UNIT – I

Ans 1(a)

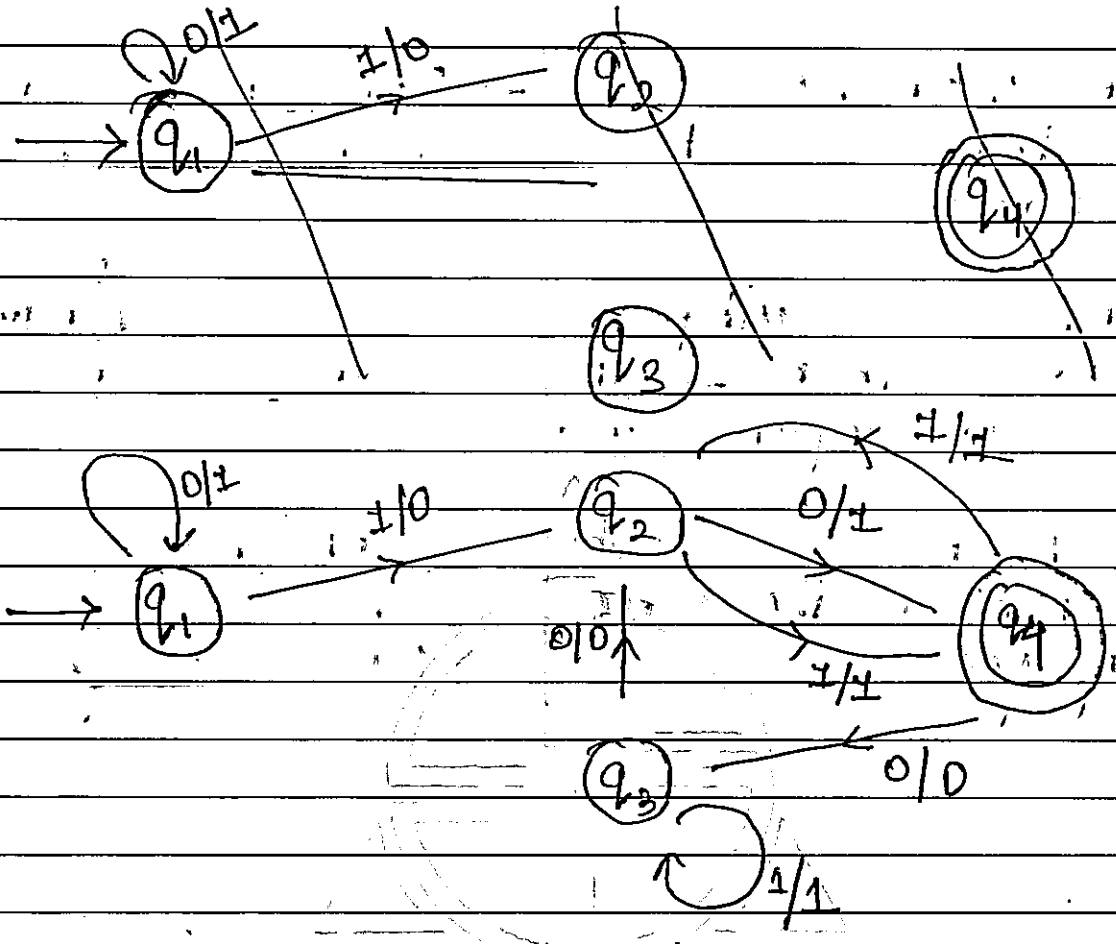
Differences between NFA and DFA

1. NFA stands for Non-Deterministic Finite Automata	1. DFA stands for deterministic Finite Automata.
2. NFA has ~~multip~~ one or more than 1 path for one input value.	2. DFA has only one path for one input value
3. NFA has less number of states for the same language	3. DFA has more number of states for the same language.

Ans Given: Mealy Machine transition table

	q = 0		q = 1	
	State	output	State	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

Finite Automata for the Moore Machine



q_2 and q_3 have different outputs

So,

q_2 → q_{20} → 0

q_2 → q_{21} → 1

q_3 → q_{30} → 0

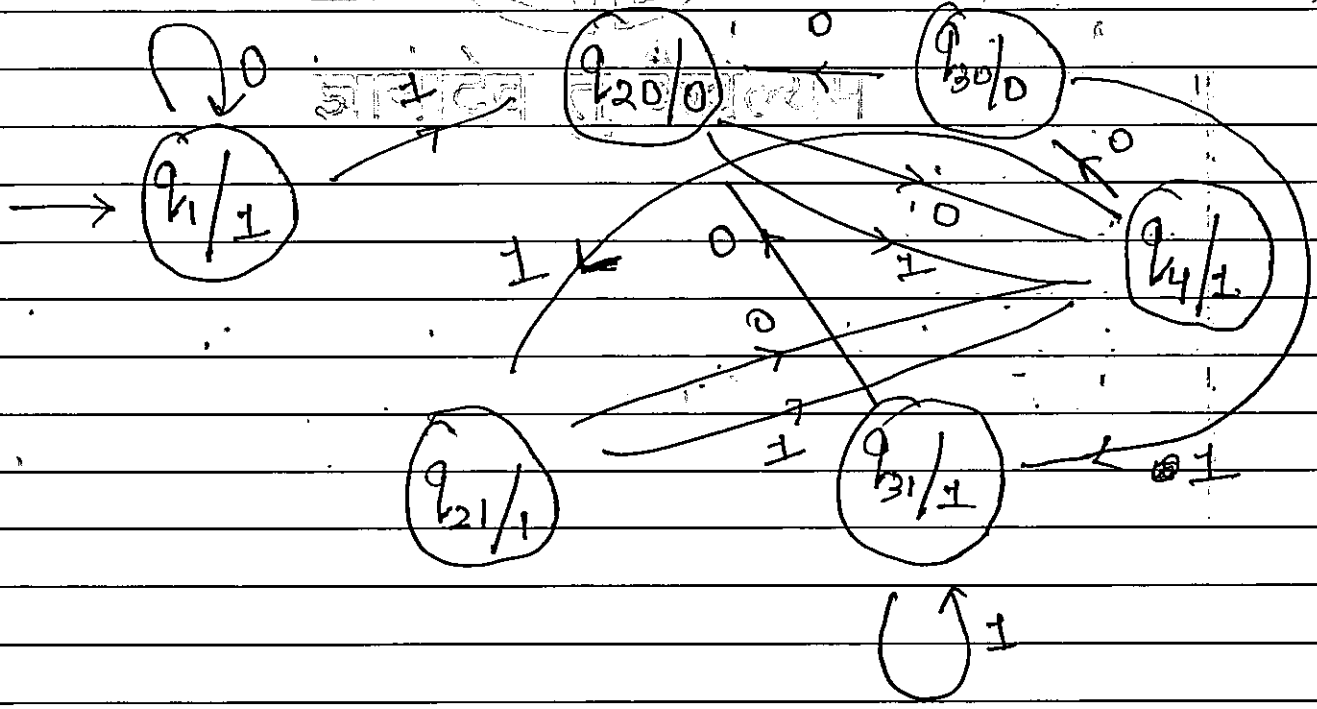
q_3 → q_{31} → 1

q_{20}, q_{21}, q_{30}, q_{31} will be the new states in the NFA in place of q_2 and q_3 Moore Machine

