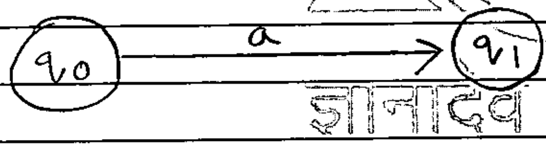
Q.1)

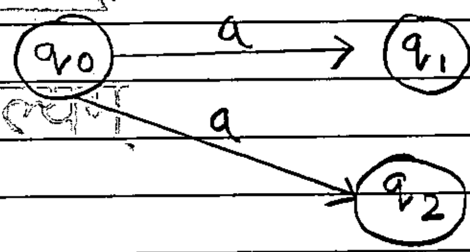
a)

	Deterministic Finite automata	Non-deterministic Finite automata
i)	DFA is a finite automata which have only one transition for each input	NFA is an automata which can have multiple transition for particular Input
ii)	DFA designing is difficult	NFA designing is not that dificult
iii)	All DFAs are NFAs	All NFAs are not DFA

iv) for ex:



for ex:



Q.1)
b)

	0	output	1	output
→q_1	q_1	1	q_2	0
q_2	q_4	1	q_4	1
q_3	q_2	0	q_3	1
q_4	q_3	0	q_2	1

In the moore machine, the output are inside the transition states.
From the given table we can observed that the q_1 and q_4 have only "1" as output and for q_2 and q_3 have '0' and "1" as output.

therefore we can write q_2 and q_3 in the following manner:

q_2 can be split into two parts

q_{21} which have ouput of "1"

q_{20} which have output of "0"

Similarly

q_3 can be split into two parts

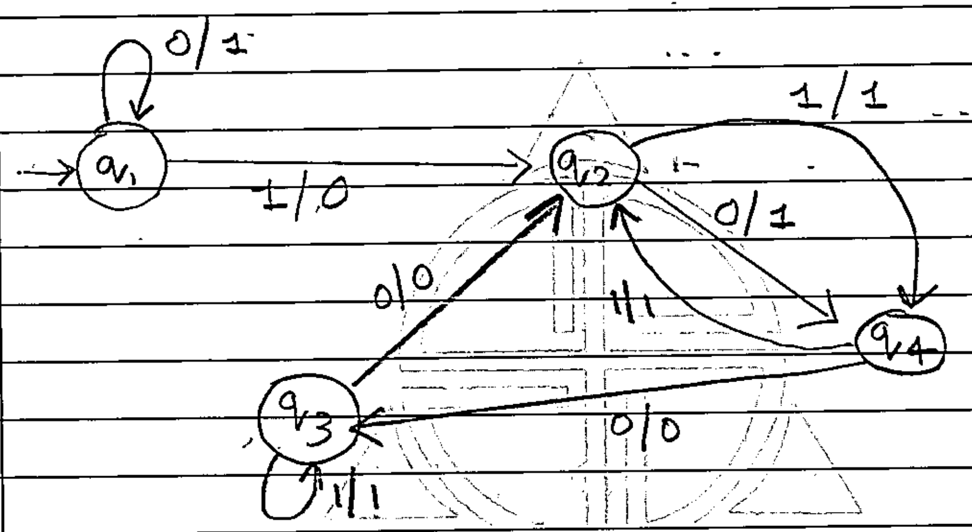
q_{31} which have output "1"

q_{30} which have output "0"

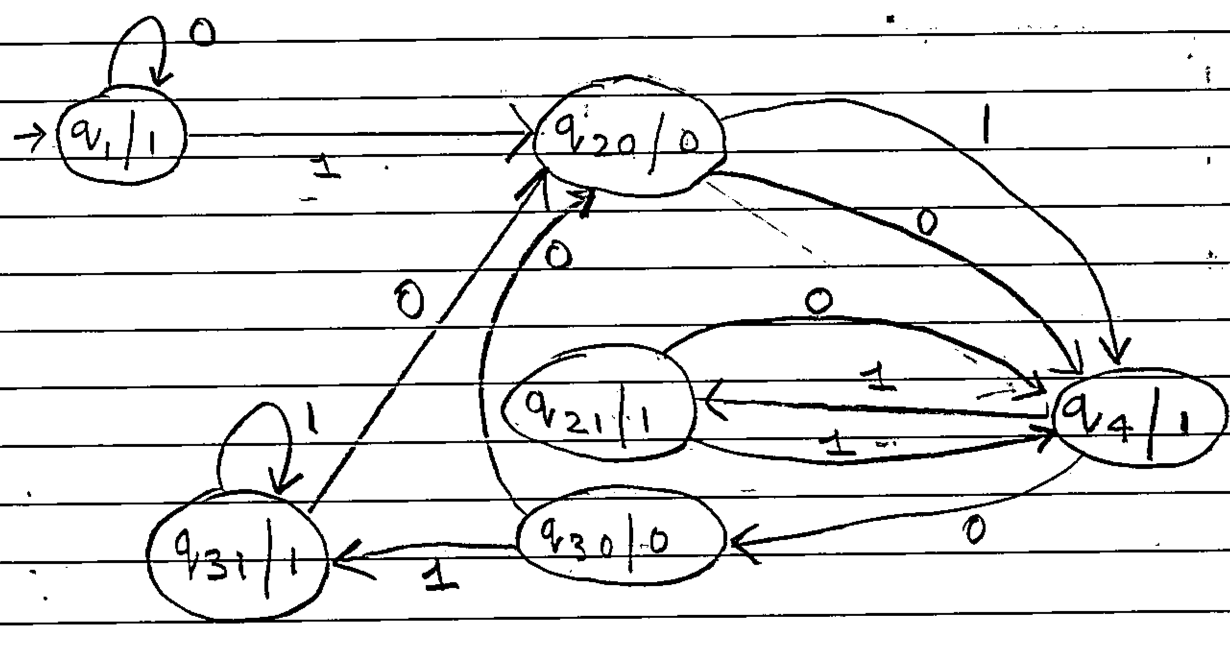
according to question, we have to construct moore machine

