_0, \delta, F\}$$

Q = set of finite states that are non-empty

Σ = set of input alphabets

q_0 = initial state.

Γ = Finite set of stack alphabets

Z_0 = initial stack symbol

δ = Transition function | mapping function used for mapping current state to Next state

F = set of Final states

→ If a transition denotes a unit transition for each input then PDA is said to be DPDA.

Ex:-

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

$L = \{wcw^R, w = (a+b)^+\}$

NPDA:-

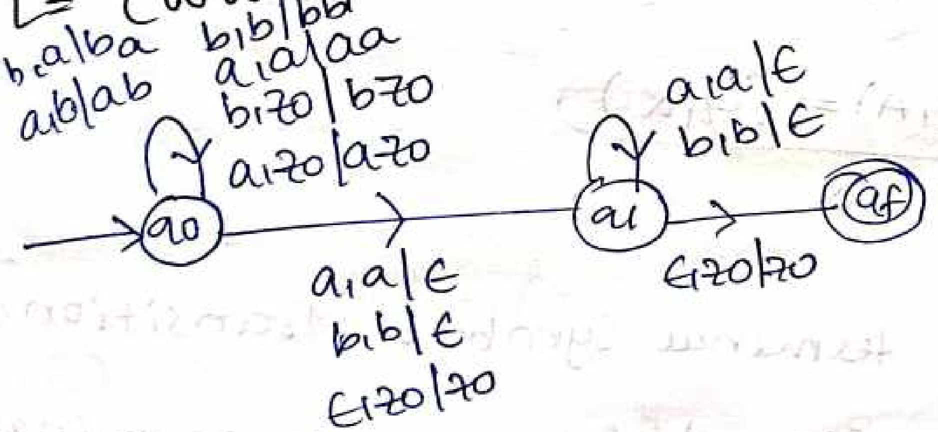
A NPDA is seven tuple machine

$M = \{Q, \Sigma, q_0, \Gamma, Z_0, \delta, F\}$

→ If a transition denotes more than one transition for a particular input symbol then PDA is said to be NPDA.

Ex:- $L = (ww^R / w = (a+b)^*)$

Q) $L = (ww^R / w = (a+b)^+)$



aa	aa
ab	ba
ba	ab
bb	bb

Equivalence of PDA's and context free grammars

(i) conversion of context free grammar to PDA:-

⇒ For constructing a PDA from given context free grammar, It is necessary to convert the context free grammar to some Normal Forms like GNF.

⇒ For a converting given context free grammar to PDA. By this method a necessary condition is first symbol on right hand side of the production rule must be terminal symbol this rule that can be used to obtain PDA from context free grammar.

Rules:-

(i) For non-terminal symbols, transition equation

$$\boxed{\delta(q, \epsilon, A) = (q, \alpha)}$$

$A \to \alpha$

(ii) For Each terminal symbol transition of

$\delta(q, a, a) = (q, \epsilon)$ for every terminal symbol ~~a~~ is given CFG

Q) construct a PDA equivalent to the following grammar

$S \rightarrow 0BB$

$B \rightarrow 0S$

$B \rightarrow 1S$

$B \rightarrow 0$

the given grammar is in minimized format

from rule (1) $A \rightarrow \alpha$

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

$S \rightarrow 0BB$ $\delta(q, \epsilon, S) = (q, 0BB)$

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

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

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

From rule (2)

inputs

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

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

(or)

$\delta(q, 0, S) = (q, BB)$

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

$\delta(q, 1, B) = (q, S)$

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

Conversion of PDA to CFG:-

Construction rules for converting PDA to CFG:-

rules for following PDA:-

⇒ we will construct a grammar G, such that $L(G) = L(M)$.

rules:-

(i) the productions for start symbol S are given by

$S \rightarrow [q_0, z_0, q]$ for each state q in Q

(ii) Each move that pops a symbol from the stack with the transition as

$$\boxed{\delta(q, a, z_i) = (q_1, \epsilon)}$$

includes a production as

$[q, z_i, q_1] \rightarrow a$ for q_1 in Q

(iii) Each move that does not pop the symbol from stack with transition as transition of $\delta(q, a, z_0) = (q_1, z_1 z_2 z_3 z_4 \ldots\ldots)$ includes a production as

$$[q, z_0, q_m] \rightarrow a [q_1 z_1 q_2] [q_2 z_2 q_3] [q_3 z_3 q_4] \ldots\ldots [q_{m-1} z_{m-1} q_m]$$

Ex:
Give the equivalence context free grammar for the following PDA.

$M = \{\{q_0, q_1\}, \{a, b\}, \{z, z_0\}, \delta, q_0, z_0, \phi\}$

where δ is defined by

$\delta(q_0, b, z_0) = (q_0, zz_0)$

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

$\delta(q_0, b, z) = (q_0, zz)$

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

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

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

$M = \{\{q_0, q_1\}, \{a, b\}, \{z, z_0\}, \delta, q_0, z_0, \phi\}$

$S \rightarrow [q_0, z_0, q_0] \mid [q_0\ z_0\ q_1]$

(i) $[q_0, z_0, q_0] \rightarrow b\,[q_0, z, q_0]\,[q_0, z_0, q_0]$

$b\,[q_0, z, q_1]\,[q_1, z_0, q_0]$

$[q_0, z_0, q_1] \rightarrow b\,[q_0, z, q_0]\,[q_0, z_0, q_1]$

$b\,[q_0, z, q_1]\,[q_1, z_0, q_1]$

$\left[\because \begin{matrix} q_0 \rightarrow q_0 \rightarrow q_0 \\ q_0 \rightarrow q_1 \rightarrow q_0 \end{matrix}\right]$

(ii) $[q_0, z_0, q_0] \rightarrow \epsilon$

(iii) $[q_0, z, q_0] \rightarrow b\,[q_0, z, q_0]\,[q_0, z, q_0]$

$b\,[q_0, z, q_1]\,[q_1, z, q_0]$

$[q_0, z, q_1] \rightarrow b\,[q_0, z, q_0]\,[q_0, z, q_1]$

b $[q_0, z, q_1]$ $[q_1, z, q_1]$

(iv) $[q_0, z, q_0] \rightarrow a[q_1, z, q_0]$

$[q_0, z, q_1] \rightarrow a[q_1, z, q_1]$

(v) ~~$[q_0, z_0, q_0] \rightarrow b[q_0, z, q_0]$~~

$[q_0, z, q_1] \rightarrow b[q_0, z, q_1]$

v) ~~$[q_1, z_0,$~~

v) $[q_1, z, q_1] \rightarrow b$

(vi) $[q_1, z_0, q_0] \rightarrow a[q_0, z_0, q_0]$

$[q_1, z_0, q_1] \rightarrow a[q_0, z_0, q_1]$

Q) convert PDA to context free grammar

PDA is given by

$\{\{p, q\}, \{0, 1\}, \{x, z\}, \delta, q, z, \phi\}$

$\delta(q, 1, z) = \{(q, xz)\}$

$\delta(q, 1, x) = \{(q, xx)\}$

$\delta(q, \epsilon, x) = \{(q, \epsilon)\}$

$\delta(q, 0, x) = \{(p, x)\}$

$\delta(p, 1, x) = \{(p, \epsilon)\}$

$\delta(p, 0, z) = \{(q, z)\}$

$S \rightarrow [p, z, q] \mid [p, z,$

(i) $[q, z, q] \rightarrow 1 [q \; z \; q]$

$[q, z, p] \rightarrow 1 [q \; z \; p]$

(ii) $\cancel{[q, x, q]} \rightarrow 1 [$

$[q, x, p$

$S \rightarrow [q, z, q] \mid [q, z, p]$

(i) $[q, z, q] \rightarrow 1 [q, x, q] \; [q, z, q]$

$/ 1 [q, x, p] \; [p, z, q]$

$[q, z, p] \rightarrow 1 [q, x, q] [q, z, p]$

$/ 1 [q, x, p] [p, z, p]$

(ii) $[q, x, q] \rightarrow 1 [q, x, q] \; [q, x, q]$

$/ 1 [q, x, p] \; [p, x, q]$

$[q, x, p] \rightarrow 1 [q, x, q] \; [q, x, p]$

$/ 1 [q, x, p] \; [p, x, p]$

(iii) $[q, x, q] \rightarrow \epsilon$

(iv) $[q, x, q] \rightarrow 0 [p, x, q]$

$\rightarrow 0 [p, x, p]$

(v) $[p, x, p] \rightarrow 1$

(vi) $[p, z, p] \to 0[q, z, q]$

$[p, z, q] \to 0[p, z, p]$

$[q\ z\ q] = A$

$[q\ z\ p] = B$

$[p\ z\ q] = C$

$[p\ z\ p] = D$

$[q\ x\ q] = E$

$[q\ x\ p] = F$

$[p\ x\ p] = G$

$[p\ x\ q] = H$

$S \to A \mid B$

$A \to 1EA \mid 1FC$

$B \to 1EB \mid 1FD$

$E \to 1EE \mid \cancel{1FH} \mid 1FH$

$F \to 1EF \mid 1FG$

$E \to \epsilon \mid \cancel{0H} \mid 0H$

$F \to 0G$

$G \to 1$

$C \to 0A$

$\cancel{G \to 0}$

$D \to 0B$

Final gramm

$E \to 1EE \mid$ [illegible]

$F \to 1EF \mid 1FG$

$E \to \epsilon \mid$ [illegible]

$F \to 0G$

$G \to 1$

simplifying the grammar

in the above grammar first identify the non terminals that are not defined and eliminate the productions that refers to there productions

similarly,
use the procedure of eliminating the useless symbols and useless productions and then complete grammar is as ~~foll~~ minimized grammar (By removing useless products)

First Example simplifying:-

[q0, Z0, q0] = A

[q0, Z0, q1] = B

[q0, Z1, q0] = C

[q0, Z1, q1] = D

[q1, Z1, q1] = E

[q1, Z0, q1] = F

[q1, Z0, q0] = G

[q1, Z1, q0] = H

$S \to A | B$

$A \to bCA | \underline{bDG}$

$\times B \to bCB | bDF$

$A \to \epsilon$

$C \to bCC | bDH$

$D \to \underline{bCD} | \underline{bDE} | aE$ ✓

✓ $E \to b$

✓ $F \to aE$

$\times F \to aB$

✓ $G \to aA$

$\times H \to aB$

Q) $M = \{ \{q_0, q_1\}, \{0,1\}, \{0,1,z_0\}, \delta, q_0, z_0, \phi \}$

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

$\delta(q_0, 0, z_0) = (q_0, 0z_0)$

$\delta(q_0, 0, 0) = (q_0, 00)$

$\delta(q_0, 1, 0) = (q_0, 10)$

$\delta(q_0, 1, 1) = (q_0, 11)$

$\delta(q_0, 0, 1) = (q_1, \epsilon)$

$\delta(q_1, 0, 1) = (q_1, \epsilon)$

$\delta(q_1, 0, 0) = (q_1, \epsilon)$

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

$S \rightarrow [q_0, z_0, q_0] \mid [q_0, z_0, q_1]$

1) $[q_0, z_0, q_1] \rightarrow \epsilon$

2) $[q_0, z_0, q_0] \rightarrow$ 0 $[q_0, 0, q_0]$ $[q_0, z_0, q_0]$

| 0 $[q_0, 0, q_1]$ $[q_1, z_0, q_0]$

$[q_0, z_0, q_1] \rightarrow$

0 $[q_0, 0, q_0]$ $[q_0, z_0, q_1]$

| 0 $[q_0, 0, q_1]$ $[q_1, z_0, q_1]$

3) $[q_0, 0, q_0] \rightarrow$ 0 $[q_0, 0, q_0]$ $[q_0, q_0]$

| 0 $[q_0, 0, q_1]$ $[q_1, 0, q_0]$

$[q_0, 0, q_1] \rightarrow$ 0 $[q_0, 0, q_0]$ $[q_0, 0, q_1]$

| 0 $[q_0, 0, q_1]$ $[q_1, 0, q_1]$

4) $[q_0, 0, q_0] \rightarrow$ | $[q_0, 1, q_0]$ $[q_0, 0, q_0]$
|| $[q_0, 1, q_1]$ $[q_1, 0, q_0]$

