As Per New VTU Syllabus w.e.f 2015-16
Choice Based Credit System(CBCS)

SUNSTAR

SUNSTAR EXAM SCANNER

AUTOMATA THEORY & COMPUTABILITY

(V SEM.B.E. CSE / ISE)

SYLLABUS

Automata Theory & Computability			
[AS PER CHOICE BASED CREDIT SYSTEM (CBCS) SCHEME] (EFFECTIVE FROM THE ACADEMIC YEAR 2016 - 2017)			
Subject Code	15CS54	IA Marks	20
Number of Lecture Hours/Week	04	Exam Marks	80
Total Number of Lecture Hours	50	Exam Hours	03

MODULE 1

Why study the Theory of Computation, Languages and Strings: Strings, Languages. A Language Hierarchy, Computation, Finite State Machines (FSM): Deterministic FSM, Regular languages, Designing FSM, Nondeterministic FSMs, From FSMs to Operational Systems, Simulators for FSMs, Minimizing FSMs, Canonical form of Regular languages, Finite State Transducers, Bidirectional Transducers. **10 Hours**

MODULE 2

Regular Expressions (RE): what is a RE?, Kleene's theorem, Applications of REs, Manipulating and Simplifying REs. Regular Grammars: Definition, Regular Grammars and Regular languages. Regular Languages (RL) and Nonregular Languages: How many RLs, To show that a language is regular, Closure properties of RLs, to show some languages are not RLs. **10 Hours**

MODULE 3

Context-Free Grammars(CFG): Introduction to Rewrite Systems and Grammars, CFGs and languages, designing CFGs, simplifying CFGs, proving that a Grammar is correct, Derivation and Parse trees, Ambiguity, Normal Forms.

Pushdown Automata (PDA): Definition of non-deterministic PDA, Deterministic and Non-deterministic PDAs, Non-determinism and Halting, alternative equivalent definitions of a PDA, alternatives that are not equivalent to PDA. **10 Hours**

MODULE 4

Context-Free and Non-Context-Free Languages: Where do the Context-Free Languages(CFL) fit, Showing a language is context-free, Pumping theorem for CFL, Important closure properties of CFLs, Deterministic CFLs. Algorithms and Decision Procedures for CFLs: Decidable questions, Un-decidable questions.

Turing Machine: Turing machine model, Representation, Language acceptability by TM, design of TM, Techniques for TM construction. **10 Hours**

MODULE 5

Variants of Turing Machines (TM), The model of Linear Bounded automata: Decidability: Definition of an algorithm, decidability, decidable languages, Undecidable languages, halting problem of TM, Post correspondence problem.

Complexity: Growth rate of functions, the classes of P and NP, Quantum Computation: quantum computers, Church-Turing thesis. **10 Hours**

Question paper pattern:

- The question paper will have TEN questions.
- There will be TWO questions from each module.
- Each question will have questions covering all the topics under a module.
- The students will have to answer FIVE full questions, selecting ONE full question from each module.

Fifth Semester B.E. Degree Examination
CBCS - Model Question Paper - 1
AUTOMATA THEORY AND COMPUTABILITY

Time: 3 hrs. Max. Marks: 80

Note : ***Answer any FIVE full questions, selecting ONE full question from each module.***

MODULE - 1

1. a. **Define the terms :**
   i. Kleene closure and kleene plus
   ii. Language
   iii. Alphabet
   iv. Finite automata **(08 Marks)**

Ans. **i. Kleene closure and kleene plus :**

The kleene closure Σ^* is defined as follows :

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup .....$

Which is the set of words of any length. Each string is made up of symbols only from Σ.

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

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

$\Sigma^* = \{\varepsilon, 0,1,00,01,10,11.....\}$

The kleene plus is a variation of kleene star operator. The kleene plus is denoted by Σ^+ is defined as follows :

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ....$

which is the set of words of any length except the null string.

Ex : $\Sigma^+ = \{0,1,00,01,10,11....\}$

ii. Language : A language can be defined as a set of strings obtained from Σ^* where Σ is set of alphabets of a particular language. Formally, a language 'L' over Σ is subset of Σ^* which is denoted by $L \subseteq \Sigma^*$

Ex : $L = \{\varepsilon,01,0011,000111,.....\}$

iii. Alphabet :

A language consist of various symbols from which the word s,statements can be obtained. These symbols are called alphabets.