∴ Melay to Moore.

Let M be the state diagram for given melay machine.



Let M' be the moore machine:



The transition table for moore machine will be:

	0	1	output
$\rightarrow q_0$	q_1	q_{20}	1
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{31}	q_{20}	q_{31}	1
q_{30}	q_{20}	q_{31}	0
q_4	q_{30}	q_{21}	1

final answer

Q.1) c)

My Hill Nerode theorem, will helps to Convert the DFA to minimize DFA.

So, first we have to make box, and by using box filling method, we have to construct a minimized DFA.

listing out the final and non-final states:

Final states: q_2, q_3, q_5, q_6
non-final states: q_1, q_4, q_7, q_8, q_0

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
q_0									
q_1	✓								
q_2	✓	✓							
q_3	✓	✓	✓						
q_4	✓	✓							
q_5	✓	✓	✓	✓	✓				
q_6	✓	✓	✓	✓	✓	✓			
q_7									
q_8									

We will first mark all final states...
Then, we will check one by one ...

$\delta(q_1, 0) = q_2$ The pairs q_2, q_1 is already
$\delta(q_0, 0) = q_1$ marked, so, we have to
~~mark~~ q_1, ~~q_0 also.~~

$\delta(q_4, 0) = q_5$, the pairs q_5, q_1 is already
$\delta(q_0, 0) = q_1$ marked so we have to mark
q_4, q_0 ...

$\delta(q_4, 0) = q_5$, q_5, q_2 is already marked
$\delta(q_1, 0) = q_2$ therefore have to mark q_4, q_1

$\delta(q_4, 0) = q_5$ It is not marked, so we have
$\delta(q_2, 0) = q_7$ to check for 1 input.
$\delta(q_4, 1) = q_6$ It is unmarked pairs q_6, q_8, so
$\delta(q_2, 1) = q_8$ have to ~~mark~~ leave (q_4, q_2).

$\delta(q_4, 0) = q_5$ q_5, q_8 is not marked,
$\delta(q_3, 0) = q_8$ we move for 1.

$\delta(q_4, 1) = q_6$ q_6, q_7 not marked
$\delta(q_3, 1) = q_7$ so have to leave untick.

$\delta(q_7, 0) = q_1$ and $\delta(q_7, 1) = q_7$ so untick
$\delta(q_0, 0) = q_7$ $\delta(q_0, 1) = q_4$

$\delta(q_7, 0) = q_2$ and $\delta(q_7, 1) = q_7$ so untick
$\delta(q_1, 0) = q_7$ $\delta(q_1, 1) = q_3$

$\delta(q_7, 0) = q_7$ doesnot exist
$\delta(q_2, 0) = q_7$
$\delta(q_7, 1) = q_7$ so leave untick.
$\delta(q_2, 1) = q_8$

$\delta(q_7, 0) = q_7$ ×
$\delta(q_3, 0) = q_8$
$\delta(q_7, 1) = q_7$ ×
$\delta(q_3, 1) = q_7$

$\delta(q_7, 0) = q_7$ it is untick
$\delta(q_4, 0) = q_5$
$\delta(q_7, 1) = q_7$ so leave untick
$\delta(q_4, 1) = q_6$

$\delta(q_7, 0) = q_7$ ×
$\delta(q_5, 0) = q_7$
$\delta(q_7, 1) = q_7$ so leave untick.
$\delta(q_5, 1) = q_8$

$\delta(q_7, 0) = q_7$ ×
$\delta(q_6, 0) = q_7$
$\delta(q_7, 1) = q_7$ ×
$\delta(q_6, 1) = q_8$

~~$\delta(q_7, 0)$~~ = now check for q_8.
~~$\delta(q$~~

$\delta(q_8, 0) = q_8$ × $\delta(q_8, 1) = q_8$ } it is untick.
$\delta(q_0, 0) = q_1$ $\delta(q_0, 1) = q_4$