$[q_0, 0, q_1]$ $\rightarrow$ | $[q_0, 1, q_0]$ $[q_0, 0, q_1]$
|| $[q_0, 1, q_1]$ $[q_1, 0, q_1]$

5) $[q_0, 1, q_0] \rightarrow$ | $[q_0, 1, q_0]$ $[q_0, 1, q_0]$
|| $[q_0, 1, q_1]$ $[q_1, 1, q_0]$

$[q_0, 1, q_1]$ $\rightarrow$ | $[q_0, 1, q_0]$ $[q_0, 1, q_1]$
|| $[q_0, 1, q_1]$ $[q_1, 1, q_1]$

6) $[q_0, 1, q_1] \rightarrow \epsilon$

7) $[q_1, 1, q_1] \rightarrow \epsilon$

8) $[q_1, 0, q_1] \rightarrow \epsilon$

9) $[q_1, z_0, q_1] \rightarrow \epsilon$

$[q_0, z_0, q_0]$ — A
$[q_0, z_0, q_1]$ — B
$[q_0, 0, q_0]$ — C
$[q_0, 0, q_1]$ — D
$[q_0, 1, q_0]$ — E
$[q_0, 1, q_1]$ — F

$[q_1, 1, q_1]$ — G
$[q_1, 0, q_1]$ — H
$[q_1, z_0, q_1]$ — I
$[q_1, z_0, q_0]$ — J
~~$[q_0, z_0, q_1]$ — K~~
$[q_1, 0, q_0]$ — K
$[q_1, 1, q_0]$ — L

$S \to A | B$

$B \to \epsilon$

$A \to 0CA \mid \underline{0DJ}$

$B \to 0CB \mid \underline{0DI}$

$C \to 1EC \mid 1FK$

$D \to 1ED \mid \underline{1FG}$

$E \to 1EE \mid 1FL$

$F \to 1EF \mid \underline{1FG}$

$G \to$

$F \to 0$

$G \to 0$

$H \to 0$

$I \to \epsilon$

Q) $P = \{\{q\}, \{i, e\}, \{x, z\}, \delta, q, z, \phi\}$

$\delta(q, i, z) = \{(q, xz)\}$

$\delta(q, \epsilon, x) = \{(q, \epsilon)\}$

$\delta(q, \epsilon, z\} = \{(q, \epsilon)\}$

$S \to [q, z, q]$

$[q, z, q] - i[q, x, q], [q, z, q]$

$[q, x, q] - \epsilon$

$[q_1 Z_1 q] \to \epsilon$

✓ $S \to A$

✓ $A \to i\,B\,A$

✓ $B \to \epsilon$

✓ $A \to \epsilon$

(All are usefull symbols)

Unit-5

Turing Machine

→ The Machine is able to move R/W read/write head left & right over the tape as it performs its compositions.

→ It can read & write symbols as it places

→ This considerations lead turing to the following formal definition

Definition:-

A turing machine is 7 tuple machine

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

Q = Finite set of states

Σ = input alphabet

Γ = tape alphabet which always includes blank

$\delta = Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

q_0 = initial state

B = Special symbol indicates of a blank cell

F = set of final states

→ The Machine simply moves right along the tape until it hits the Blank and then machine turn ~~right~~ left and it halts.

→ At each step it just ~~right~~ writes back current symbol, remains on q_0 and moves right by one cell.

→ The transition can be defined as

$$\delta(q_0, 0) = (q_0, 0, R)$$
$$\delta(q_0, 1) = (q_0, 1, R)$$

→ once the machine hits a blank it moves one cell to the left and stops

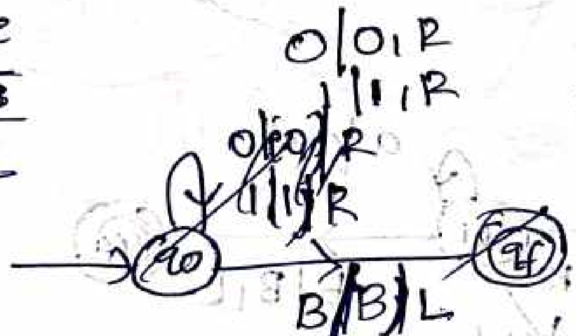
$$\delta(q_0, B) = (q_f, B, L)$$

Q) Design a turing machine to accept the string belongs to language $(0+1)^*$

$\{\epsilon, 0, 1, 00, 01, 10, 11, 101, \ldots\}$

input tape

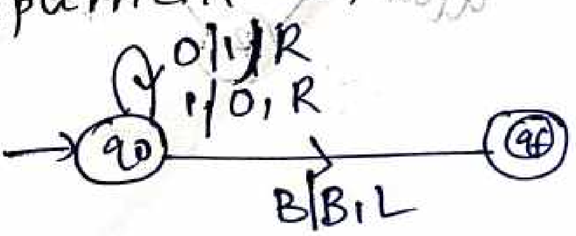
0	1	0	1	B



Q) Finding one's complement of a Binary Number.