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

$\Sigma = \{a,b,.....,z,A,B,.....,Z,0,1,.....9,.....\}$

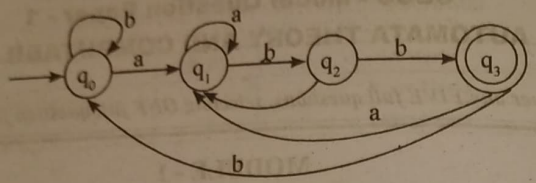
iv. Finite automata :

Finite automata (FA) are computing devices that accept/ recognize regular language. The FA acts as mathematical models and are used to study the abstract machines or abstract computing devices.

$L = \{W \in \Sigma^* \mid W$ is the language accepted by FA$\}$

b. Construct the DFA to accept string of a's and b's ending with the string abb, and also write transition table and transition function. (08 Marks)

Ans.



$\delta(q_0, a) = q_1$
$\delta(q_0, a) = q_0$
$\delta(q_1, a) = q_1$
$\delta(q_1, a) = q_2$
$\delta(q_2, a) = q_1$
$\delta(q_2, a) = q_3$
$\delta(q_3, a) = q_1$
$\delta(q_3, a) = q_0$

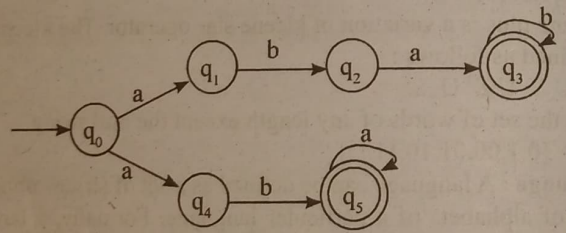
	δ	a	b
→	q_0	q_1	q_0
	q_1	q_1	q_2
	q_2	q_1	q_3
*	q_3	q_1	q_0

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
q_0 is start state
$F = \{q_3\}$
δ is transition function

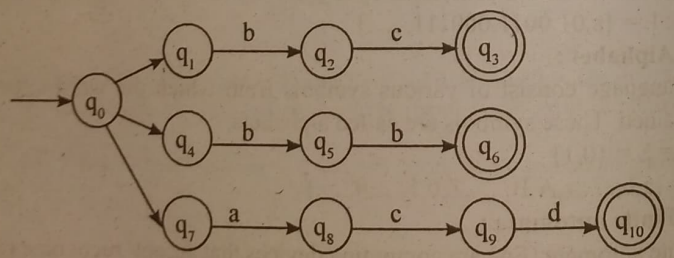
OR

2. a. Obtain the NFA to accept the language
i. $L = \{w \mid w \in abab^n$ or aba^n where $n \geq 0\}$
ii. $L = \{w \mid w \in 0101$ or 101 or $011\}$ (08 Marks)

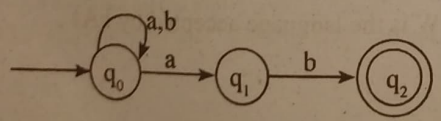
Ans. i.



ii.



b. Convert the following NFA to its equivalent DFA using subset construction method. (08 Marks)



Ans. Step 1 : q_0 is the start state
Step 2 : $\Sigma = \{a,b\}$
Step 3 : $Q_N = \{q_0, q_1, q_2\}$
$Q_D = \{\{\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0,q_1\}, \{q_0,q_2\}, \{q_1,q_2\}, \{q_0,q_1,q_2\}\}$
Step 4: Final states from the above subset
$F_D = \{\{q_2\}, \{q_0,q_2\}, \{q_1,q_2\}, \{q_0,q_1,q_2\}\}$
Step 5 : Identify transition function

For state ϕ :

Input symbol = a
$\delta_D(\phi,a) = \phi$

Input symbol = b
$\delta_D(\phi,b) = \phi$

For state q_0 :

Input symbol = a
$\delta_D(\{q_0\}, a) = \{q_0,q_1\}$

Input symbol = b
$\delta_D(\{q_0\},b\} = \{q_D\}$

For State q_1 :

Input symbol = a
$\delta_D(\{q_1\},a) = \phi$

Input symbol = b
$\delta_D(\{q_1\}, b) = \{q_2\}$

For State q_2 :