Transition table for N.F.A –

	q = 0 State	q = 1 State	Output
→ q_1	q_1	q_{20}	1
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{30}	q_{20}	q_{31}	0
q_{31}	q_{20}	q_{31}	1
q_4	q_{30}	q_{21}	1

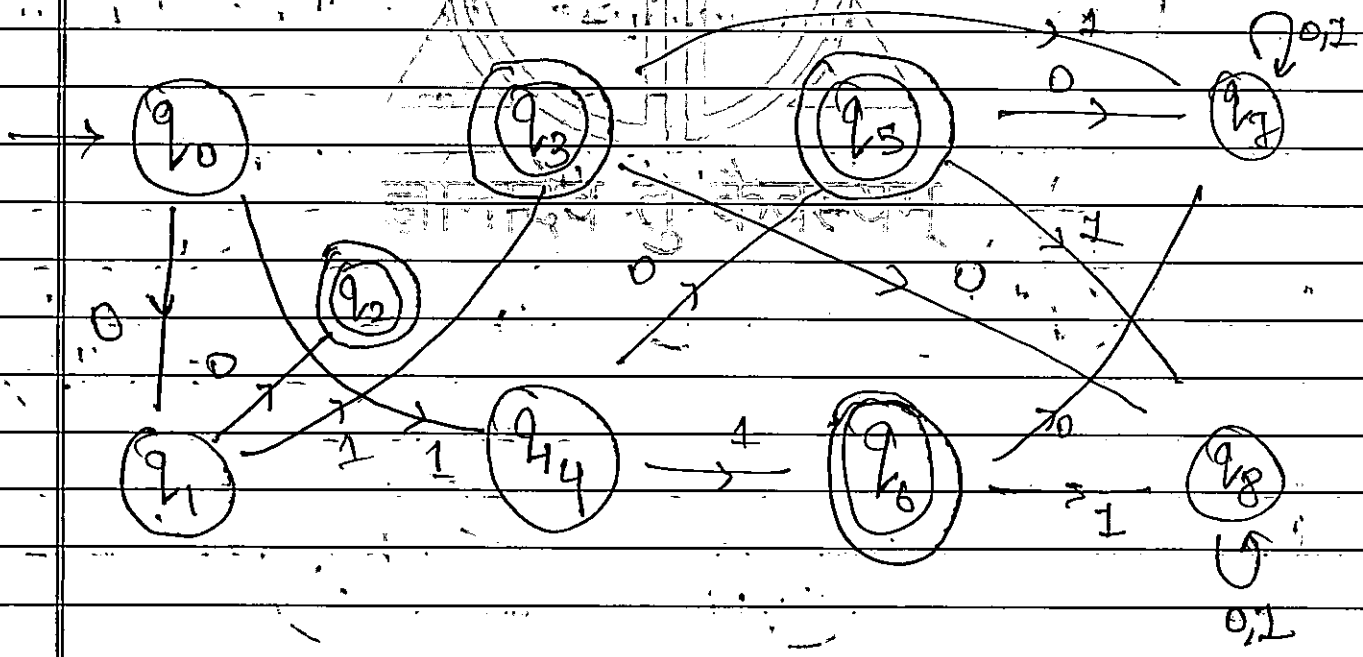
Finite Automachine of Moore Machine



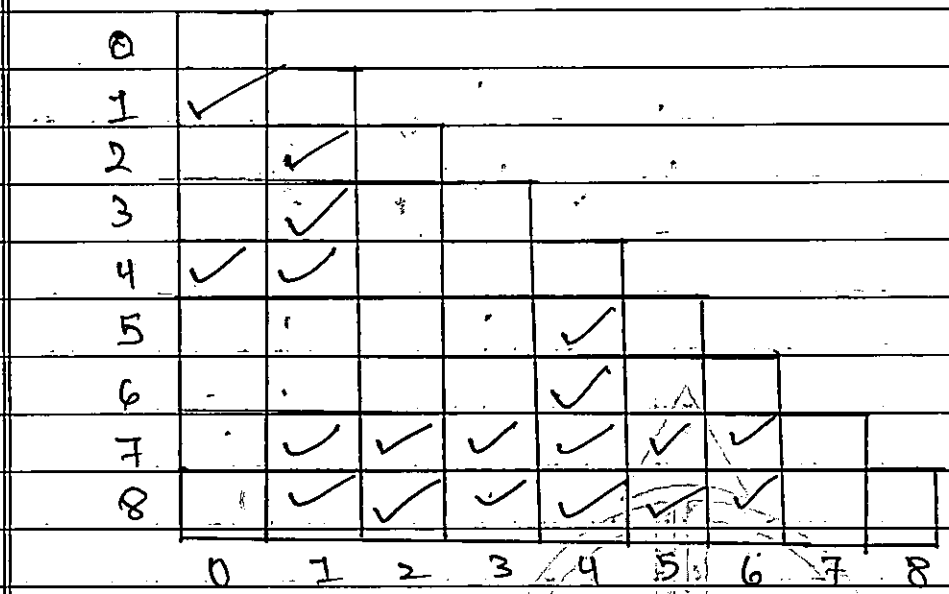
Ans 1 (c) Given DFA transition table:-

State/Σ	0	1
→ q_0	q_1	q_4
q_1	q_2	q_3
(q_2)	q_7	q_8
(q_3)	q_8	q_7
q_4	q_5	q_6
(q_5)	q_7	q_8
(q_6)	q_8	q_8
q_7	q_7	q_7
q_8	q_8	q_8

Finite Automata of DFA transition table:-



Making half matrix for the DFA



The given condition can be written as

$[q_0 q_1 q_4 q_7 q_8]\ [q_2][q_3][q_5][q_6]$

~~Note~~ ✓

Possible pairs → q_0q_1, q_0q_4, q_0q_7, q_0q_8, q_1q_4

~~q_1q_1~~ q_1q_7, q_1q_8, q_4q_7, q_4q_8, q_7q_8

	input = 0	input = 1	
q_0q_1	q_0 q_1 (q_2)	q_4 (q_3)	✗
q_0q_4	q_0 q_4 (q_5)	q_4 (q_6)	✗
q_0q_7	q_0 q_7	q_4 q_7	✓