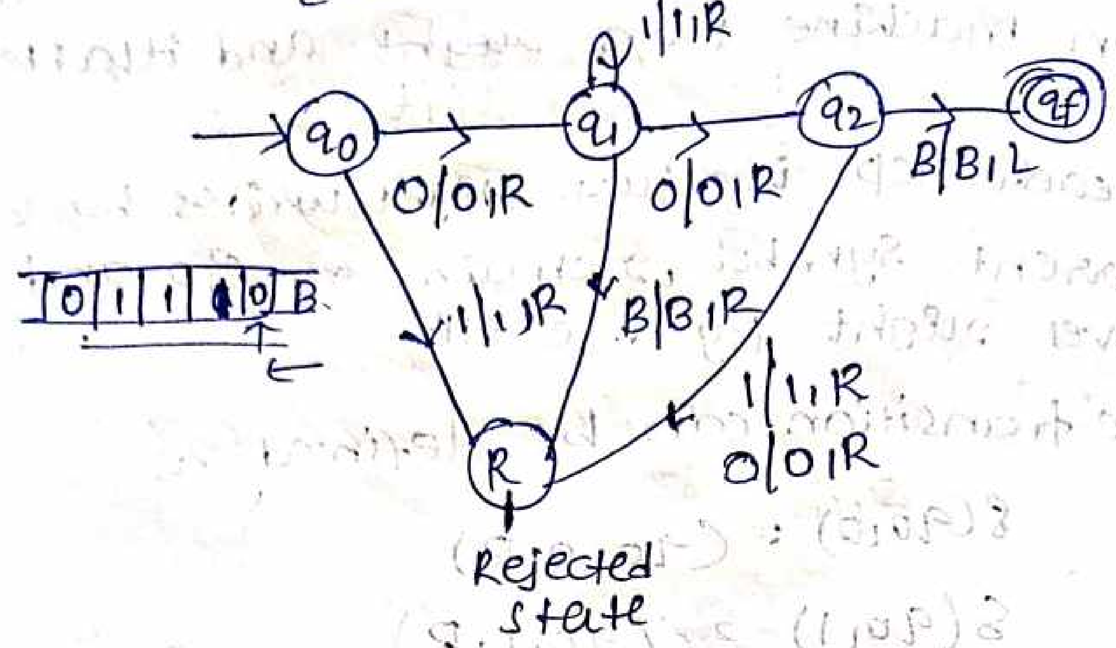
0101
1010

0	1	0	1	B



Q) 01^*0

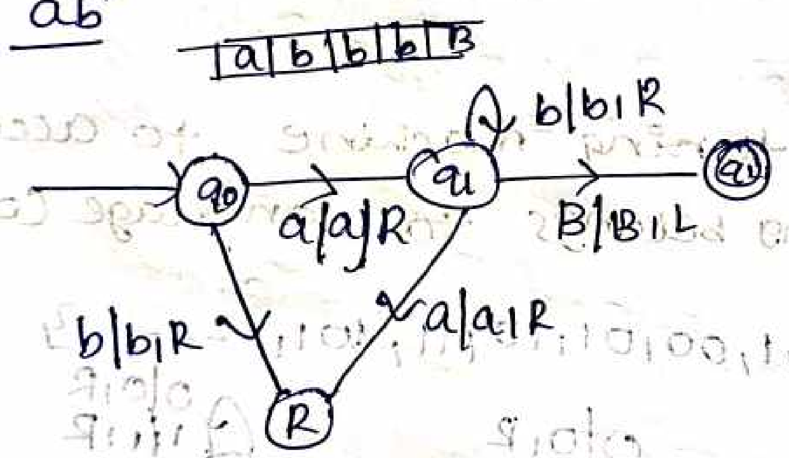
$\{00, 010, 0110, \ldots\}$



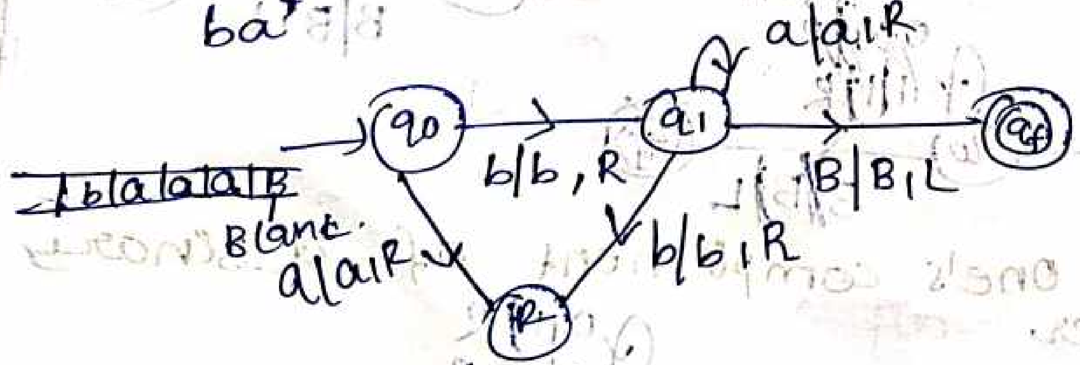
Q) AB^* (ఠ) BA^*

ab^* (ఠ) ba^*

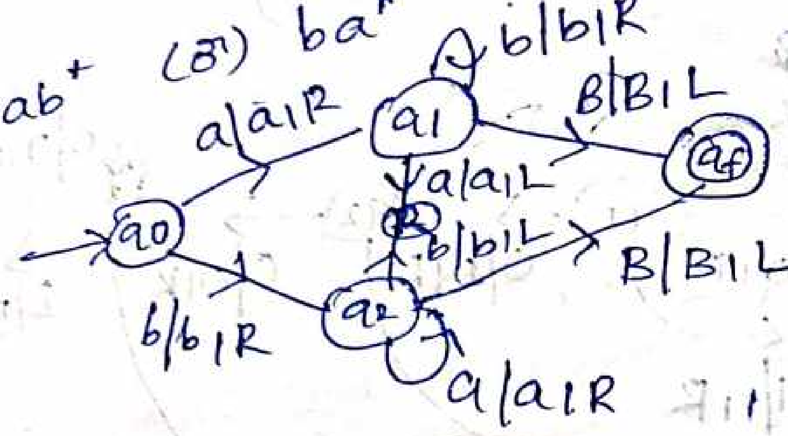
$\underline{ab^*}$



ba^*

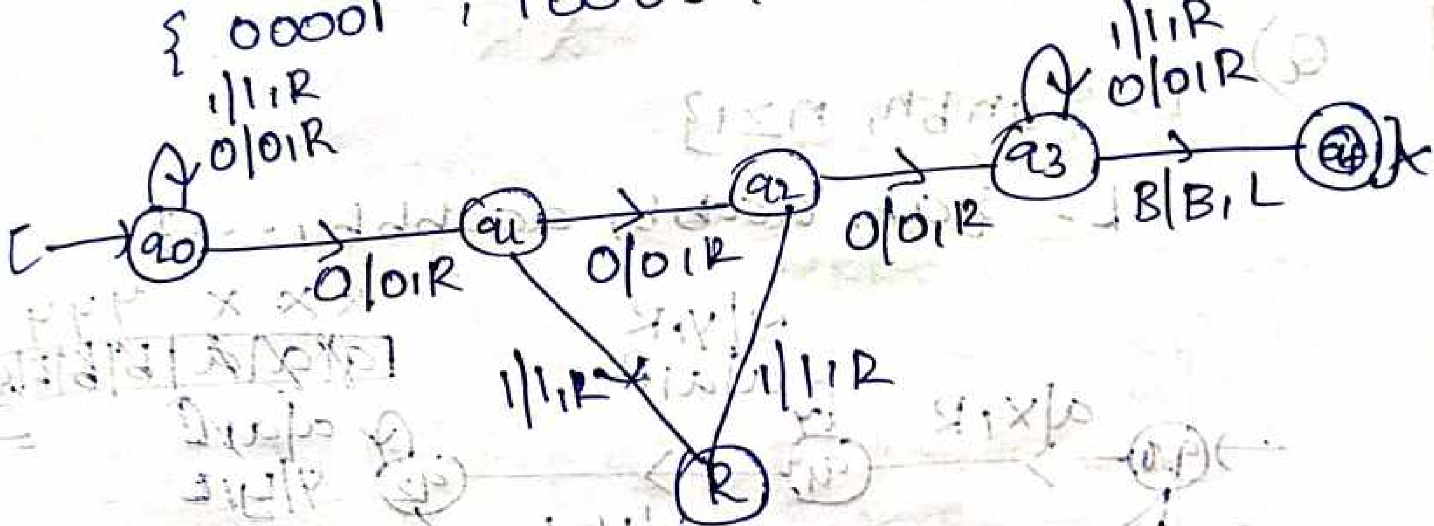


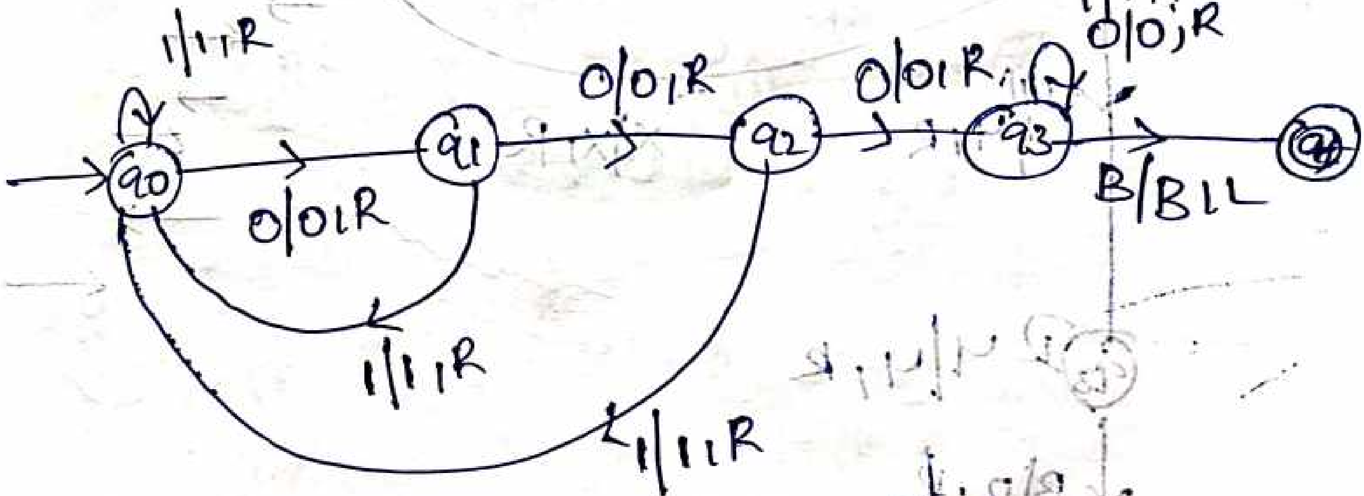
ab^+ (or) ba^*



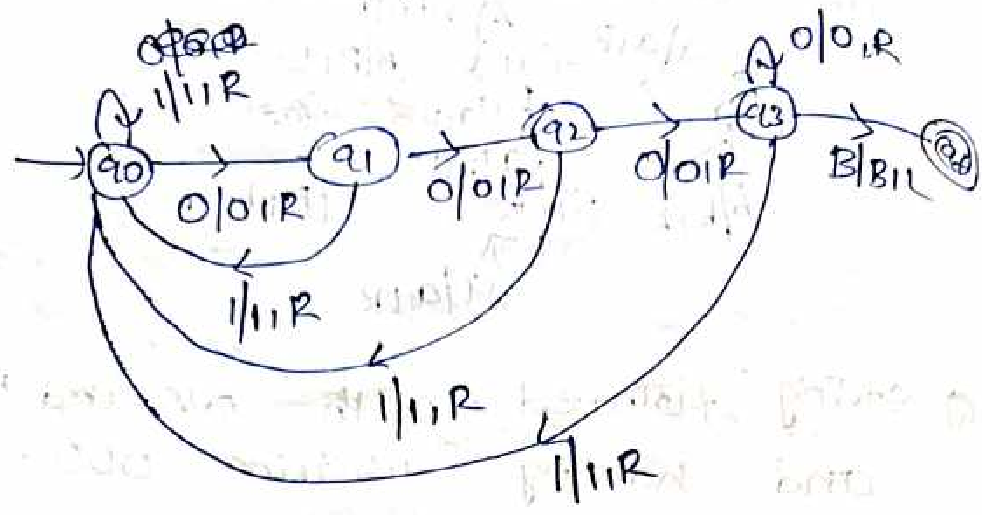
Q string formed with 0's and 1's and having substring 000.

{00001, 10000, 00000111 ...}.



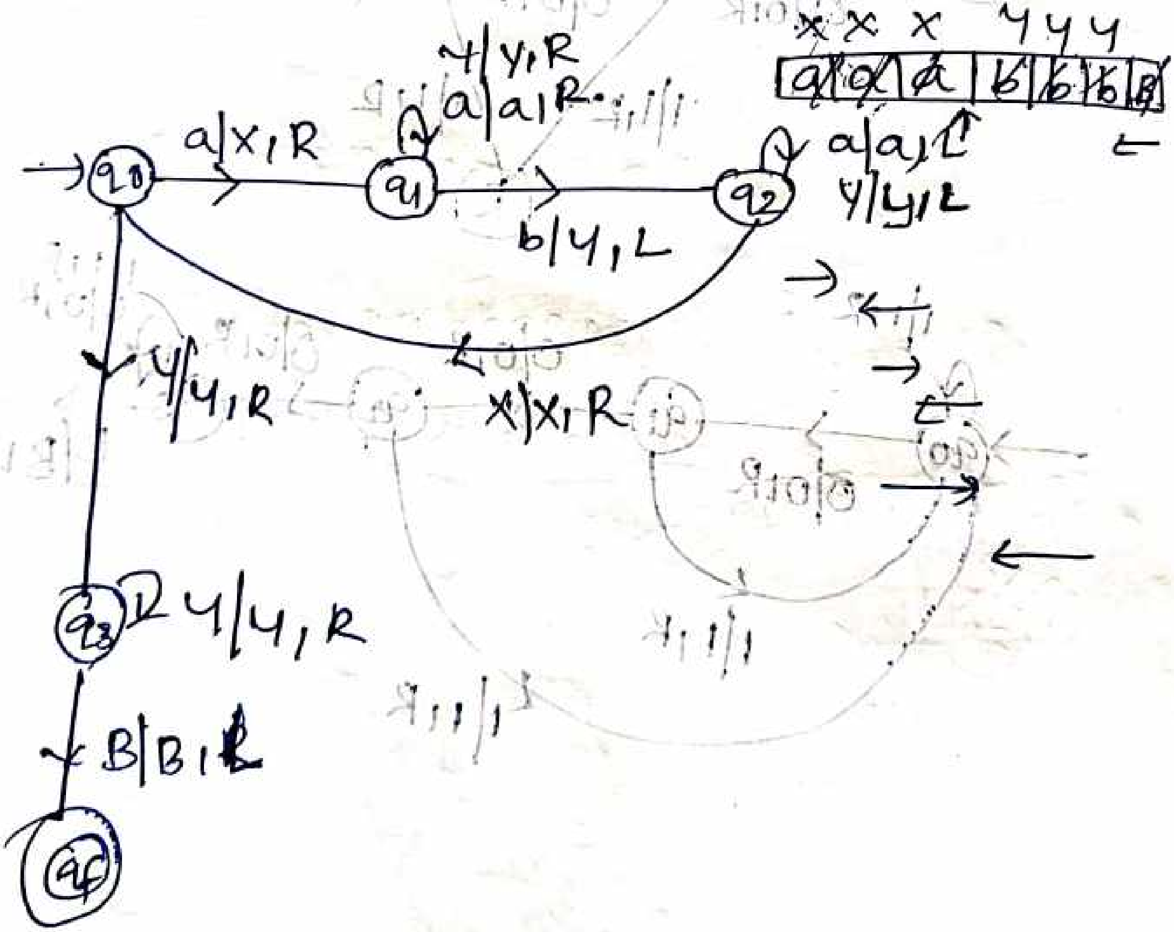


Q) Ends with 3 zeroes (000)