Input symbol = a
$\delta_D(\{q_2\},a) = \phi$

Input symbol = b
$\delta_D(\{q_2\},b) = \phi$

For state $\{q_0,q_1\}$:

Input symbol = a

$$\delta_D(\{q_0,q_1\},a) = \delta_N(\{q_0,q_1\},a)$$
$$= \delta_N(\{q_0\},a) \cup \delta_N(\{q_1\},a)$$
$$= \{q_0,q_1\} \cup \phi$$
$$= \{q_0,q_1\}$$

Input symbol = b

$$\delta_D(\{q_0,q_1\},b) = \delta_N(\{q_0,q_1\},b)$$
$$= \delta_N(\{q_0\},b) \cup \delta_N(\{q_1\},b)$$
$$= \{q_0\} \cup \{q_2\}$$
$$= \{q_0,q_2\}$$

For state $\{q_0,q_2\}$:

Input symbol = a

$$\delta_D(\{q_0,q_2\},a) = \delta_N(\{q_0,q_2\},a)$$
$$= \delta_N(\{q_0\},a) \cup \delta_N(\{q_2\},a)$$
$$= \{q_0,q_1\} \cup \phi$$
$$= \{q_0,q_1\}$$

Input symbol = b

$$\delta_D(\{q_0,q_2\},b) = \delta_N(\{q_0,q_2\},b)$$
$$= \delta_N(\{q_0\},b) \cup \delta_N(\{q_2\},b)$$
$$= \{q_0\} \cup \phi$$
$$= \{q_0\}$$

For state $\{q_1,q_2\}$:

Input symbol = a

$$\delta_D(\{q_1,q_2\},a) = \delta_N(\{q_1,q_2\},a)$$
$$= \delta_N(\{q_1\},a) \cup \delta_N(\{q_2\},d)$$
$$= \phi \cup \phi$$
$$= \phi$$

Input symbol = b

$$\delta_D(\{q_1,q_2\},b) = \delta_N(\{q_1,q_2\},b)$$
$$= \delta_N(\{q_1\},b) \cup \delta_N(\{q_2\},b)$$
$$= \{q_2\} \cup \phi$$
$$= \{q_2\}$$

For state $\{q_o,q_1,q_2\}$:

Input symbol = a

$\delta_D(\{q_o,q_1,q_2\},a) = \delta_N(\{q_o,q_1,q_2\},a)$

$= \delta_N(\{q_o\},a) \cup \delta_N(\{q_1\},b) \cup \delta_N(\{q_2\},a)$

$= \{q_o,q_1\} \cup \phi \cup \phi$

$= \{q_o,q_1\}$

Input symbol = b

$\delta_D(\{q_o\},b) \cup \delta_N(\{q_1\},b) \cup \delta_N(\{q_2\},b)$

$= \delta_N(\{q_o,q_1,q_2\},b) = \delta_N(\{q_o,q_1,q_2\},b)$

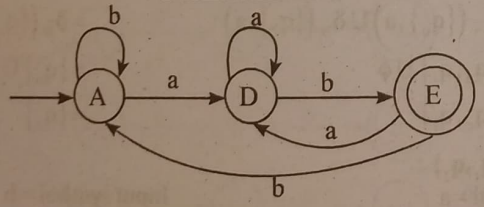
$= \{q_o\} \cup \phi \cup \{q_2\}$

$= \{q_o,q_2\}$

δ	a	b
ϕ(	ϕ	ϕ
$\{q_o\}$(A)	$\{q_o,q_1\}$	$\{q_o\}$
$\{q_1\}$(B)	ϕ	$\{q_2\}$
*$\{q_2\}$(C)	ϕ	ϕ
$\{q_o,q_1\}$(D)	$\{q_o,q_1\}$	$\{q_o,q_2\}$
*$\{q_o,q_1\}$(E)	$\{q_o,q_1\}$	$\{q_o\}$
*$\{q_1,q_2\}$(F)	ϕ	$\{q_2\}$
*$\{q_o,q_1,q_2\}$ (G)	$\{q_o q_1\}$	$\{q_o,q_2\}$

Even though we have 8 states observing from the table that 'A' is start state. The states reachable from A are A,D,E rest of the states are not reachable hence can be eliminated.