$q_0 q_8$ — q_0 q_1 q_4 / q_8 q_8 q_8 ✓

$q_1 q_4$ — q_1 q_2 q_3 / q_4 q_5 q_6 ✓

$q_1 q_7$ — q_1 q_2 q_3 / q_7 q_8 q_7 ✗

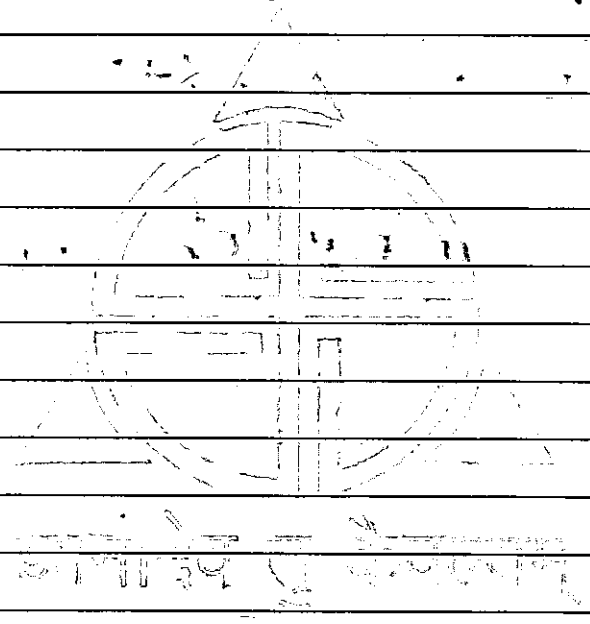
$q_1 q_8$ — q_1 q_2 q_3 / q_8 q_8 q_8 ✗

$q_4 q_7$ — q_4 q_5 q_6 / q_7 q_7 q_7 ✗

$q_4 q_8$ — q_4 q_5 q_6 / q_8 q_8 q_8 ✗

$q_7 q_8$ — q_7 q_7 q_7 / q_8 q_8 q_8 ✓

After → $[q_0 q_7 q_8]$ $[q_2]$ $[q_3]$ $[q_5]$ $[q_6]$ $[q_1]$ $[q_4]$



UNIT — II

Ans 2(a) Closure property of regular grammar refers to:

K^* ; here K^* contains all values of K including NULL values.

$K = \{\epsilon, K, K+1, K+2 \cdots \infty\}$

n = number of inputs.

UNIT – II

Ans b) Given regular expression

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

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

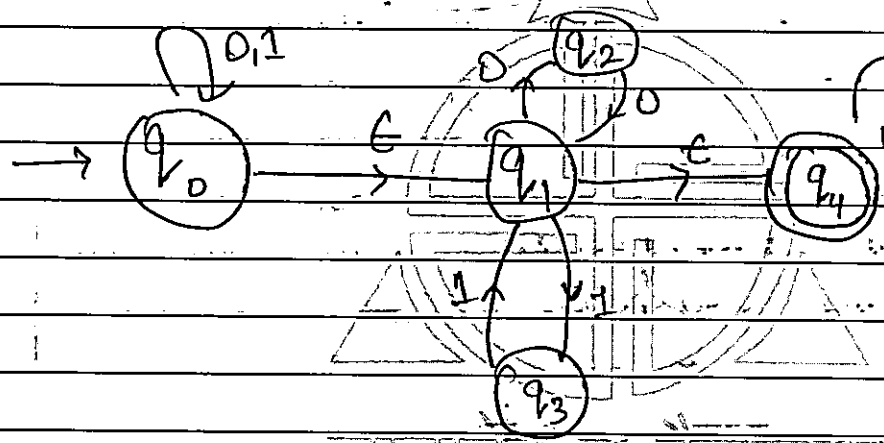
$\Sigma = \{0, 1\}$

$\delta = Q \times \Sigma$

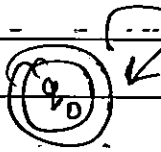
$q_0 \rightarrow q_0$

$F \rightarrow q_4$

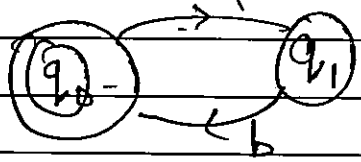
Finite automata for the regular expression