$\delta(q_8, 0) = q_8$ × $\delta(q_8, 1) = q_8$ ×
$\delta(q_1, 0) = q_2$ $\delta(q_1, 1) = q_3$

$\delta(q_8, 0) = q_8$ × $\delta(q_8, 1) = q_8$ ×
$\delta(q_2, 0) = q_7$ $\delta(q_2, 1) = q_8$

$\delta(q_8, 0) = q_8$ × $\delta(q_8, 1) = q_8$ ×
$\delta(q_3, 0) = q_8$ $\delta(q_3, 1) = q_7$

$\delta(q_8, 0) = q_8$ × $\delta(q_8, 1) = q_8$ ×
$\delta(q_4, 0) = q_5$ $\delta(q_4, 1) = q_6$

$\delta(q_8, 0) = q_8$ × $\delta(q_8, 1) = q_8$ ×
$\delta(q_5, 0) = q_7$ $\delta(q_5, 1) = q_8$

$\delta(q_8, 0) = q_8$ × $\delta(q_8, 1) = q_8$ ×
$\delta(q_6, 0) = q_7$ $\delta(q_6, 1) = q_8$

$\delta(q_8, 0) = q_8$ × $\delta(q_8, 1) = q_8$ ×
$\delta(q_7, 0) = q_7$ $\delta(q_7, 1) = q_7$