	δ	a	b
→	A	D	A
	D	D	E
*	E	D	A



Module - 2

3. a. What is regular expression ? Obtain a regular expression to a language consisting of strings of 0's and 1's with almost one point of consecutive 0's. (08 Marks)

Ans. A regular expression is recursively defined as follows :

1. ϕ is a regular expression denoting an empty language.
2. ε is a regular expression indicates the language containing an empty string.

3. a is a regular expression which denotes the language containing only {a}.
4. If R is a regular expression denoting the language f_R and S is a regular expression denoting the language L_s , then

a. R + S is a regular expression corresponding to the language $L_R \cup L_S$.

b. R - S is a regular expression corresponding to the language $L_R \cdot L_S$.

c. R* is a regular expression corresponding to the language L_R.

5. The expression obtained by applying any of the rules from 1 to 4 are regular expressions.

R.E = 1*+(1+01)* 0(1+01)* +(1+01)* 00(1+01)*

b. State and prove pumping lemma for regular languages. (08 Marks)

Ans. Let M = (Q,Σ,δ,q_o,F) be an FA and has 'η' number of states. Let L be the regular language accepted by M. Let frequency string x can be broken into three substrings u,v and w such that

$$X = uvw$$

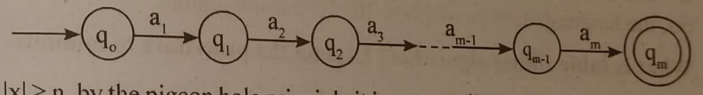
Satisfying the following constraints :

$V \neq \varepsilon$ ie., $|V| \geq 1$

$|uv| \leq \eta$

Then $uv^i w$ is in L for $i \geq 0$

Let $X = a_1a_2a_3 \ldots a_m$ where $m \geq n$ and each a_2 is in Σ. Here , n represent the states of DFA. Since we have m input symbols, naturally we should have m+1 states in the sequence $q_o,q_1,q_2,\ldots q_m$, where q_o will be the start state and q_m will be the final state as shown below.

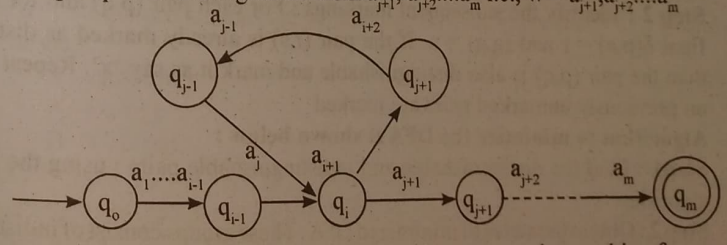


Since $|x| \geq n$, by the pigeon hole principle it is not possible to have distinct transitions. Once of the state can have a loop. Let the string x is divided into three substrings as shown below.

The first group is the string prefix from $a_1a_2\ldots a_i$ i.e., $u = a_1a_2\ldots a_i$

The second group is the loop string from $a_{i+1},a_{i+2}\ldots a_{j-1}$ i.e., $V = a_{i+1}, a_{i+2}\ldots a_{j-1}a_j$

The third group is the string suffix from $a_{j+1}, a_{j+2}\ldots a_m$ i.e., $w = a_{j+1},a_{j+2}\ldots a_m$



Observe from above figure that, the prefix string u taxes the machine from q_o to q_i , the loop string v takes the machine from q_i to q_i (Note $q_i = q_j$) and suffix string w toches the machine from q_j to q_f . The minimum string that can be accepted by the above FA is uw with i = 0.

But, when i = 1, the string uvw is accepted by DFA when i = 2, it is accepted. So if i > 0 the machine goes from qo to qj on input string u, circles from qi to qi based on the value of i and then goes to accepting state on input string w. The machine will be in state qi. Since qi and qj are same we can input the string $a_{i+1}\ a_{i+2}$ ....a_j any number of times and the machine will stay in qj only. Finally, if the input string is $a_{j+1}\ a_{j+2}$,.... a_m the machine enters into final state q_m.