Q) $L = \{a^n b^n, n \geq 1\}$

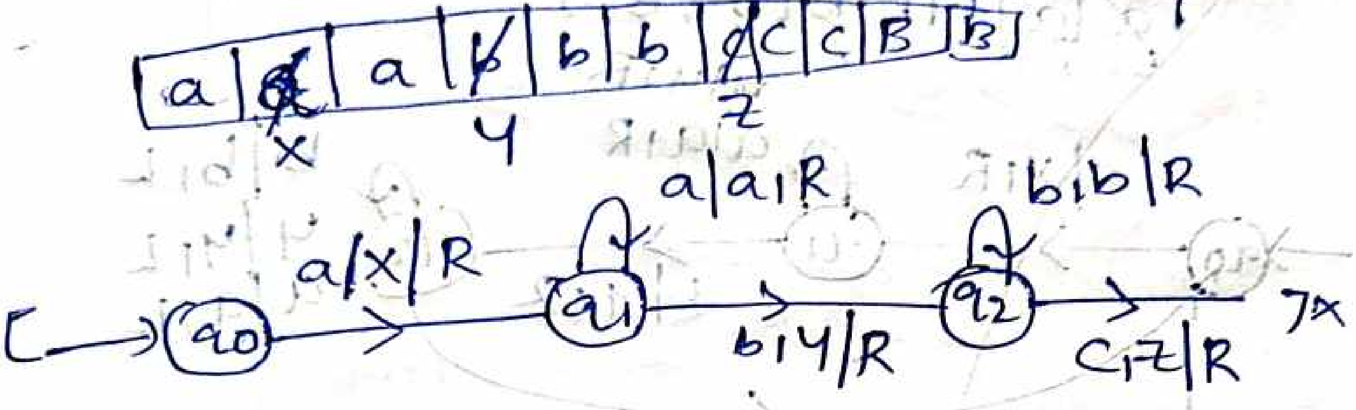
$L = \{ab, aabb, aaabbb, \ldots\}$



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

~~{(a...aabb~~

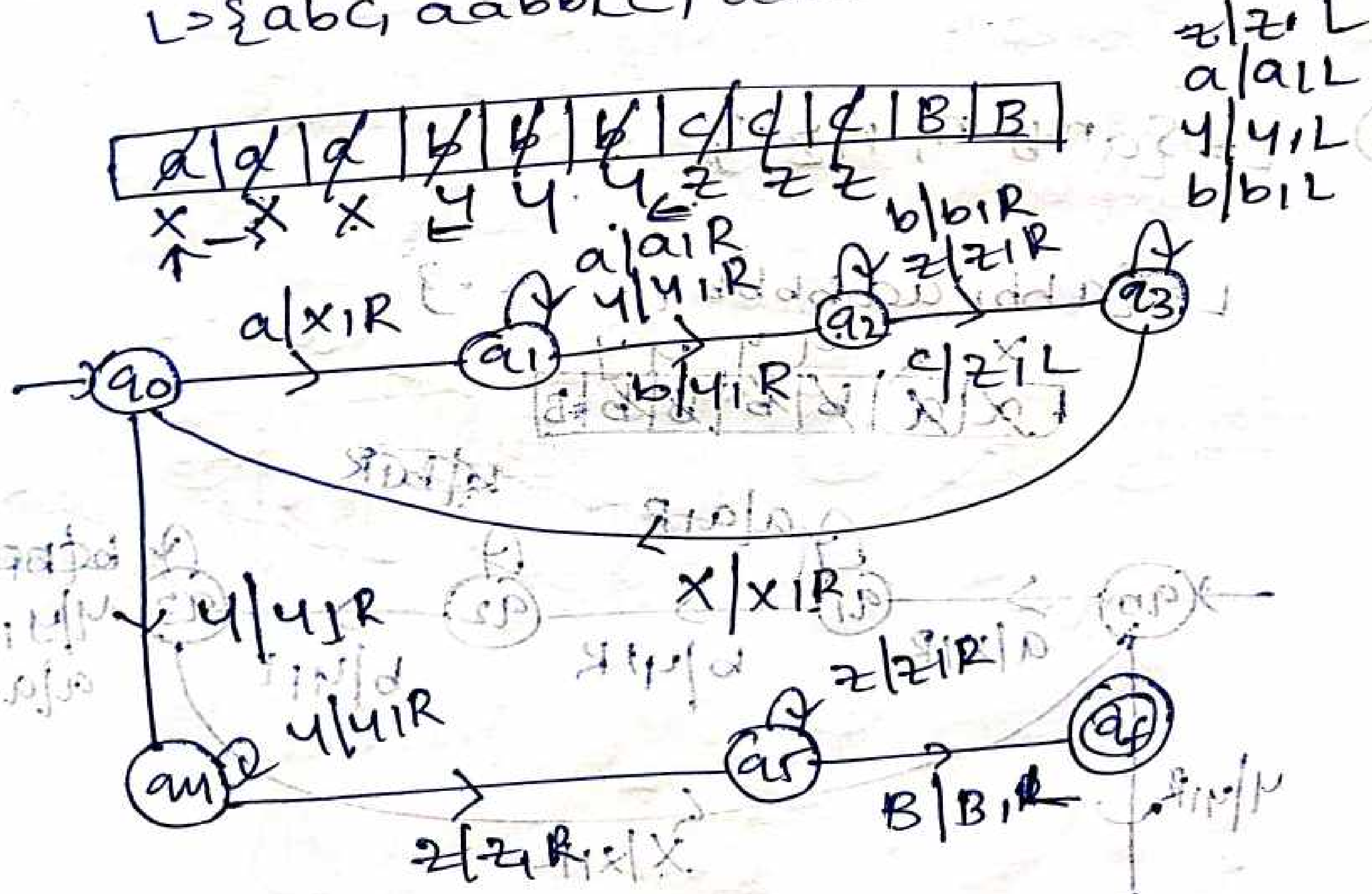
$L=\{abc, aabbcc, aaabbbccc, \ldots\}$



Q) ~~$L=\{a^n b^n$~~

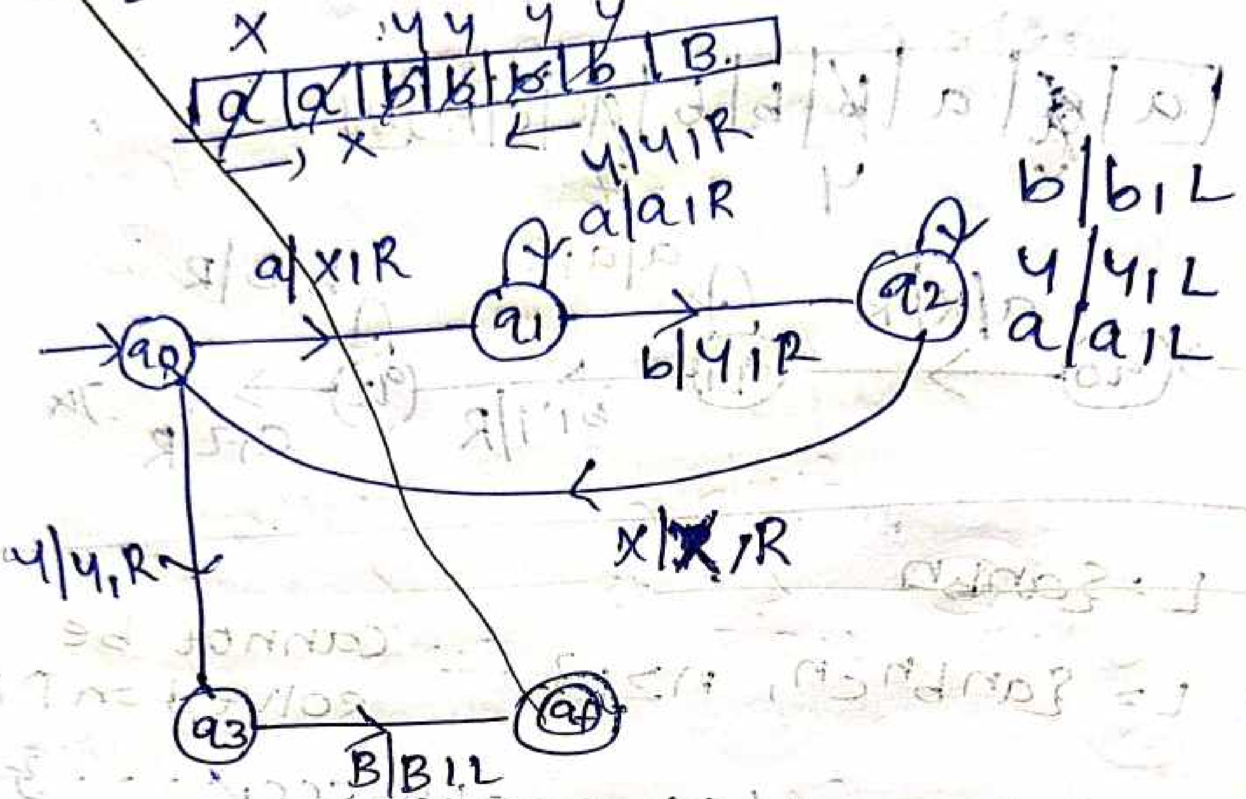
Q) $L=\{a^n b^n c^n, n\geq 1\}$ — cannot be solved on PDA