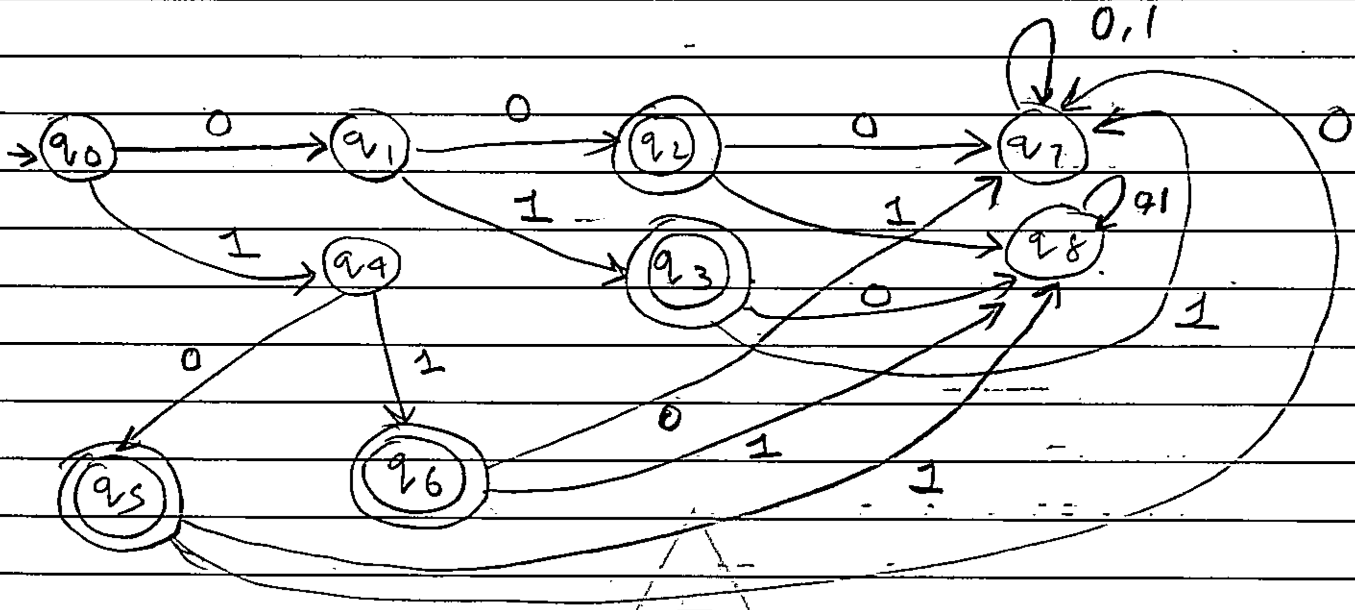
Now again check for untick box

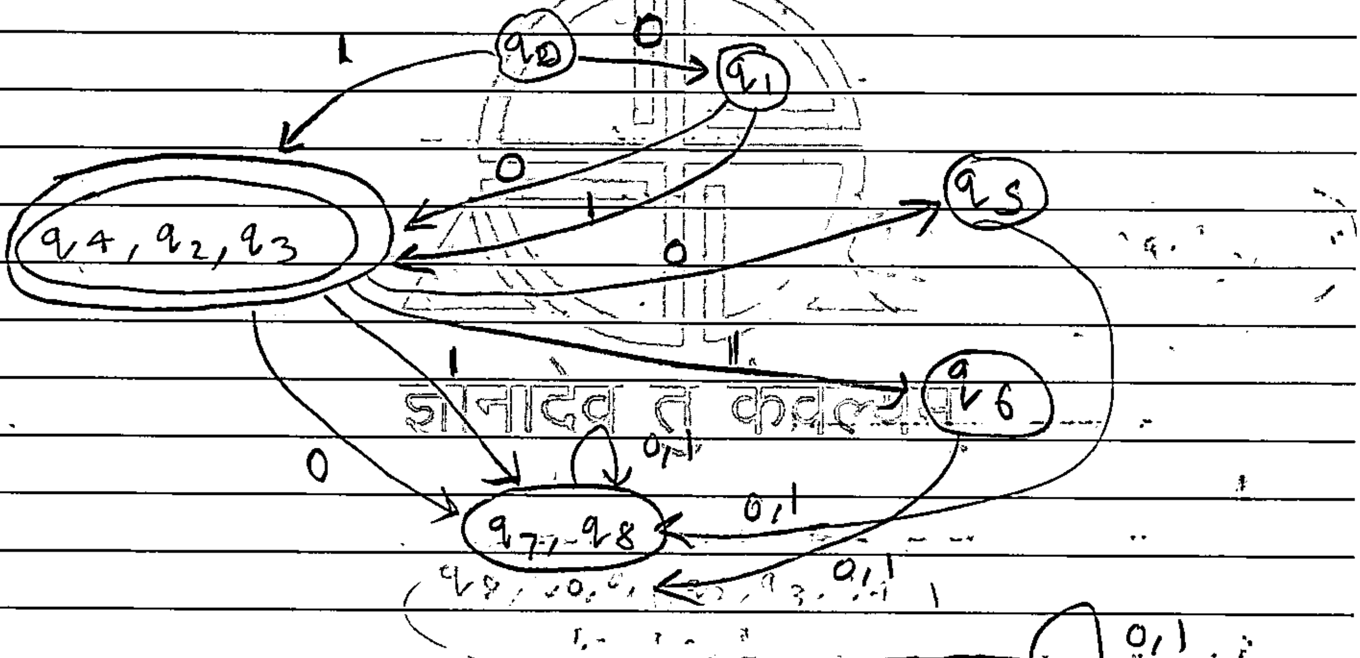
$\delta(q_4, 0) = q_5$ × $\delta(q_4, 1) = q_6$
$\delta(q_2, 0) = q_7$ $\delta(q_2, 1) = q_8$
So leave untick.

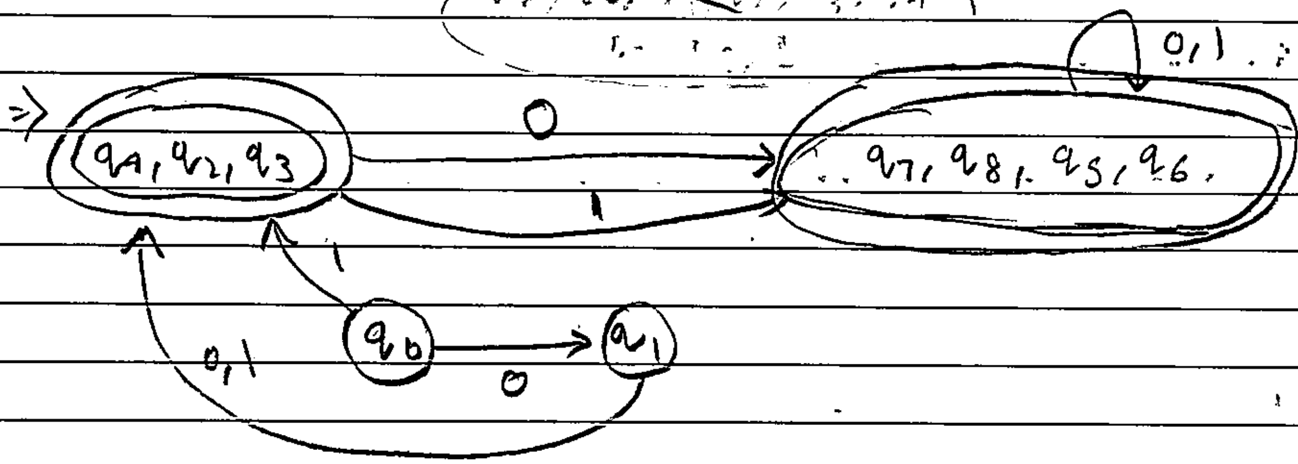
$\delta(q_4, 0) = q_5$ × $\delta(q_4, 1) = q_6$ ×
$\delta(q_3, 0) = q_8$ $\delta(q_3, 1) = q_7$

Now pair up the unmarked boxes

: (q_4, q_2) and (q_4, q_3)
the common is q_4
therefore : (q_4, q_2, q_3)
and
$(q_7, q_0, q_1, q_2, q_3, q_4, q_5, q_6)$
and
$(q_8, q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7)$.


q0
q1
q2
q7
q8
q3
q4
q5
q6
0,1
0
1


q0
q1
q4, q2, q3
q5
q6
q7, q8
0,1
0
1


q4, q2, q3
q7, q8, q5, q6
q0
q1
0,1
0
1

UNIT-2

Q.2)

a) There are various closure property of regular grammar.