4. a. Show that L = {ww^R} w∈(0+1)*} is not regular. **(08 Marks)**

Ans. Let L is regular and n be the number of states in FA. Consider the string

$$X = \overbrace{1.....10.....0}^{w}\overbrace{0.....01.....1}^{R}$$
$$\underbrace{\quad}_{n}\ \underbrace{\quad}_{n}\ \underbrace{\quad}_{n}\ \underbrace{\quad}_{n}$$

Where 'n' is the number of the states of FA, w = 1....10....0 and reverse of w is given by w^R = 0....01.....1

Since |x| ≥ n we can split the string x into uvw such that |uv| ≤ n and |V| ≥ 1 as shown below

$$X = 1....\underbrace{1}_{u}\ \underbrace{0....00.....01}_{v}\ \underbrace{.....1}_{w}$$

Where |u| = n-1 and |v| = 1 so that |uv| = |u| + |v| = η -1 + 1 = η which is true according to pumping, lemma, uvw ∈ L for i = 0,1,2.....

If i is 0 i.e., v does not appear and so the number of is on the left of x will be less than the number of is on the right of x and so the string is not of the form ww^R. So uvw L when i = 0. This is a contradiction to the assumption that the language is regular. So, the language L = {ww^R | w ∈ {0+1}*} is not regular.

b. What is table filling algorithm? Explain the procedure to minimize DFA. **(08 Marks)**

Ans. The table filling algorithm is used to find the set of states that are distinguishable and indistinguishable states. The algorithm is recursively defined as shown below

Step 1 : Identify the initial markings : for each pair (p,q) where p ∈ Q and q ∈ Q, if p ∈ F and p ∈ F or vice versa then, the pair (p,q) is indistinguishable and mark the pair (p,q).

Step 2 : Identify the subsequent markings : For each pair (p,q) and for each a ∈ Σ, final δ(p,a) = r and (q,a) = a. If the pair (r,s) is already marked as distinguishable than the pair (p,q) is also distinguishable and mark it as say 'x'. Repeat step 2 until no previously unmarked pairs are marked

Algorithm to minimize the DFA is shown below :

Step1 : Find the distinguishable and indistinguishable pairs : using the table filling algorithm.

Step 2 : Obtain the states of minimized DFA : These groups consist of indistinguishable pairs obtained in previous step and individual distinguishable pairs obtained in previous step and individual distinguishable states. The groups obtained are the states of minimized DFA.

Step 3 : Compute the transition table if $[p_1,p_2,....p_k]$ is a group and if δ($[p_1,p_2,....p_k]$,

a) = $[r_1,r_2,...r_m]$ and label the edge with the symbola. Follow this procedure for each group obtained in step 2 and for each at Σ.

Step 4 : Identify the start state : if one of the component in the group $[p_1,p_2,...p_k]$ consist of a start state of given DFA then $[p_1, p_2, ...p_k]$ is the star state of minimized DFA.

Step 5 : Identify the final state if the group $[p_1, p_2, ...p_k]$ contains a final state of given of A then the group $[p_1, p_2, ...p_k]$ is final state of minimized DFA.

Module - 3

5. a. Explain the classification of grammars with example. **(08 Marks)**

Ans.

Types of grammars
- Type0 grammar (phrase structured grammer)
- Type1 grammar (Context sensitive grammer)
- Type2 grammar (Context free grammer)
- Type3 grammar (Regular grammer)

Type 0 : A grammar G = (V,T,P,S) is said to be type 0 grammer or unrestricted grammer or propose structured grammer if all the production are of the of the form α – β where α∈(VUT)⁺ αvδ β∈(V U T)*

Ex : S → a Ab | ε
aA → bAA
bA → a

Type 1 : A grammer G = (V,T,P,S) is said to be type 1 grammar or context sensitive grammer if all the productions are of the form α → β as in type grammar. But, there is a restriction on length of β. The length of β must be at least as much as the length of α ie., |β| ≥ |α| and α and β∈ (V U T)⁺ i.e., ε can not appear on the left hand or right hand side of any production.

Ex : S → a Ab
aA → bAA
bA → aa

Type 2 : A grammer G = (V,T,P,S) is said to be type 2 grammar or context free grammar if all the productions are of the form A → α where α∈(V U T)* and A is non-terminal.

Ex : S → aB |bA| ε
A → aA | b
B → bB |a|ε

Type 3 : The grammer G = (V,T,P,S) is said to be type 3 grammar or regular grammar if the grammar is right linear or left linear. A grammer g is said to be right linear if all production are the form.