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



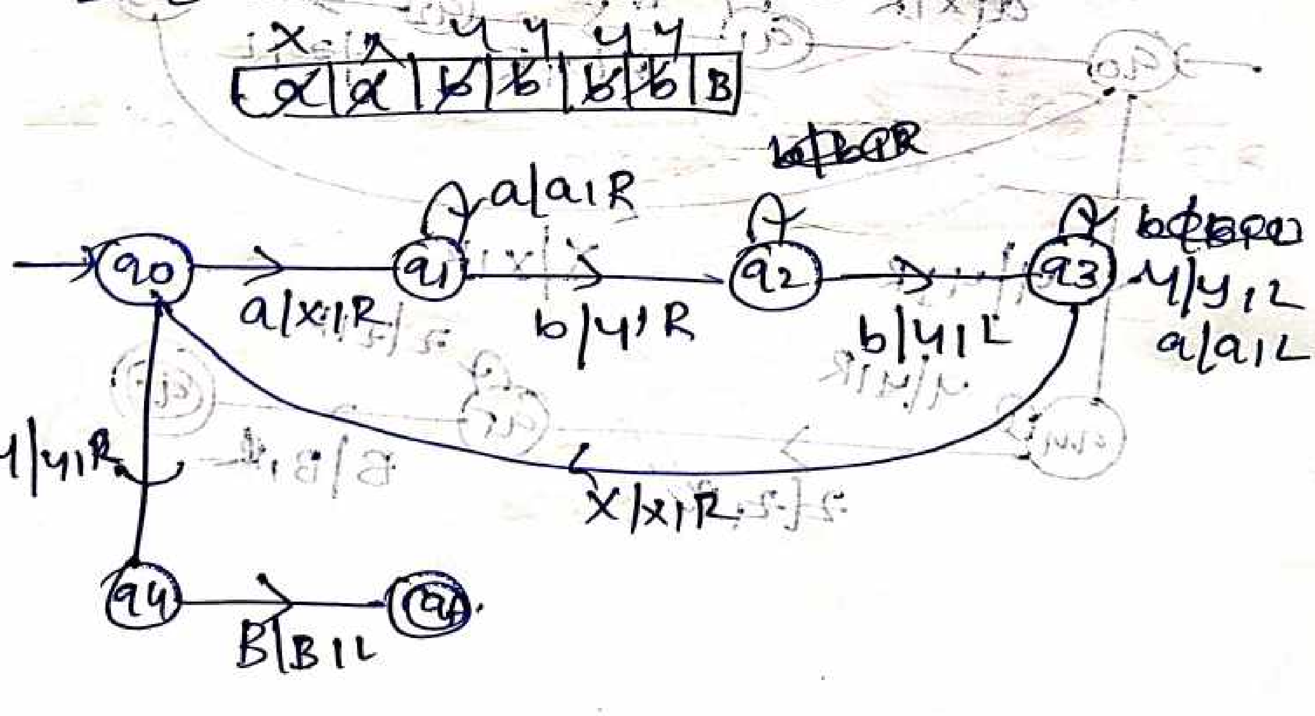
Q) $L = \{a^n b^{2n},\ n \geq 1\}$

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



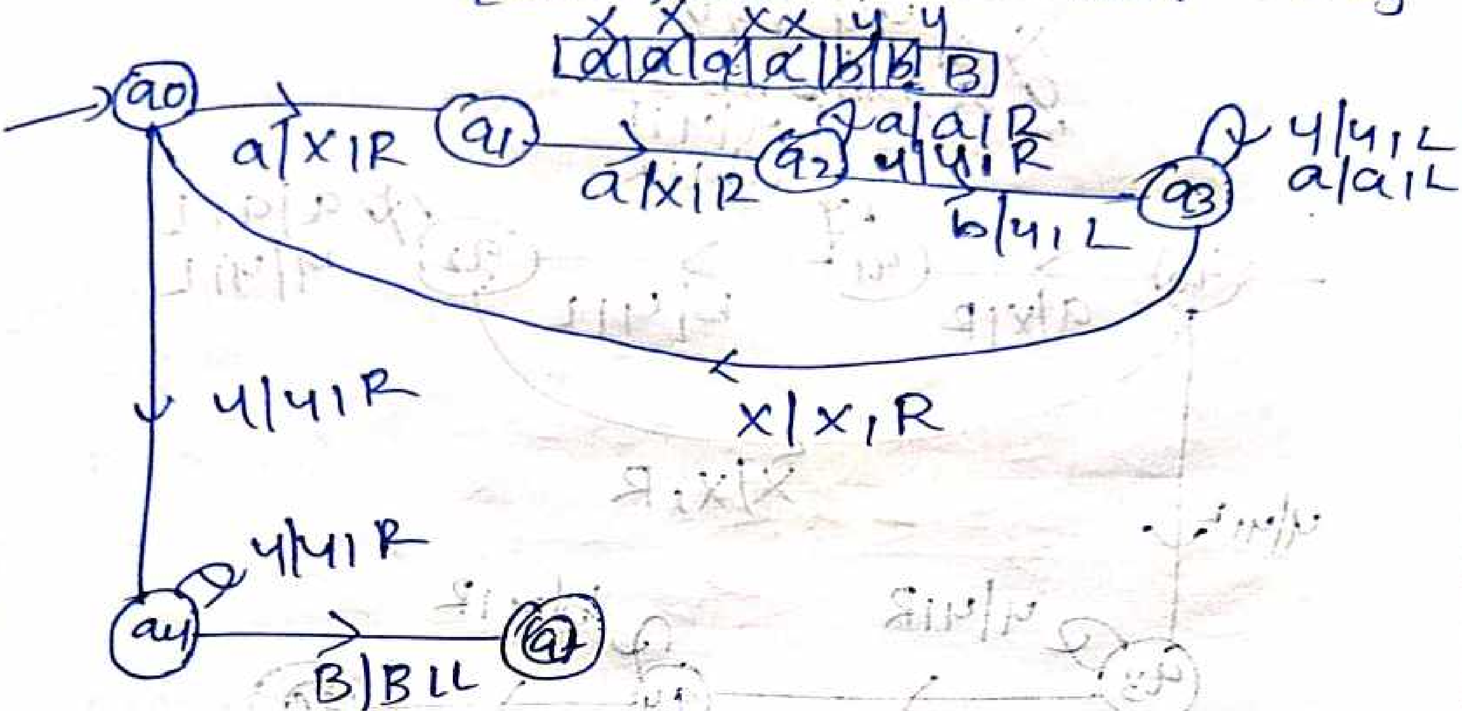
Q) $L = \{a^n b^{2n},\ n \geq 1\}$

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



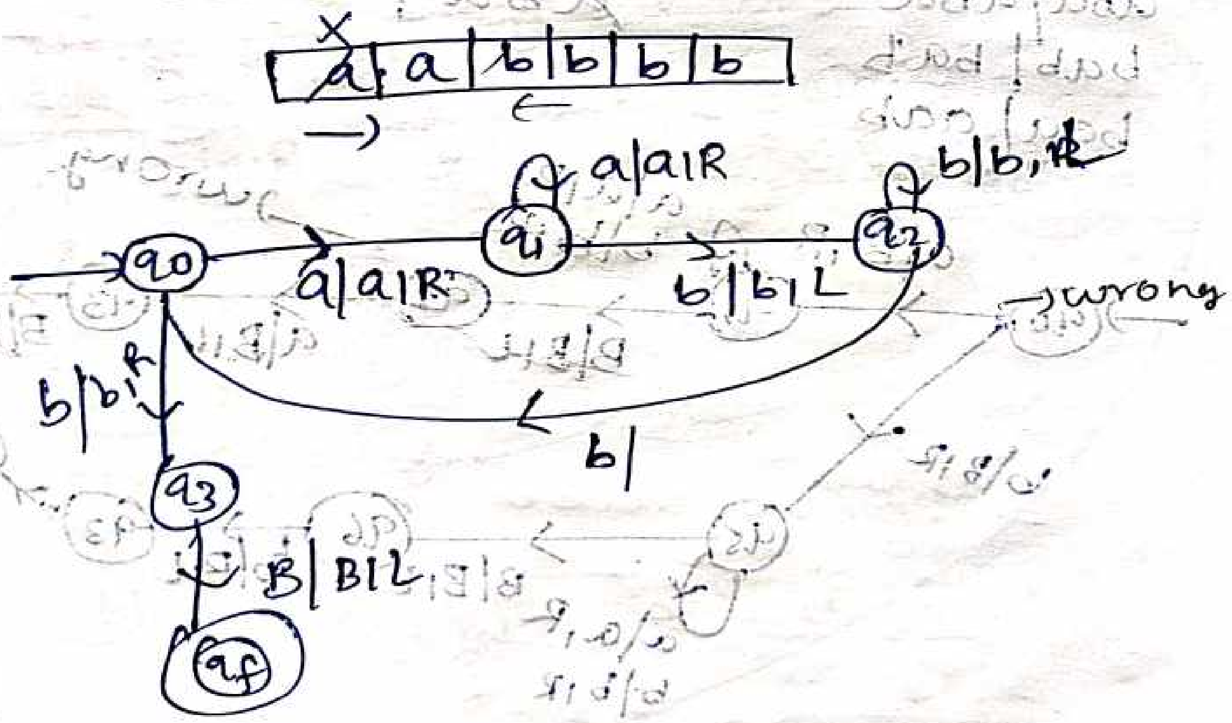
Q) $L = \{a^{2n} b^n, n \geq 1\}$.

$\{aab, \cancel{aabb}\ aaaabb, \ldots\}$



Q) $L = \{a^i b^j, i < j\}$.