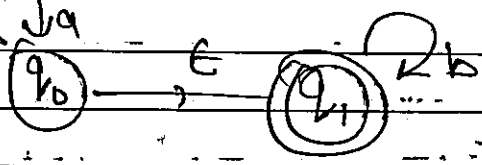
$(a+b)^*$ =



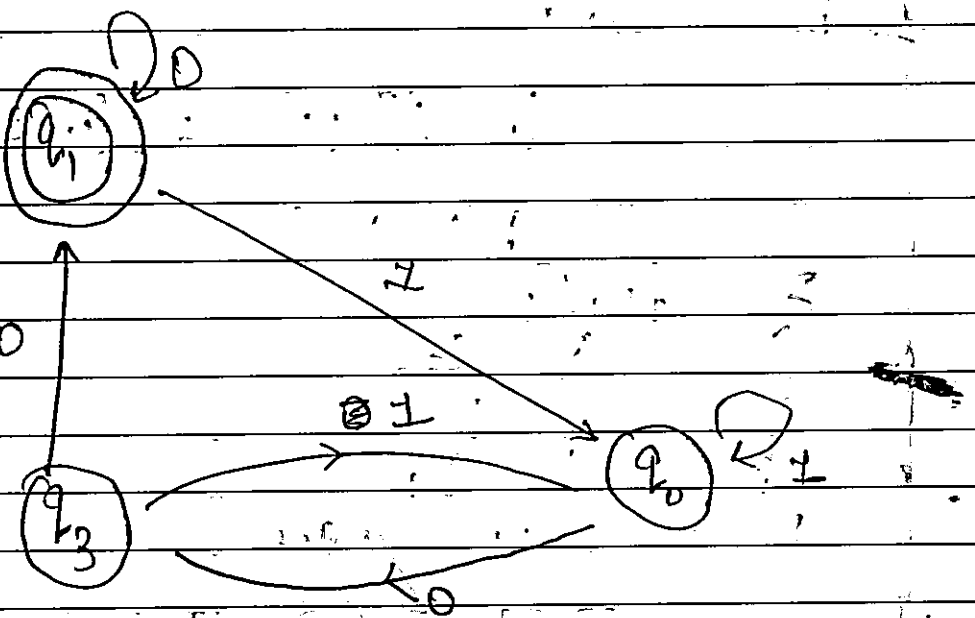
$(aa)^*$ =



a^*b^* =

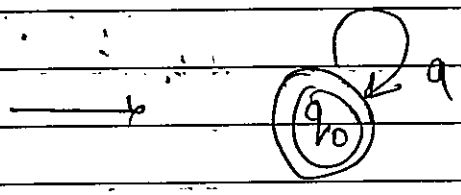


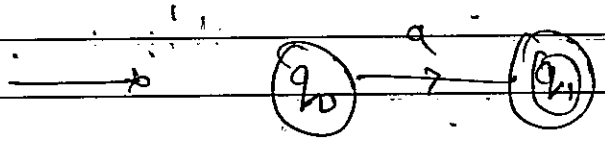
Ans 2(d) Given Automata

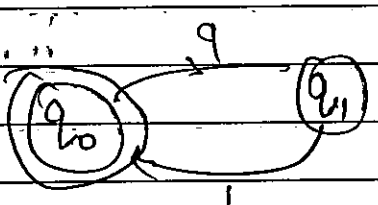


Regular expression from the following transition system :—

$$R.E = 0^*(1.1^*(01)^*0)^*$$

a^* 

a 

$(ab)^*$ 

UNIT – III

Ans 3(a)(i) Regular expression for the set of all string having odd number of 1's

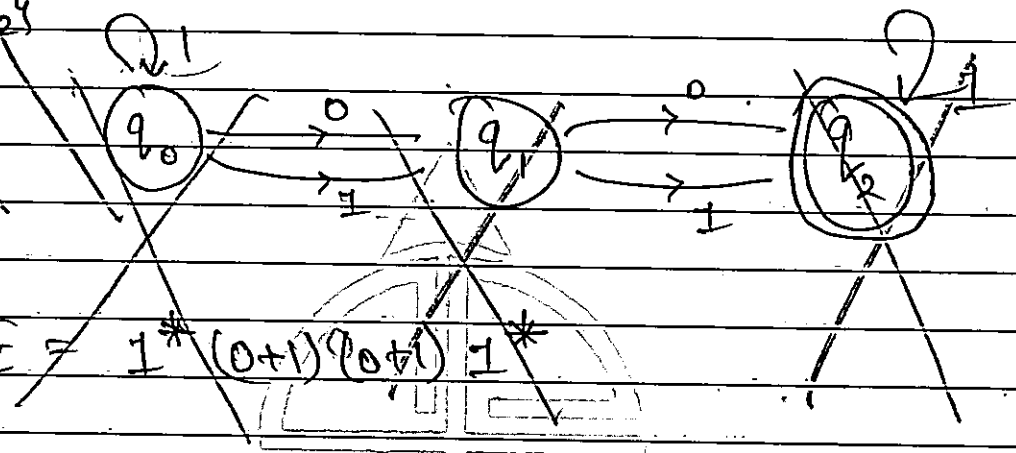
Strings with odd number of 1's = {1, 01, 10, 111, 1011 ...}

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

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

$q_0 \rightarrow q_0$

$F \rightarrow q_2$



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

Ans

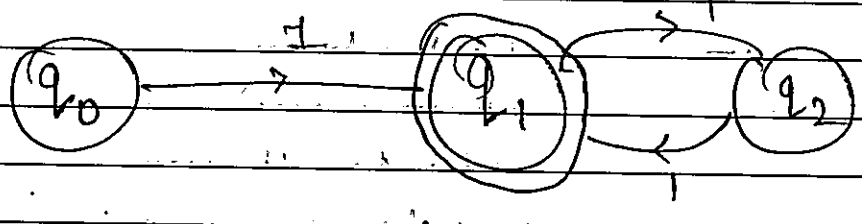
$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{1\}$

$q_0 \rightarrow q_0$

$F \rightarrow q_1$

$\delta \rightarrow Q \times \Sigma$



Regular Expression = $1(11)^*$

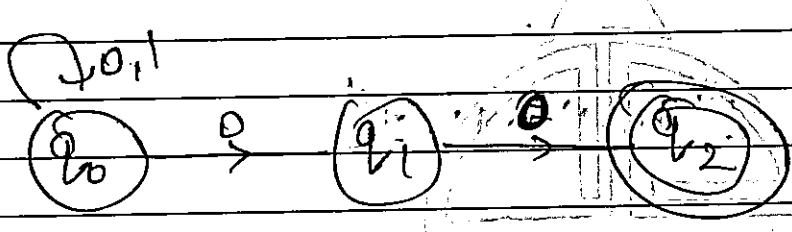
Ans 3(a)(2) Regular Expression for the set of all string ending in 00 :—

$Q \Rightarrow \{q_0, q_1, q_2\}$

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

$q_0 \rightarrow q_0$

$F \rightarrow q_2$

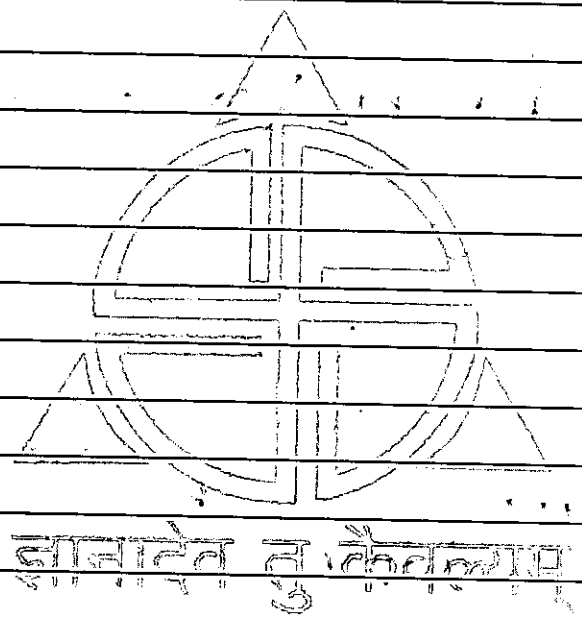


Regular Expression $= (0+1)^* 00$

Ans 3(b) Chomsky classification of grammars has a total of 4 types to determine the language used :

Grammar Type →	
Type 0	Recursive Enumerable Grammar
Type 1	Context Sensitive Grammar

Type 2	Context Free Grammar
Type 3	Regular Grammar



Ans 3 (d) Given Production rules:

$S \longrightarrow 0B \mid 1A$

$A \longrightarrow 0 \mid 0S \mid 1AA$

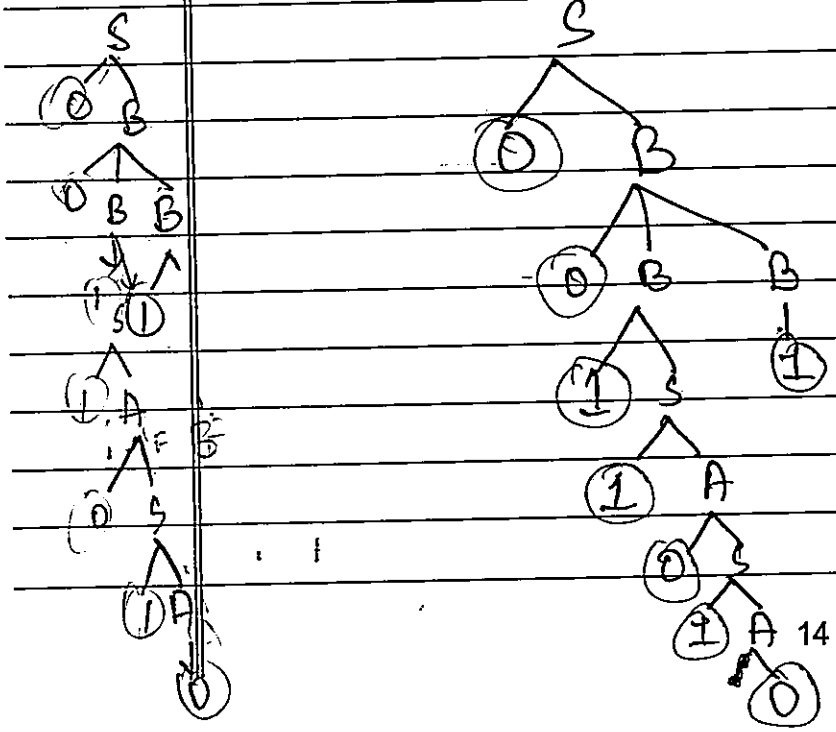
$B \longrightarrow 1 \mid 1S \mid 0BB$

Given string $w = 00110101$

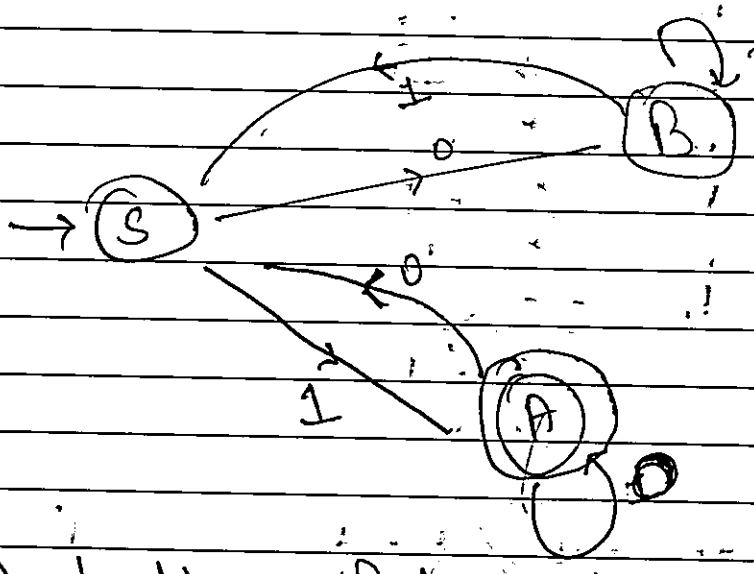
$G = \{$

Step 1 Finite Automata for the given string

Step 2 Derivation tree for the following string



Step 3 Finite Automata of the Derivation tree

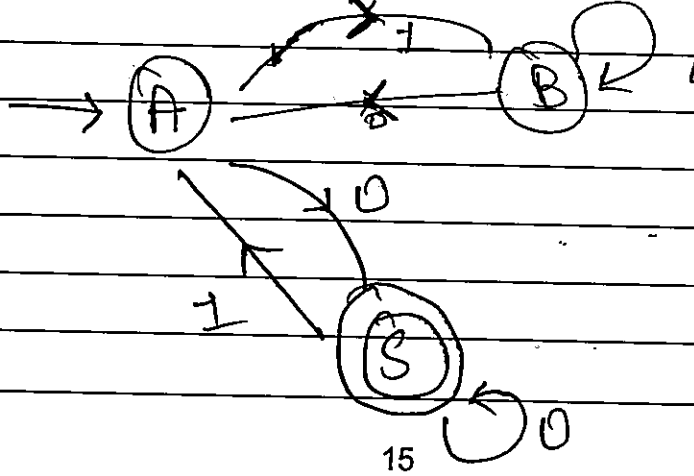


Production Rule LMD

S → 0B
B → 0BB
B → 1S
B → 1

Step 4 Swap the initial and final state and the direction of transition

New Finite Automata



LMD of the new Finite Automata

A → 0S

A → 1B

B → 0B

B → 1B

B → 0A

A → 0

S → 1A

S → 0

Step 5 Swap the initial and final state and write RMD

S → A0

S → B1

B → B0

B → B1

B → S0

S → 0

A → S1

A → 0

Ans 4(a)

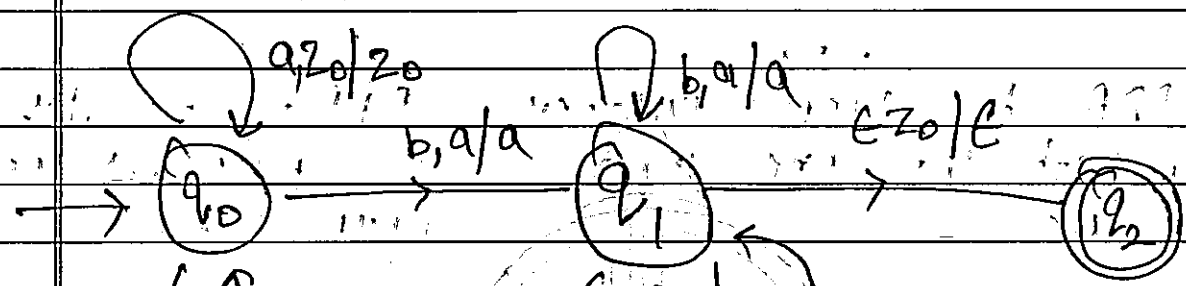
NPDA	DPDA
1) NPDA stands for non-Deterministic pushdown Auto	
1. NPDA stands for Non-deterministic Push Down Automata.	1. DPDA stands for Deterministic Push Down Automata
2. NPDA has more number of path for one input value.	2. DPDA has only one path for one input value.
3. NPDA has less number of states	3. DPDA has more number of states
4. Value at stack can be same Sometimes.	4. Value at stack is unique for all.