1) Union of regular grammar

$L_1 = \{\varepsilon, a, aa, aaa \ldots\}$
$L_2 = \{a, aa, aaa \ldots\}$
$L_1 \cup L_2 = \{\varepsilon, a, aa, aaa \ldots\}$

2) Intersection of regular grammar

$L_1 = \{\varepsilon, a, aa, aaa \ldots\}$
$L_2 = \{a, aa, aaa \ldots\}$
$L_1 \cap L_2 = \{a, aa, aaa, \ldots\}$

3) kleen clousure

$a^* = \{\varepsilon, a, aa, aaa, aaaa \ldots\}$

4) positive clousure

$a^+ = \{a, aa, aaa \ldots\}$

Q.2)

c)

let L be a regular grammar. we have to prove that L is not regular.
we will use pumping lemma theorem.

The following three conditions should be match.

i) $|y| > 0$
ii) $|xy| \le P$
iii) for every $i \geq 0$, xy^iz, let $P = 4$,

$\therefore$ $L = \{aaaabbbb..\}$

Case-1: $x = \epsilon$, $y = aaaabbbb$, $z = \epsilon$

$i=2$; (aaaabbbb)(aaaabbbb) mismatch.

Case-2: $x = a$, $y = aaaa$, $z = bbbb$

ज्ञानादेव तु कैवल्यम्

$i=2$: aaaaaaaaabbbb mismatch

Case-3: $x = aaaa$, $y = bbb$, $z = b$

$i=2$; aaaabbbbbbb mismatch

Case 4: $x = aa$ $y = aabb$ $z = bb$

$i=2$; aaaabbaabbbb mismatch

i) $|y| > 0$

$8 > 0$
$4 > 0$
$3 > 0$
$4 > 0$

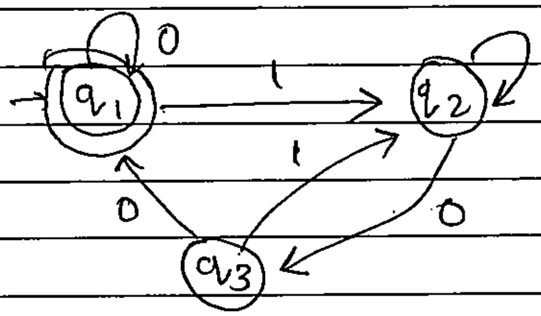
ii) $|xy| \leq p$

$8 \leq 4$ ✗
$4 \leq 4$ ✓
$3 \leq 4$ ✓
$4 \leq 4$ ✓

hence, from above observation we can say that the

$L = \{a^n b^n\}$ is not regular.

Q. 2)
d)



by applying arden's theorem, we can calculate the regular expression, Arden's theorem, says that P and Q are transition states, then if P is not null, then,

$R = Q + RP$, it can be written as,

$R = QP^*$

So, first we will write the incoming states

~~$q_0 = \varepsilon +$~~

$q_1 = \varepsilon + q_1 0 + q_3 0$ — (1)

$q_2 = 1 q_2 + q_3 1$ — (2)

$q_3 = q_2 0$ — (3)

now put eqn (3) into eqn (2)

$\therefore q_2 = 1 q_2 + (q_2 0) 1$

$q_2 = q_2 (1 + 01)$ in the form of $R = Q + RP$, $Q = \varepsilon$

$q_2 = \varepsilon . (1 + 01)^*$

$\boxed{q_2 = (1 + 01)^*}$ — (4)

put the 4^{th} eqn into q_3

$q_3 = q_2 0$

$\boxed{q_3 = (1+01)^* 0}$ — (5)

now, put the 5^{th} eq^n into eq (1)

$q_1 = \varepsilon + q_1 0 + q_3 0$

$q_1 = \varepsilon + q_1 0 + ((1+01)^* 0) 0$

$\cancel{q_1 = q_1 (0 + (1+01)^* 00)}$

$\underset{R}{q_1} = \varepsilon + \underbrace{(1+01)^* 00}_{Q} + \underset{RP}{q_1 0}$ $(R = QP^*)$

$\therefore \boxed{q_1 = (1+01)^* 00 . 0^*}$

so, the final state here is q_1

$\therefore \boxed{q_1 = (1+01)^* 00 . 0^*}$

final answer

Q.3)

a)

2)

all strings ending with 00

will be $(0+1)^* 00$

in starting any combination be 0 and 1 can be occured but in the end it should be 00

$\therefore \boxed{(0+1)^* 00}$

1)

$L = \{ 00001, 000111, 0000111111000, 10101, \cancel{000}\ 111100, \}$

0^*

3)
b)

chomsky classification of grammar is a classification of grammar is as follows

Type-0 → Unrestricted Grammar
↓
Type-1 → Context sensitive Grammar
↓
Type-2 → Context free grammar
↓
Type-3 → Regular grammar

Type-0

It is the most powerful among all as it is unrestrited from any kind of restriction. for ex- Turing machine, which have the ability to move left, Right, in a infinite tape, with read, write head in the tape head. It is called as unrestrited grammar because it does not have any kind of restriction, and can solve any algorithm based problem.