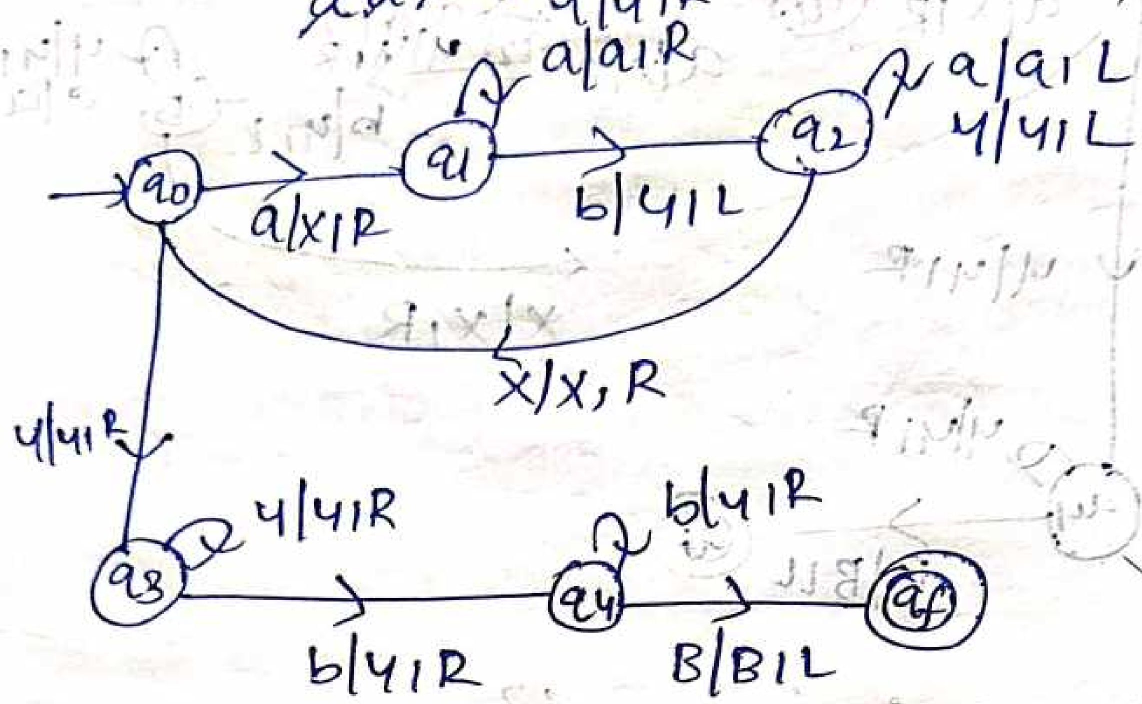
$\{b, abb, aabbb, aaabbbb, \ldots\}$



Q) $L=\{a^i b^j, n(\neq 0)\}$ $i=1$

$i<j$

x x y y y y y y B
a a b b b b b b

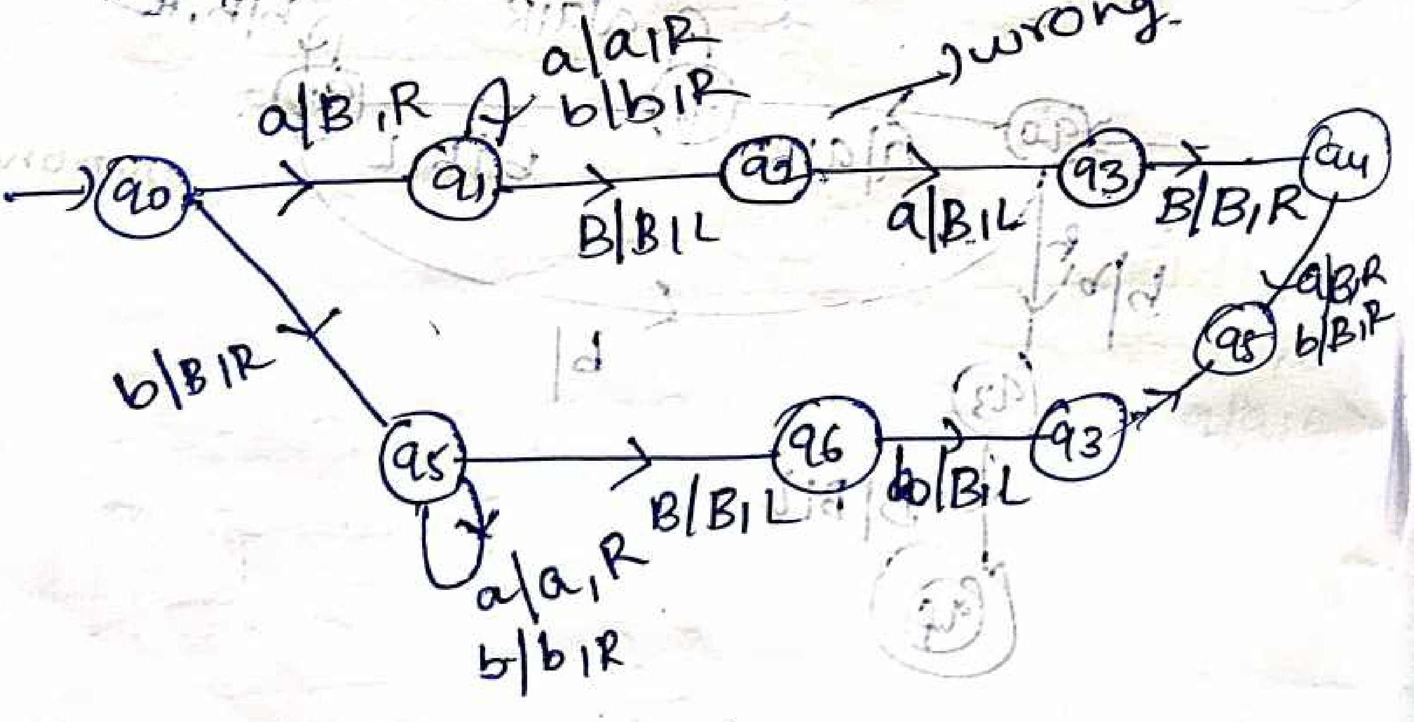


Imp

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

aba|aba
bab|bab
baa|aab

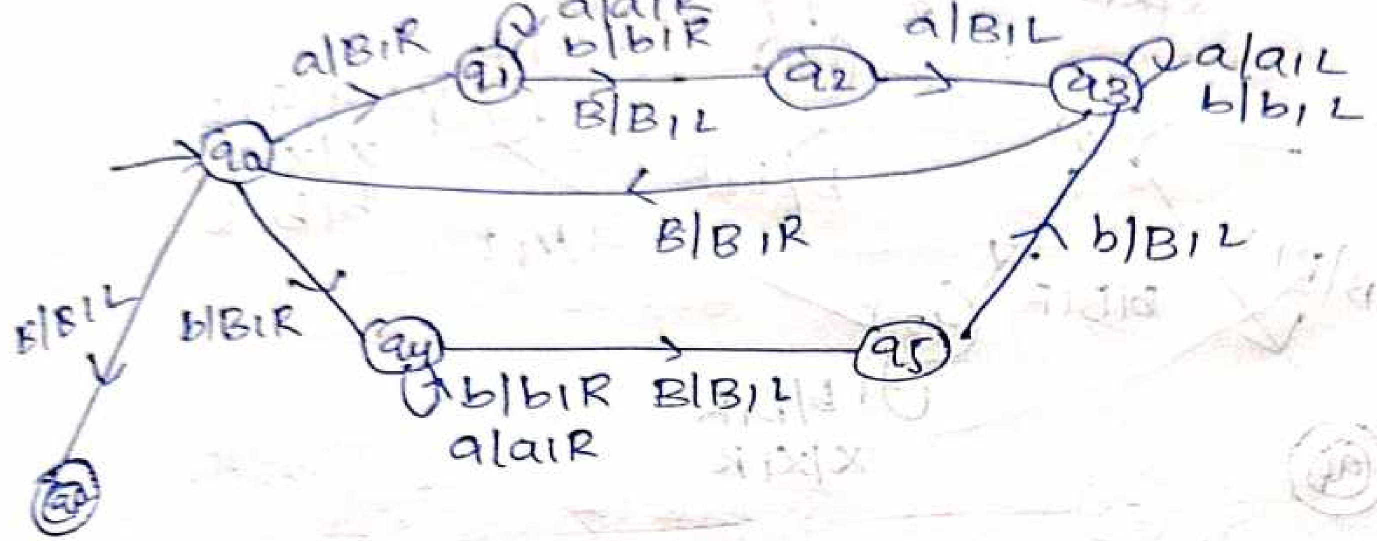
aba|aba



IMP

Q) $L = \{ww^R,\ w \in (a+b)^*\}$ – even palindrome

aaabb|bbaaa

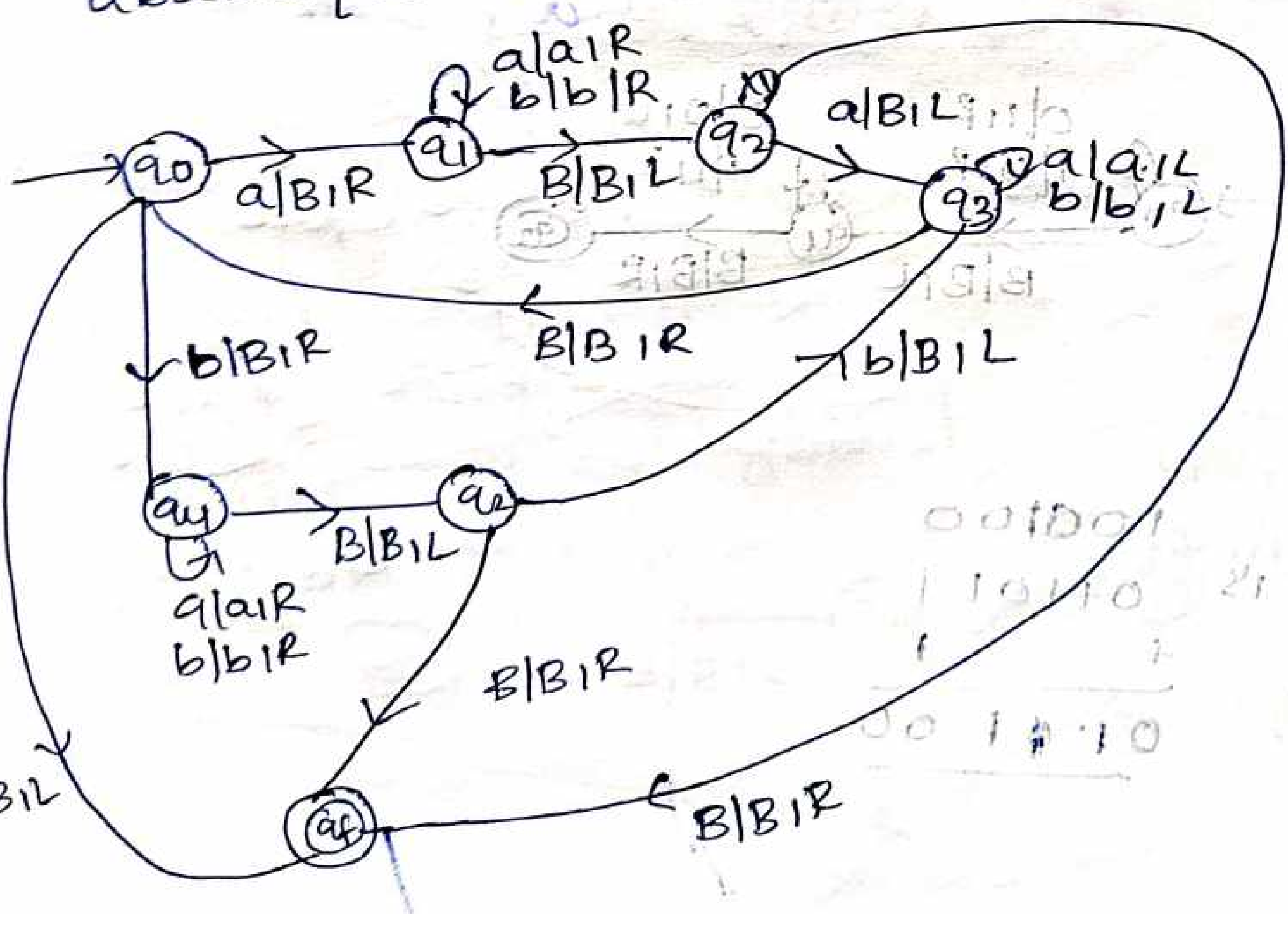


IMP(***)