Ans 4(b) Given Language

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

Sol → Let's take $n = 2$

W (string) $= aabbbb$



~~FINITE AUTO~~

PUSHDOWN AUTOMATA for given language

$Q \rightarrow \{q_0, q_1, q_2\}$

$q_0 \rightarrow q_0$

$F \rightarrow q_2$

$\Sigma \rightarrow \{Z_0, a, b, t\}$

Ans 4(a) (1) Decidable problems are those problems in which if a language is implemented e.g. $L = \{a^n b^n \mid n \geq 1\}$ and the language is computable then it's turing machine can be Made.

In decidable problem's turing machine gives 'YES' as output.

Undecidable problems are those problems in which if a language is implement and it's grammar is not computable then its turing machine cannot be made and gives 'No' as output are called undecidable problems.

(2) Ans 4 a (3)

Ans 3(a) → Partial functions are those functions in which all input values cannot be shown

e.g. $f(x) = \frac{1}{x}$

Here x cannot be 0 because $\frac{1}{0}$ is undefined.

→ Initial functions are those functions in which the relation can be represented in both numerical and alphabetical form.

e.g. Zero function
Projection function
Successor function etc.

Ans 3(b) Given 2 functions are

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

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

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

$x = 0$ → (def)

$f(2, y) = 2 * y = y + y$

Recursive values

Here the value of y is recurring (calling the value again).

so it is a recursive primitive function

$f(x, y)$

$y = 2$ (let)

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

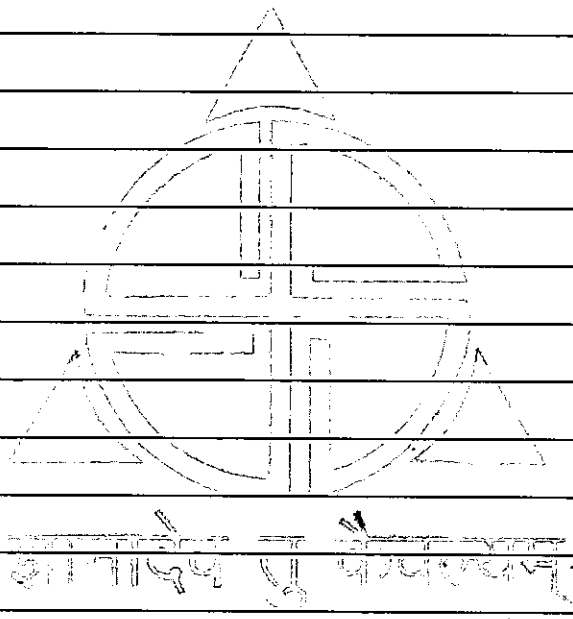
Recursive values

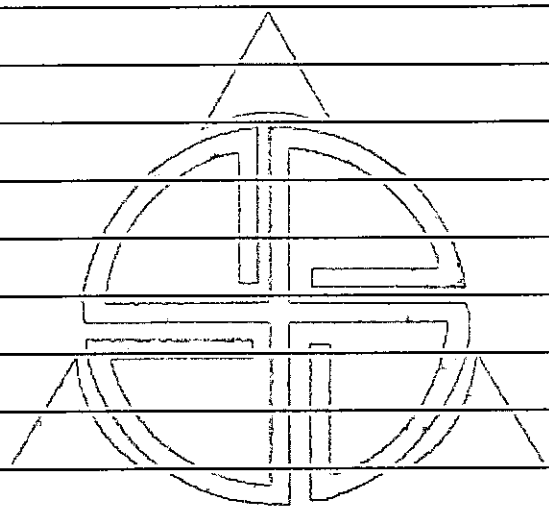
Here we can see that the value of x is recurring (calling the value from function again)

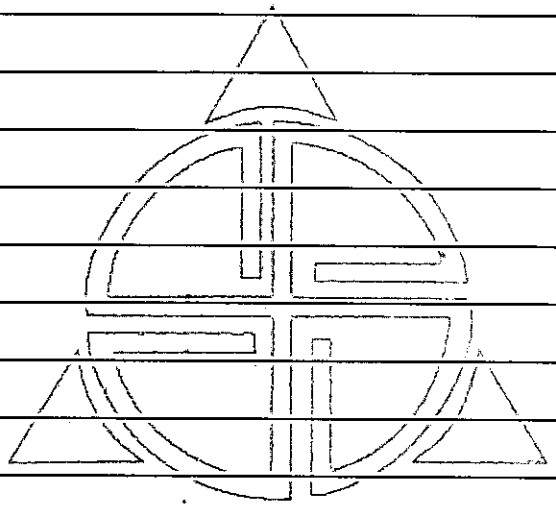
so it is a recursive primitive function

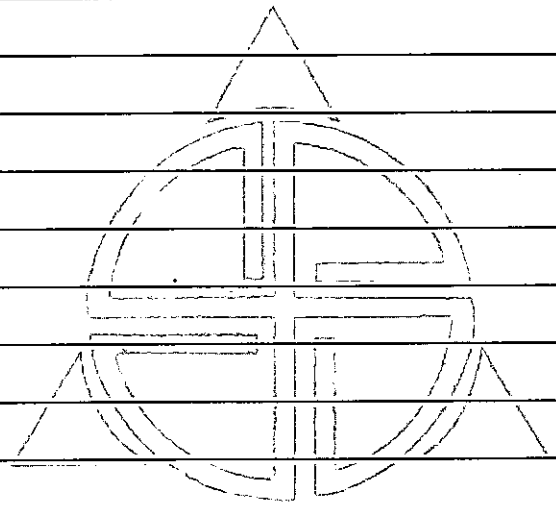
Hence Proved

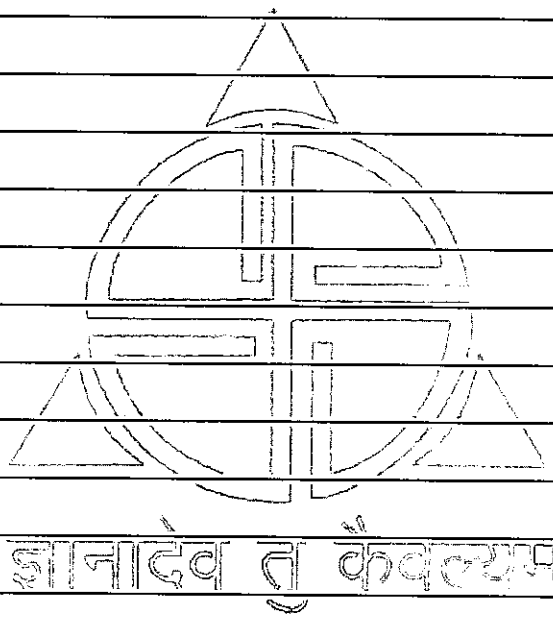
Ans 5(ii) + Space complexity refers to calculation of space taken by a file in the storage.