Type-1

It is less powerful than type-0, as once we put some restrictions on type-0, then it will ~~becao~~ become type-1. and this hierarch is known as context sensitive grammar and can't perform most of the tasks of Type-0.

Type-2

It is known as context free grammar, and once we put some restrictions on type-1, then it is known as Type 2.

Type-3

It is known as regular grammar, and it is the most restricted grammar of the heirarchy.

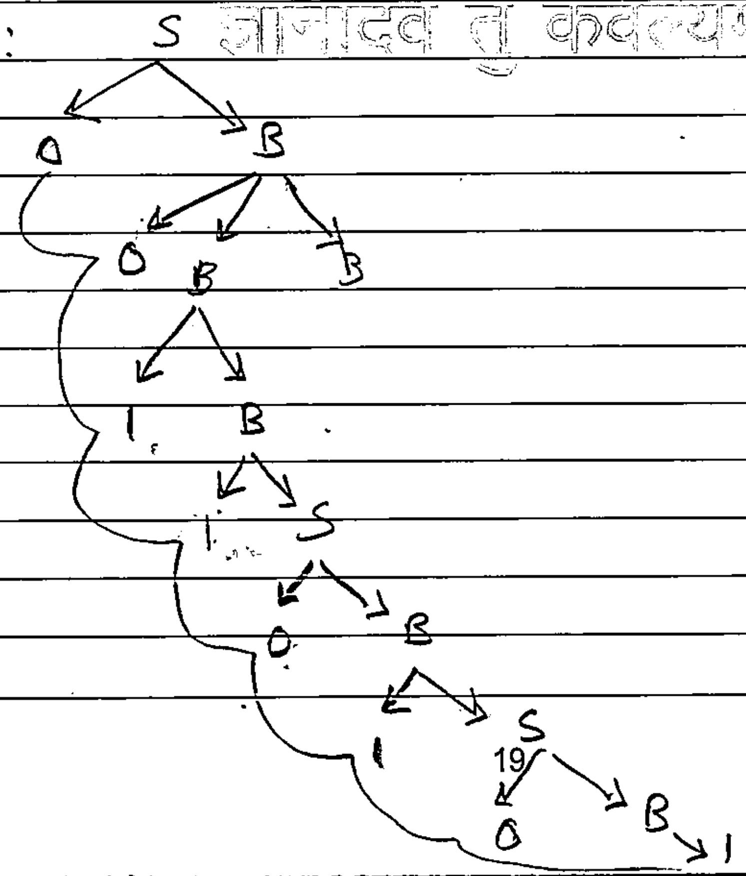
Q.3)
d)

$S \rightarrow 0B \mid 1A$
$A \rightarrow 0 \mid 0S \mid 1AA$
$B \rightarrow 1S \mid 1 \mid 0BB$

w = 00110101

LMD:		R.M.D:	
S		S	
ØB	Ø0110101	ØB	Ø0110101
ØBB	Ø110101	ØBB	Ø110101
~~1~~B	~~1~~10101	BS	110101
~~1~~S	~~1~~0101	BØB	11Ø101
ØB	Ø101	B1S	11101
~~1~~S	~~1~~01	B11A	11101
ØB	Ø1	B110	1110
~~1~~	~~1~~	~~1110~~	~~1110~~