palindrome

$L = \{aba, abba, ababa, \ldots\}$

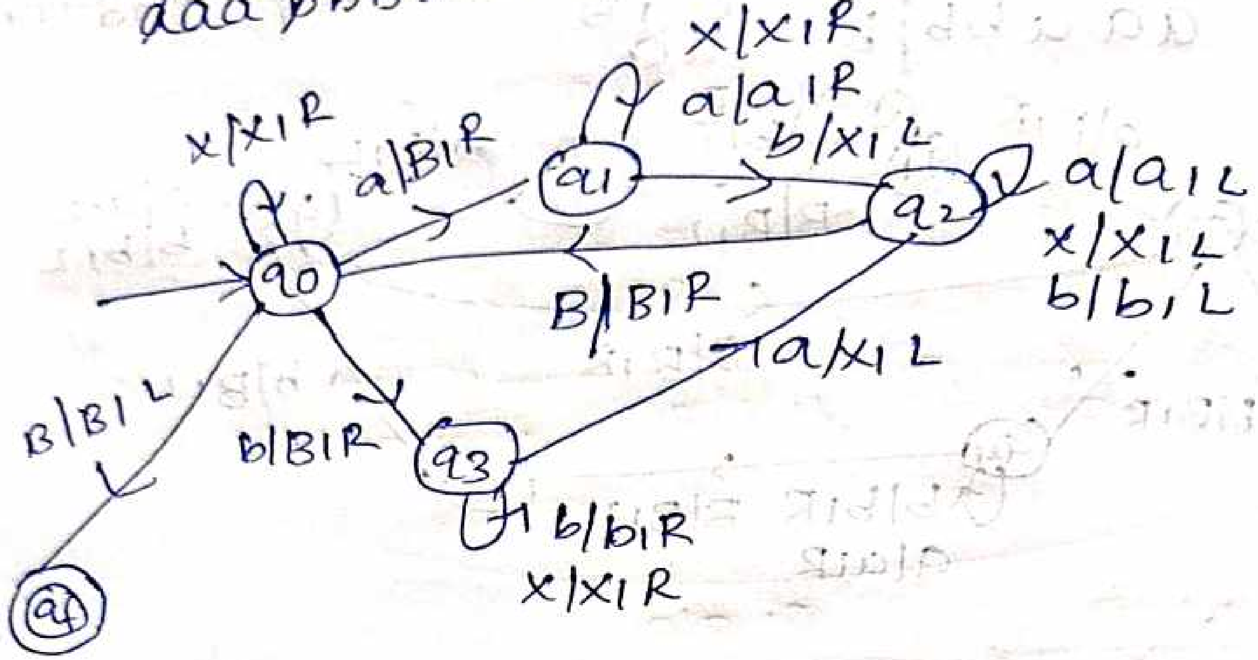
ababa|ababa



Q) $n_a(w) = n_b(w)$

B X

daa ~~b~~bbaabb

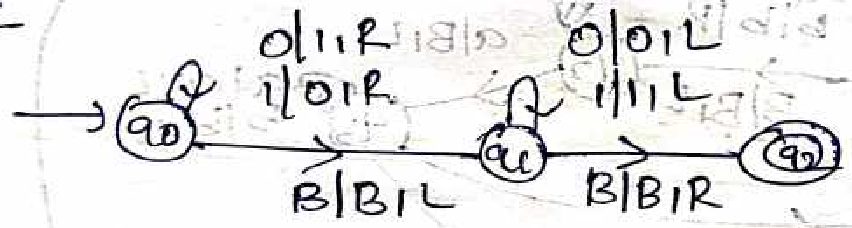


Q) Design a turing machine of 1's compliment and 2's compliment

1010

1's = 0101

1's



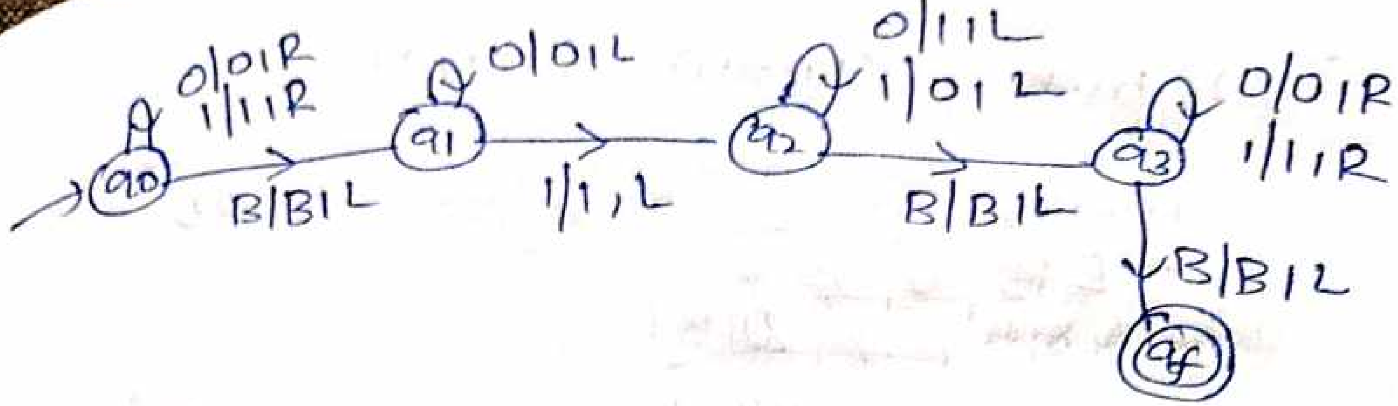
2's

100100

1's 011011

+ 1

01 ~~0~~1 00

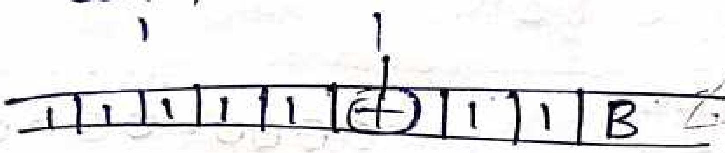


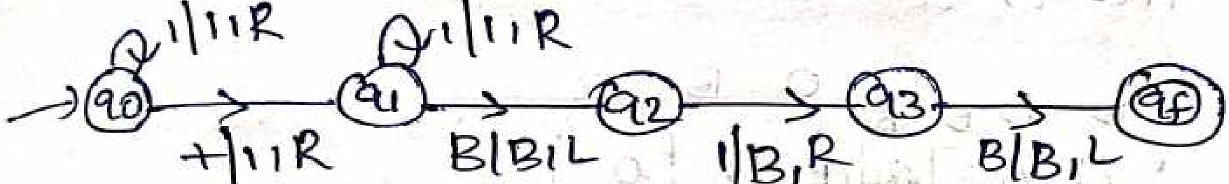
Q) f(m) = m+n, $1^m 0 1^n$ (unary operations)