A → wB or A → w

If it is left linear, all production of the form

A → Bw or A → w

Ex : S → aaB | bbA| ε

$A \rightarrow aA \mid b$
$B \rightarrow bB \mid a \mid \varepsilon$

b. Obtain a grammar to generate the following language **(08 Marks)**
L= {wwR | w∈ {a,b}*}

Ans. The language can be written as
L = {aa,bb,abba,baab,....}
Observe that the given string is a palindrome of even length. This achieved by deleting the production S → a/b. So the final grammar is given by

$S \rightarrow \varepsilon$
$S \rightarrow a\,Sa \mid bSb$

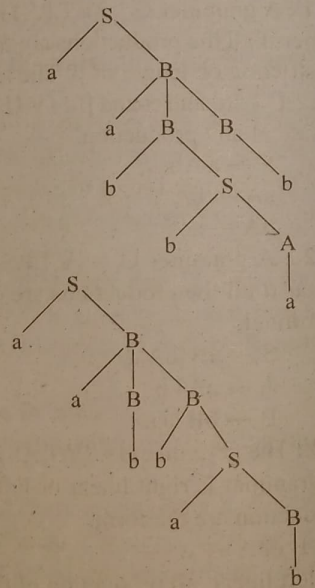
G = (V,T,P,S) can be defined as
V = {S}
T = {a,b}
P = {
$S \rightarrow asa \mid bsb$
$S \rightarrow \varepsilon$
}
S is start symbol.

OR

6. a. Is the following grammar ambiguous? **(08 Marks)**
S → aB | bA
A → as | bAA|a
B → bS|aBB|b

Ans. Left most derivation is
S ⇒ aB
⇒ aaBB
⇒ aabSB
⇒ aabbAB
⇒ aabbaB
⇒ aabbab

Left most derivation
S ⇒ aB
⇒ aaBB
⇒ aabB
⇒ aabbS
⇒ aabbaB
⇒ aabbab



For string aabbab, more than one parse tree is available , so the given grammar is ambiguous.

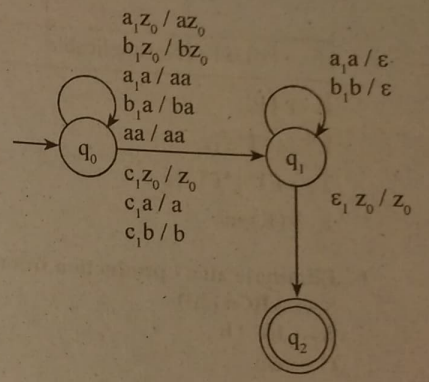
b. Obtain a PDA to accept the language L(m) = {wCw^R | w∈(a+b)*}. **(08 Marks)**

Ans. M = (Q,Σ, Γ,δ,Q_0,Z_0,f]
Where $Q = \{q_o, q_1, q_2\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b, Z_o\}$
δ:{

$\delta(q_o, a, z_o) = (q_o, aZ_o)$
$\delta(q_o, b, Z_o) = (q_o, b, Z_o)$
$\delta(q_o, a, a) = (q_o, aa)$
$\delta(q_o, b, a) = (q_o, ba)$
$\delta(q_o, a, b) = (q_o, ab)$
$\delta(q_o, c, Z_o) = (q_1 Z_o)$
$\delta(q_o, c, a) = (q_1, a)$
$\delta(q_o, c, a) = (q_1, a)$
$\delta(q_1, a, a) = (q_1, \varepsilon)$
$\delta(q_1, b, b) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon_1 Z_o) = (q_1 Z_o)$

}

$q_0 \in Q$ is start state
$z_0 \in \Gamma$ is the initial stack symbol
$F = \{q_2\}$ is final state



Module - 4

7. a. What is left recursion? Eliminate a left recursion from the following grammar.
E → E + T | T
T → T * F|F
F → (E) | id **(08 Marks)**

Ans. A grammer G is said to be left recursive if there is some non terminal A such that $A \overset{+}{\Rightarrow} A\alpha$
The left recursion in a grammer G can be eliminated as shown below
$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \ldots\ldots A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m$
Where β_i do not start with A
$A \rightarrow \beta_1 A^1 \mid \beta_2 A^1 \mid \beta_3 A^1 \mid \ldots\ldots \beta_m A \mid$
$A^1 \rightarrow \alpha_1 A^1 \mid \alpha_2 A^1 \mid \alpha_3 A^1 \mid \ldots\ldots \mid \alpha_n A^1 \mid \varepsilon$