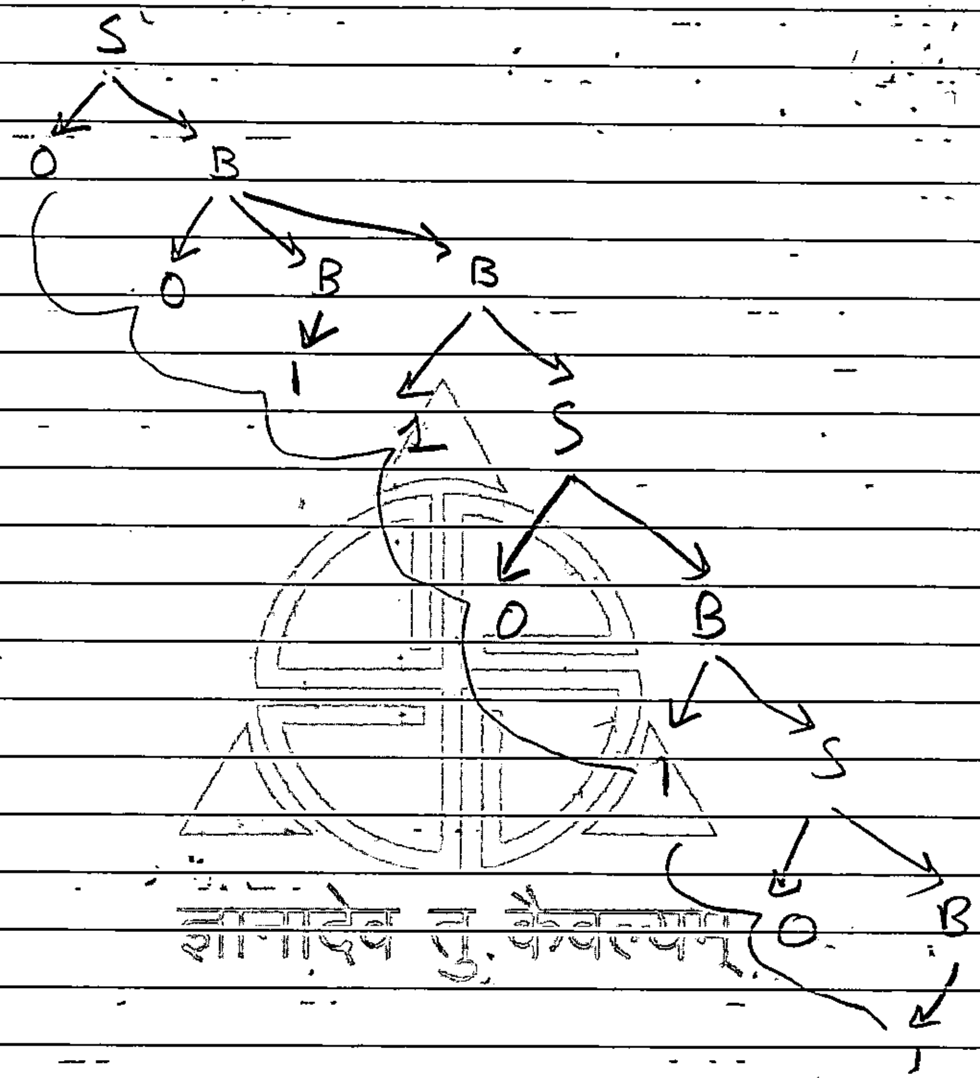
LMD:



Q.3)

RMD :

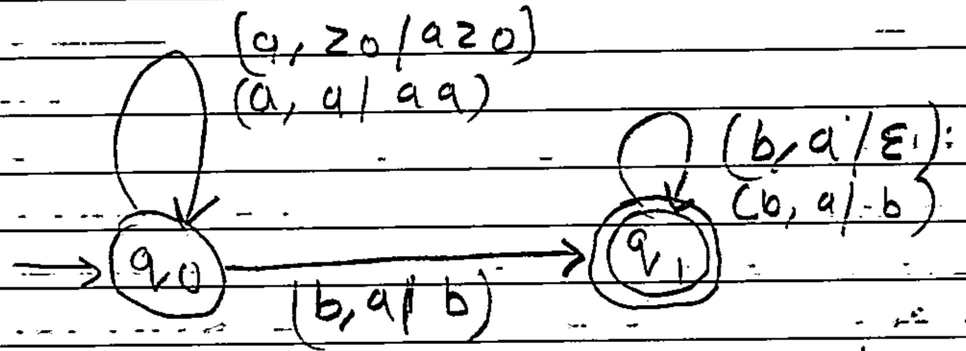
00110101



Q.4)
b)

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

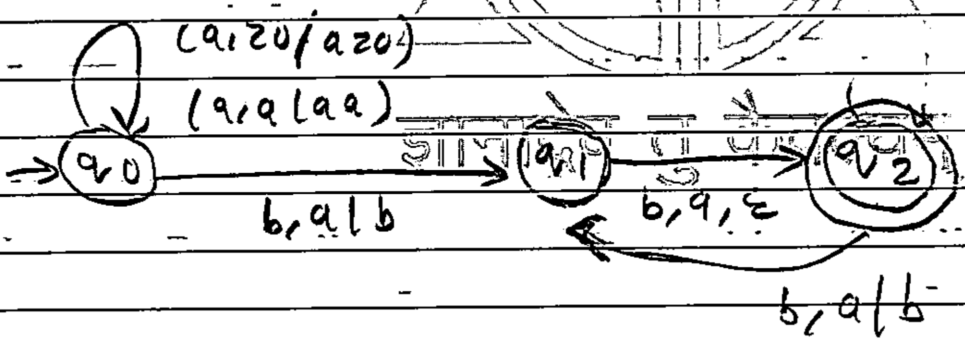
$L = \{ aabbbb, aaabbbbbb, aaaabbbbbbbb \}$



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$
$\delta(q_0, a, a) = (q_0, aa)$