5 + 2 = 7

11111 (+) 11 = 1111111

replace with 1

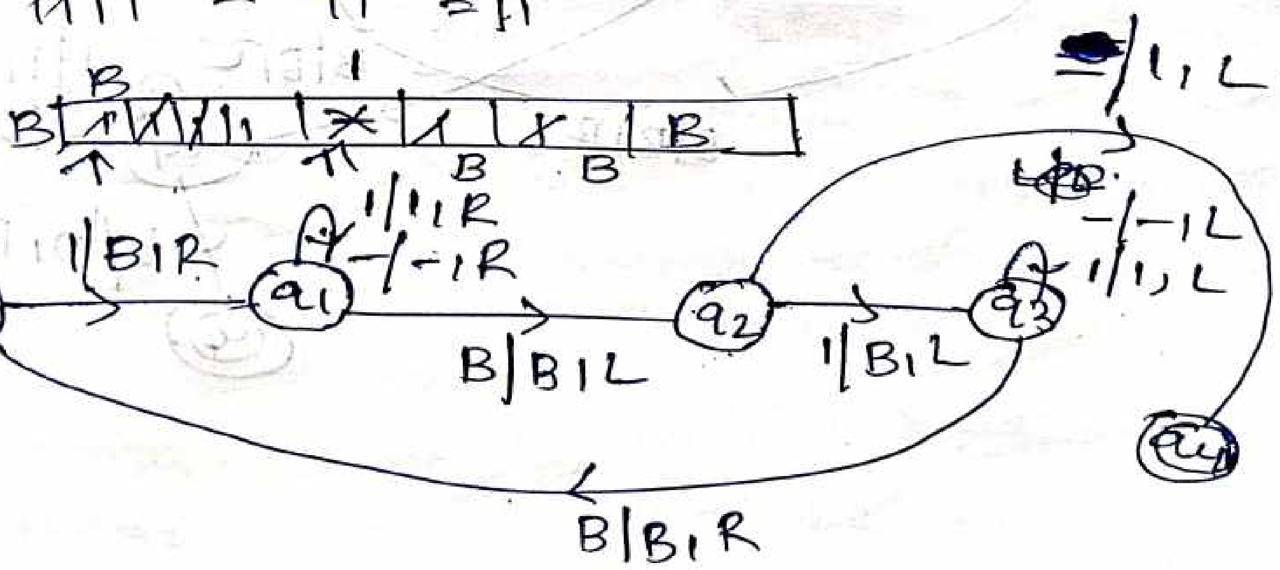




Q) f(m) = m-n (m>n) $0^m 1 0^n$

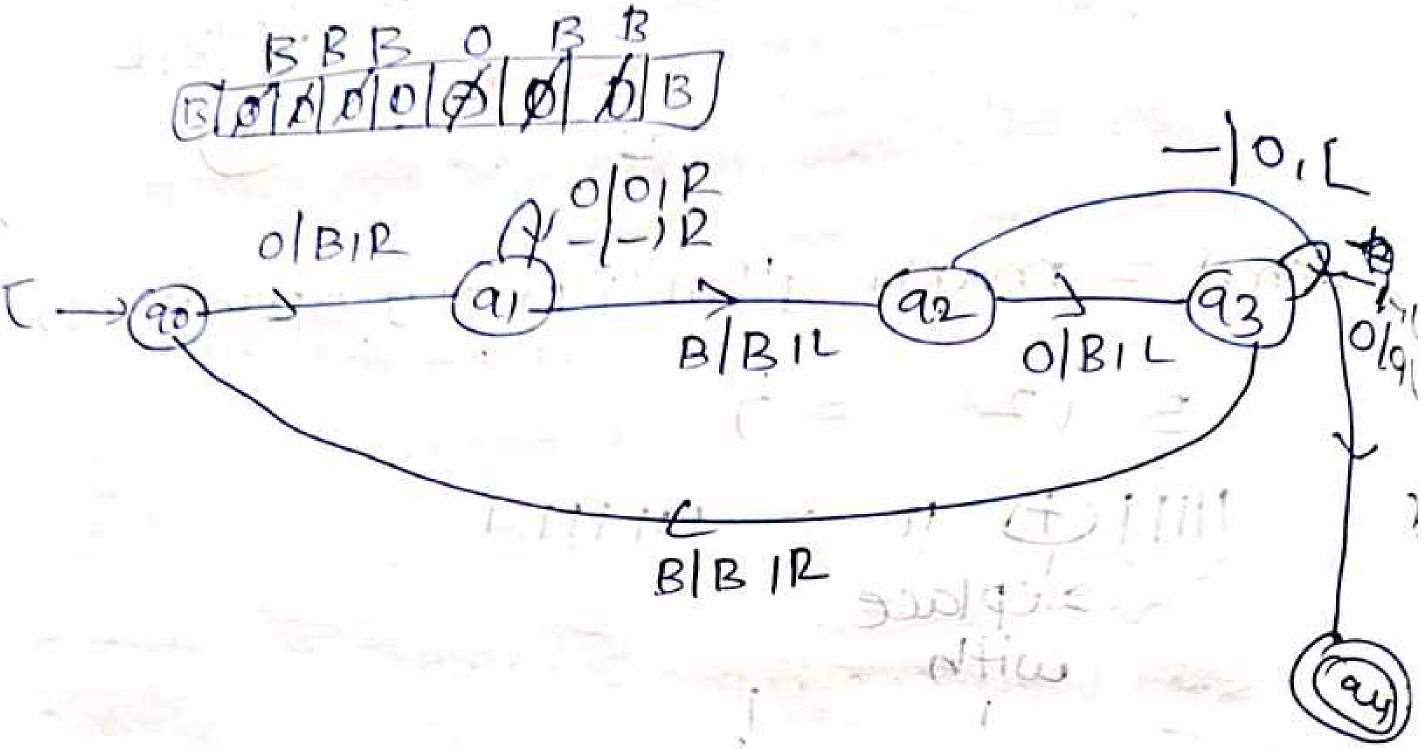
m=4 n=2

1111 - 11 = 11

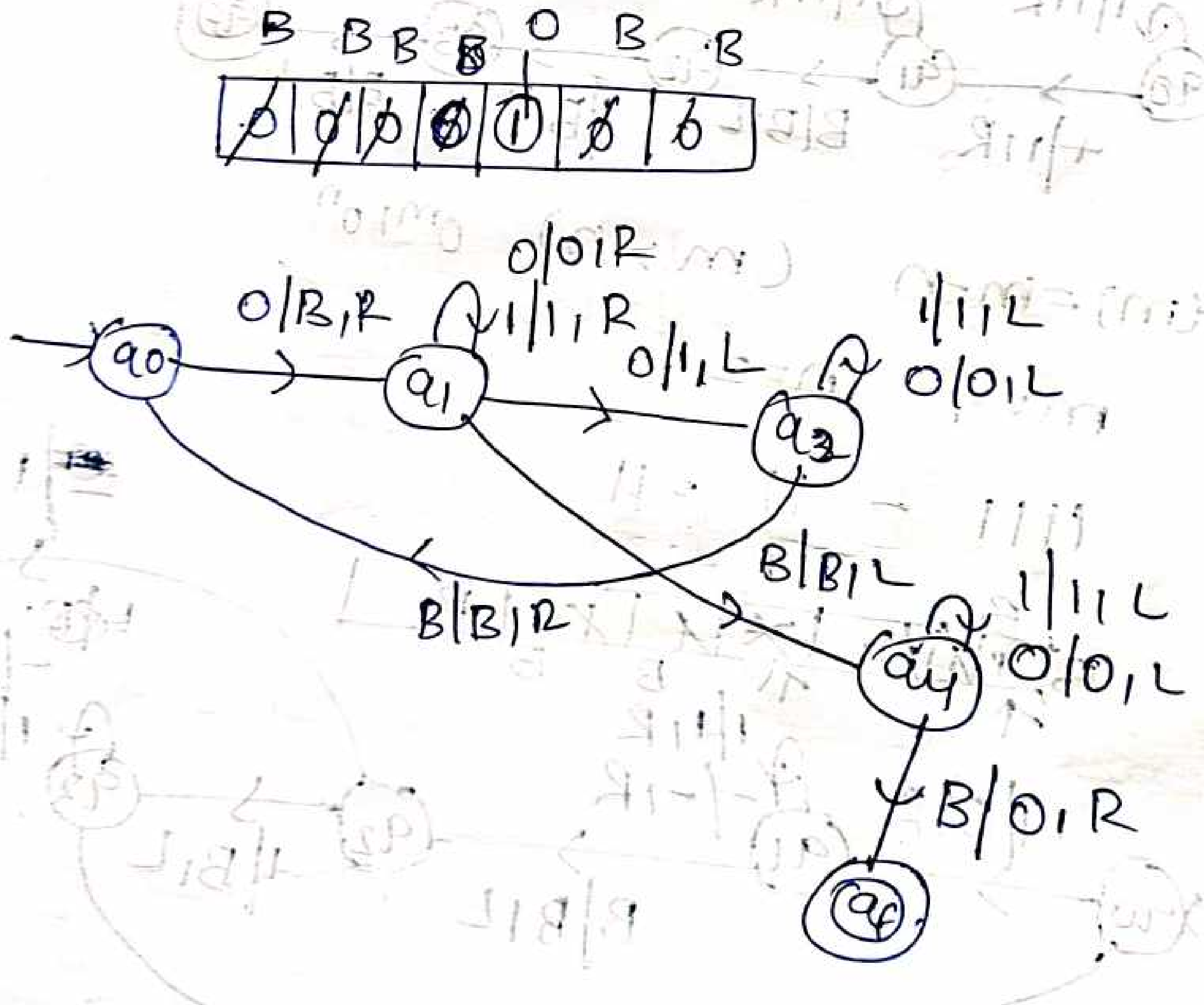


Q) f(m) = m-n (m>n)

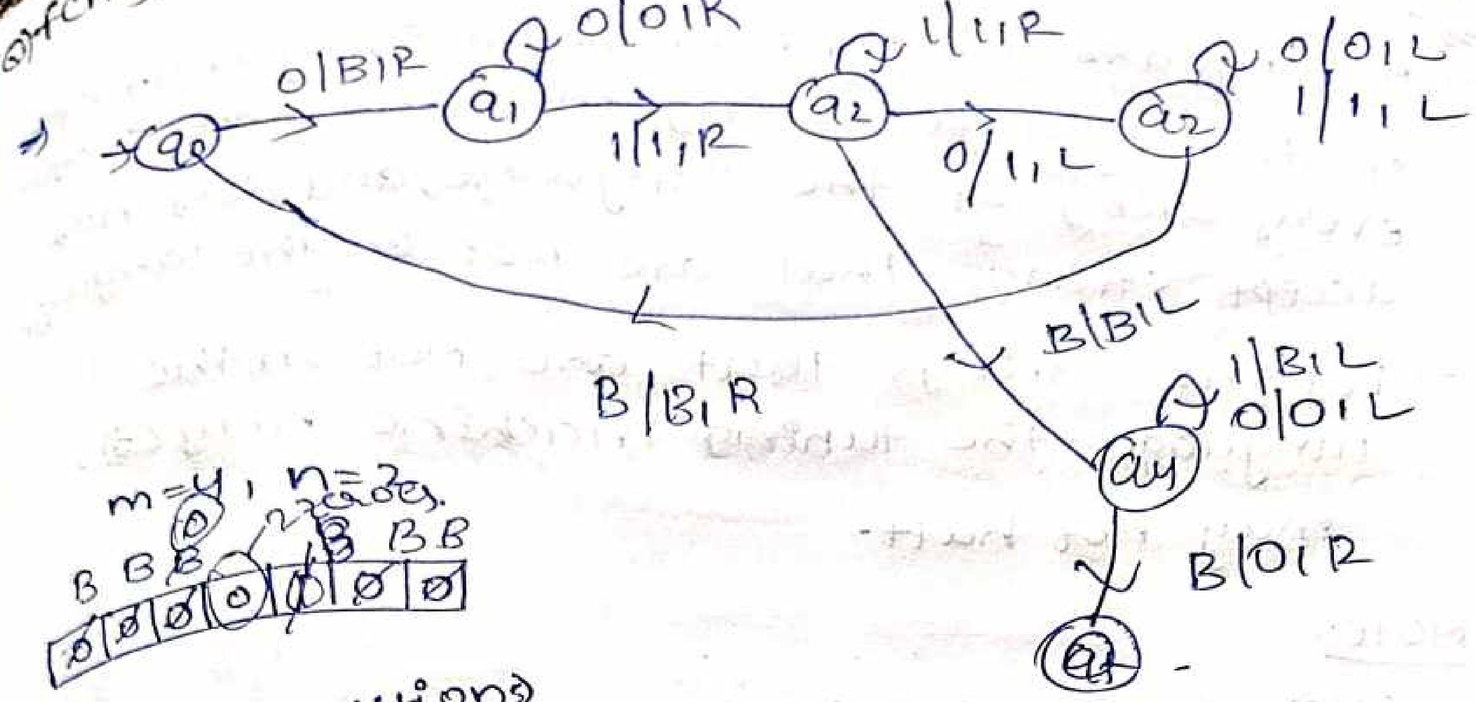
m=4, n=2 = 0000 - 00 = 00



m=4, n=2 = 0000-00=00



(Q) $f(m) = m-n$ $(m>n)$ $0^m 1 0^n$



(imp) (theory questions)

Recursive and Recursively Enumerable languages.

⇒ there are 3 possible outcomes of Executing a turing machine over a given input

⇒ The turing machine may:-

(i) Halt and accept the input

(ii) Halt and reject the input

(iii) Never Halt

⇒ A language is said to be recursive if there exist a turing machine that accepts Every string of that Language and rejects every string cover the same alphabet that is Not in the language).

Note:-

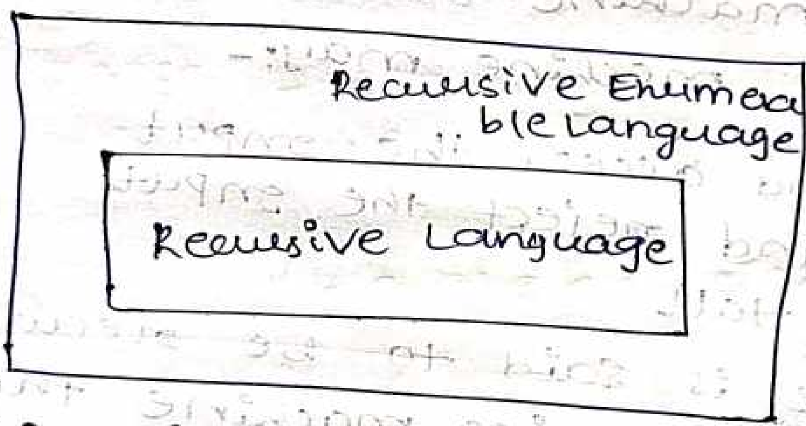
If a language L is recursive, then its compliment $\bar{L}$ must also be recursive

→ A language is said to Recursively enumera[ble] if there exist a turing machine that accep[ts] every string of the Language, and does not accept strings that are not in the languag[e]

→ For the strings that are not in the Language the turing machine may (or) may not halt.

Note:-

Every recursive language is also recursive enumerable language. It is Not obvious whether every recursively enumerable language is also recursive



*** (Imp)

Types of turing Machine:-

→ There are No. of other types of turing machines in addition to one we have seen, such as turing machine with multiple tapes, one tape But with multiple heads, two dimensional, non-deterministic turing machine etc --

→ It turns out that competitionally all these turing machines are equally powerfull. that is what one type can compute any other type can also compute. However the efficiency of competition that is how fast they can compute may vary.

ii) Non-deterministic turing machine:-

→ A non-deterministic turing machine is a machine for which like non-deterministic finite Automata, at any current state and for the tape symbol it is reading, there may be different possible actions to be performed.

→ Here, An action means a combination of writing a symbol on the tape, moving the tape head and going to next state

example:-

$L = \{ww \mid w \in (a+b)^*\}$

→ Given a string x, a non-deterministic turing machine that accepts the language L would first guess the midpoint of x, which is ~~where~~ the place where the second half of x starts. It must find the mid point by ~~playing~~ pairing of x two ~~be~~ ends symbols from of x

→ Formally a Non-deterministic turing machine transaction function takes

$Q \times T \times \{L, R\}$

$Q \times T \longrightarrow 2$

(ii) Turing machine with 2 ...

→ It is a kind of turing machine that is one finite control, one read-write head one 2D tape.

→ these cells on the tape is 2D, that is the tape has the top end and the left end. But extends indefinitely to the right and down.

→ It is divided into rows of small ~~space~~ squares. For any TM of this space type there is an equivalent TM with one D tape that is equally powerfull.

→ To simulate a new 2D tape with 1D tape first we map the squares of 2D tape to those of 1D tape diagonally as shown on the following table

2-D Tape:-

v & h are end points

v	v	v	v	v	v
h	1	2	6	7	5
h	3	5	8	4	
h	4	9	13		
h	10	12	15		
h	14				

1-D Tape:-

v	1	v	2	3	h	4	5	6	v	7	8	9	10	h	...

Transition function:-

$$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, T, B\}$$

L → Left, R → Right, T → Top, B → Bottom

→ Some turing machine with one-D Tape can simulate every move of turing machine with a 2-D tape. Hence they are atleast as powerfull as turing machine with 2-D Tape. Since, TM with 2D Tape obviously can Simulate TM with 1-D Tape, It can be said that they are equally powerfull

(iii) TM with multiple tapes:-