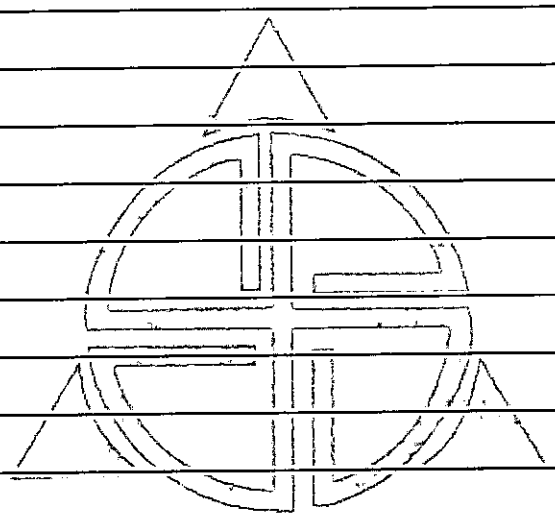


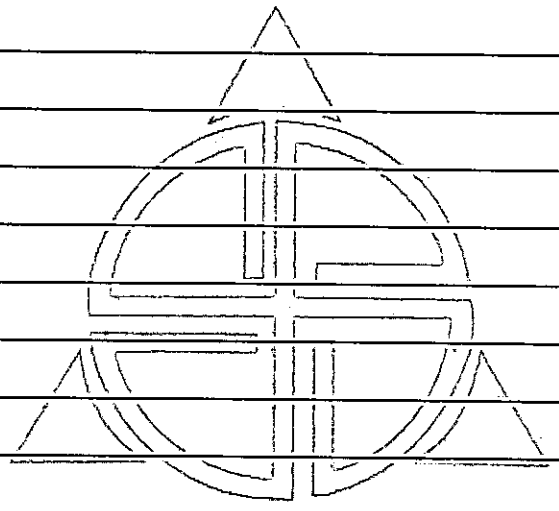

ज्ञानदेव तु केवलम्

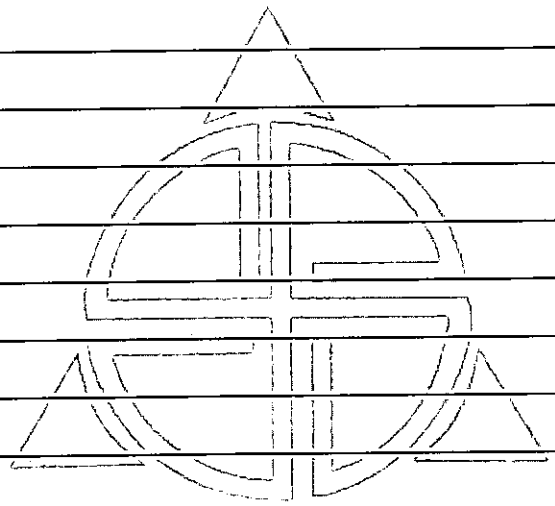

ज्ञानादेव तु कैवल्यम्

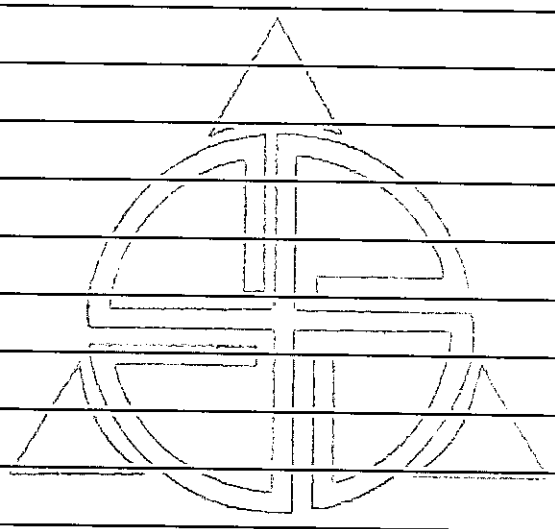


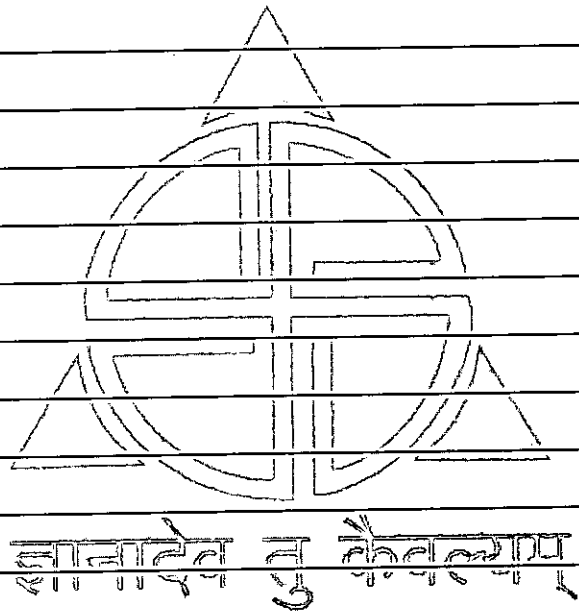

ज्ञानादेव तु कैवल्यम्

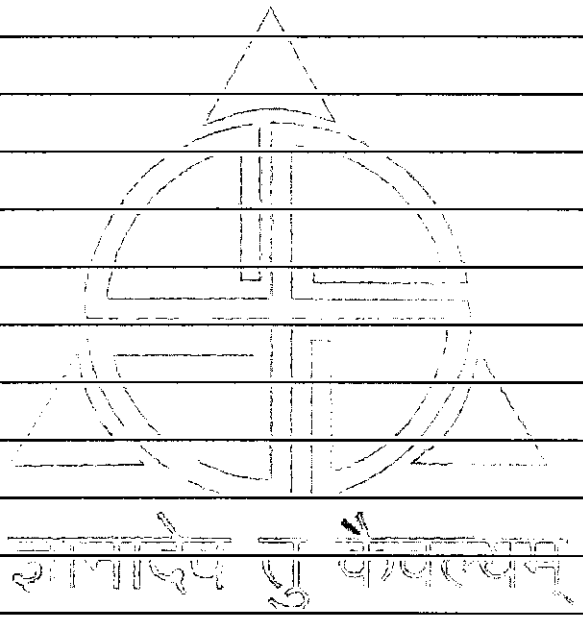