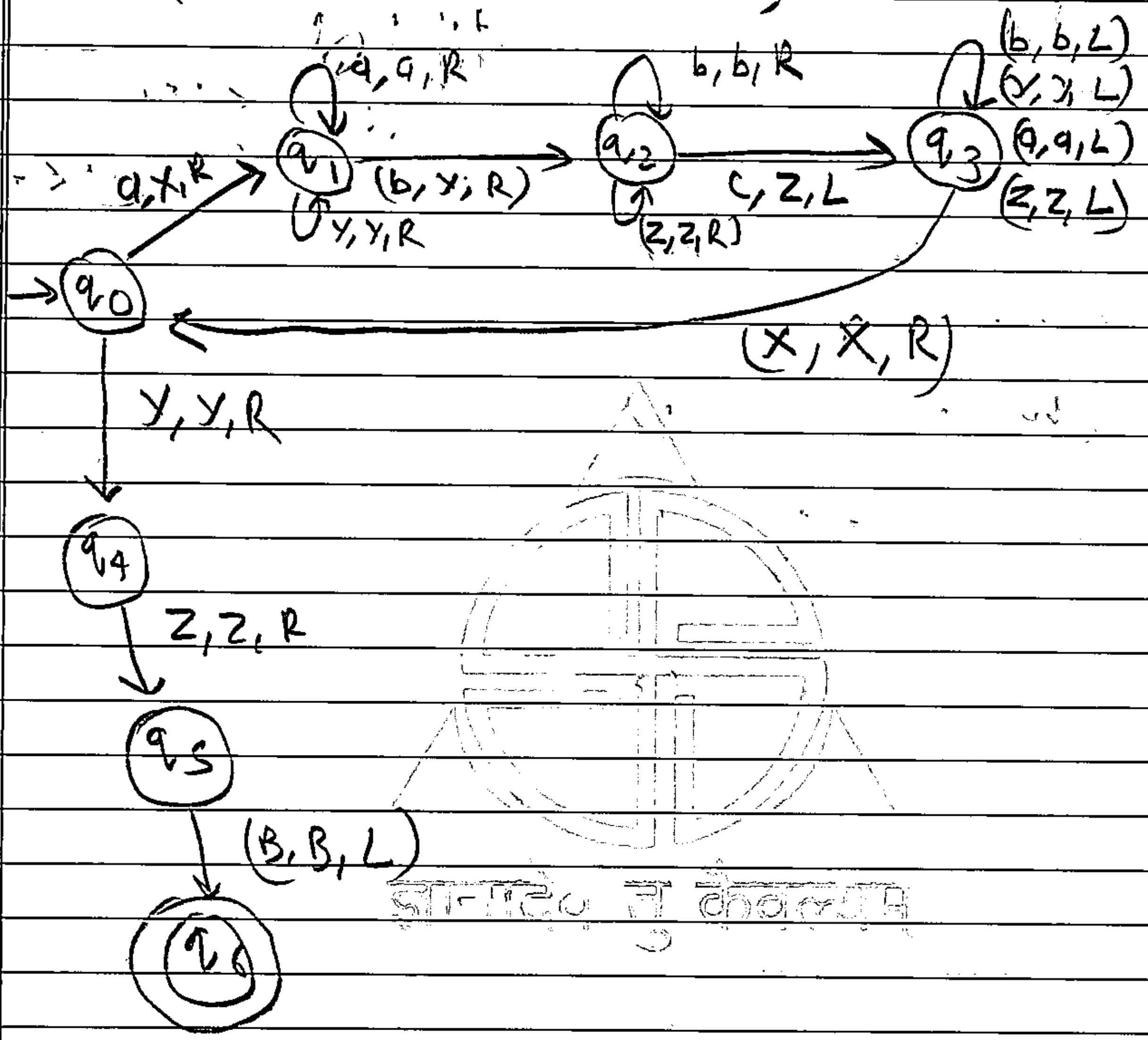
$\delta(q_0, b, a) = (q_1, b)$

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



Q.4)

c) L = { B B B a a a b b b C C C B B }



Q.5)

a)

partial function is the function which determines ~~the~~ not every solution, for a particular problem.

for ex:

$f(x) = \frac{1}{a}$; $a \neq 0$ (it does not define for $a = 0$).

initial function can be of three types

i) zero function $z(x) = 0$.

ii) successor function $S(x) = x+1$

iii) Projection function $U_2^3(0,1,2) = 1$

Q.5)

b) $f(x, y) = x * y$

To prove, it is primitive we have to follow

initial, composition, recursive function.

$f(x,y) = xy$

i) put $y = 0$

$f(x,0) = 0$

$z(f(x,0)) = 0$

ii) put $y = y+1$

$f(x,y) = xy$

$= x(y+1)$

$= xy + 1$

$= S(f(x,y)) = xy+1$

$\therefore$

$g(x, y, f(x,y)) = U_3(x, y, S(f(x,y)))$

2) $f(x,y) = x^y$

i) put $y = 0$

$f(x,y) = x^0 = 1$

$\therefore S(z(f(x,y))) = S(0) = 1$

ii) put $y = y+1$

$x^y = x^{y+1}$

$= \underline{x^y + x^1}$

$\cdot f(x,y)$ # when $y=0$, $\therefore x$.

$\therefore U_3^2(x, Z(y), f_1(x,y)) + U_3^3(x, y, f(x,y))$

Q-5)

c)

Space time complexity $O(n)$

time complexity depends on length of turing machine

$\therefore$

$O(n \log n)$