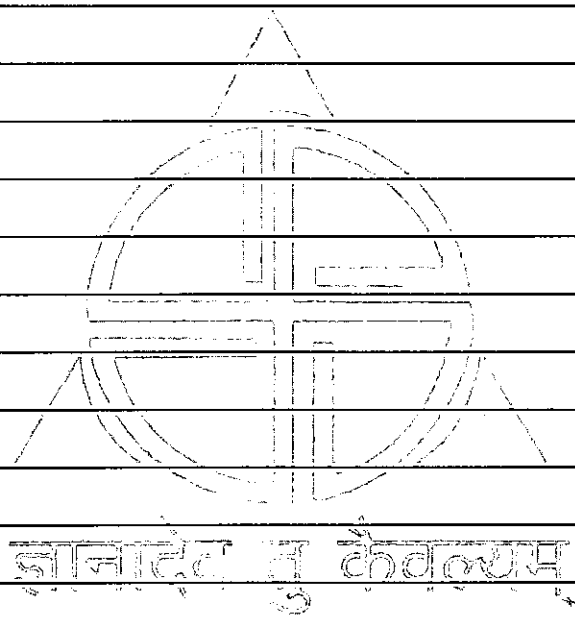


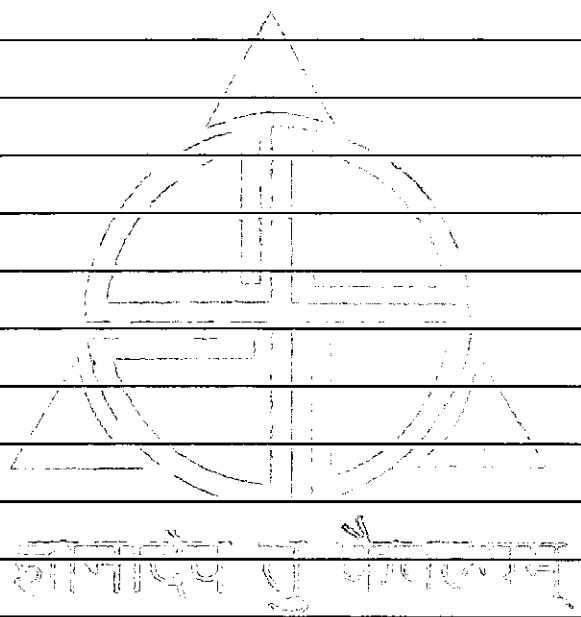


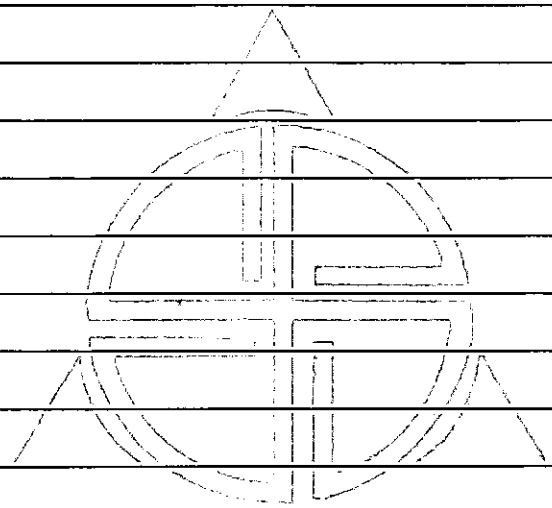


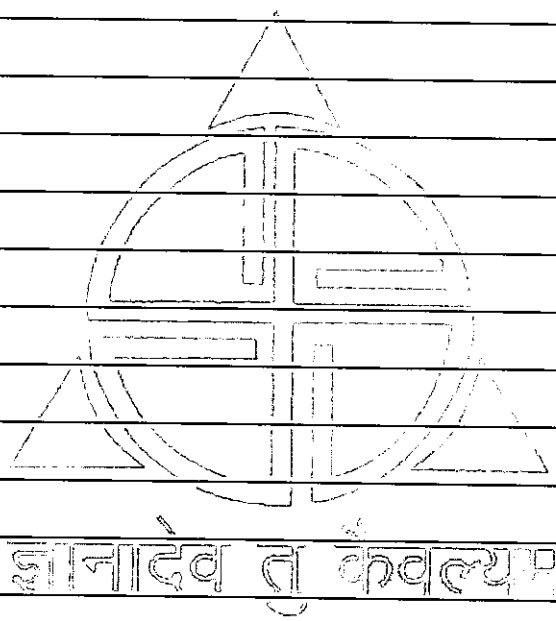


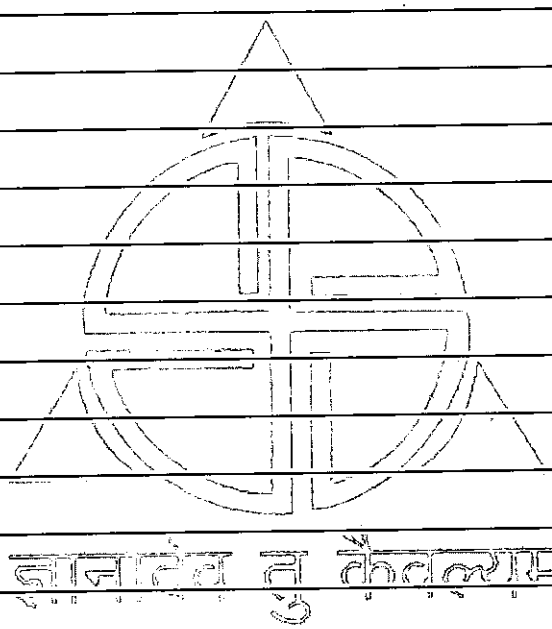


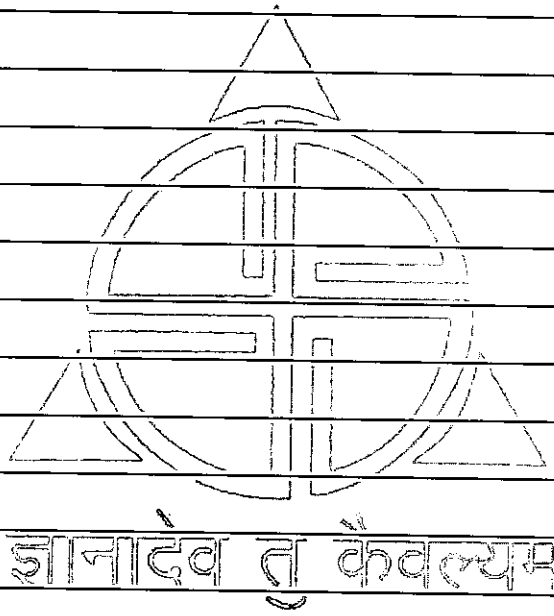





ज्ञानदेव ह गैतमाम


सत्यमेव जयते


सत्यमेव जयते


ज्ञानादेव तु कैवल्यम्