$A \to A\alpha_1/\beta_1$	Sbstitution	Without left recursion
$\epsilon \to \epsilon + T/T$	$A = \epsilon$ $\alpha_1 = +T$ $\beta_1 = T$	$\epsilon \to TE^1$ $E^1 \to +T\epsilon^1/\varepsilon$
$T \to T*F/F$	$A = T$ $\alpha_1 = *F$ $\beta_1 = F$	$T \to FT^1$ $T^1 \to *FT^1/\varepsilon$
$F \to (\epsilon)/id$	Not applicable	$F \to (\epsilon)\|id$

$E \to TE^1$

$E^1 \to +T\epsilon^1 | \varepsilon$

$T \to FT^1 | *FT^1 | \varepsilon$

$F \to (E) | id$

b. Eliminate all ε - production from the grammer. **(08 Marks)**

$S \to ABCa \mid bD$

$A \to BC \mid b$

$C \to c/\varepsilon$

$D \to d$

Ans. Step 1 : Obtain set of nulable variables from the grammer

ov	nv	Production
ϕ	B,C	$B \to \varepsilon$ $C \to \varepsilon$
B,C	B,C,A	$A \to BC$
B,C,A	B,C,A	– –

V = {B,C,A} are nullable variables

Step 2 : Construction of production p1

Productions	Resulting productions (p)
$S \to ABCa$	$S \to ABC_a \mid BCa \mid ACa \mid ABa \mid Ca \mid Aa \mid Ba \mid a$
$S \to bB$	$S \to bD$
$A \to BC \mid b$	$A \to BC \mid B \mid C \mid b$
$B \to b \mid \varepsilon$	$B \to b$
$C \to C/\varepsilon$	$C \to C$
$D \to d$	$D \to d$

$G^1 = (V^1, T^1, p^1, S)$

$V^1 = \{S, A, B, C, D\}$

$T^1 = \{a, b, c, d\}$

$p^1 = \{$

$S \to ABCa \mid BCa \mid ACa \mid ABa \mid Ca \mid Aa \mid Ba \mid a \mid bD$

$A \to BC \mid CC \mid b$

$B \to b, C \to c, D \to d$

$\}$

S is start symbol.

OR

8. a. Explain two forms of normal forms. **(08 Marks)**

Ans. There are two different types of normal forms.

1. Chomsky normal form (CNF)
2. Greibach normal form (GNF)

1. Chomsky Normal Form (CNF) : Let G = (V,T,P,S) be a CFG. The grammar G is said to be in CNF if all production are of the form.

$A \to BC$ or $A \to a$

Where A,B and C $\in$ V and a $\in$ T

Note that if a grammar is in CNF, the right hand side of the production should contain two symbols or one symbol. If there are two symbols on the right hand side those two symbols must be non - terminals and if there is only one symbol, that symbol must be a terminal.

2. Greibach normal form (GNF) : In CNF there is restriction on the number of symbols on the right hand side of teh production. Note that in CNF not more than two symbols on RHS of the production are permitted. If there is only symbol that symbol must be a terminal and if there are two symbols, those symbols must be variables.

In GNF there is no restriction the number of symbols on the right hand side but there is restriction on the terminals and variables appear on the right hand side of the production.

G = (V,T,P,S). The 'G; is said to be in GNF if all the productions are of the form A $\to a\alpha$.

Where a $\in$ T and $\alpha \in V^*$ i.e., the first symbol on the right hand side of the production must be terminal and it can be followed by zero or more variables.

b. Consider the grammar

$S \to 0A \mid 1B$

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

$B \to 1BB \mid OS \mid 0$

Obtain the grammar in CNF **(08 Marks)**

Ans.

Given productions	Action	Resulting production
$S \to 0A \mid 1B$	Replace 0 by B_0 and introduce production $B_0 \to 1$ Replace 0 by B_1 and introduce the production $B_1 \to 1$	$S \to B_0A \mid B_1B$ $B_0 \to 0$ $B_2 \to 1$
$A \to 0AA / 1S$	Replace 0 by B_0 and introduce $B_0 \to 0$ Replace 1 by B_1 and introduce $B_1 \to 1$	$A \to B_0AA \mid B_1S$ $B_0 \to 0$ $B_1 \to 1$
$B \to 1BB/0S$	Replace 0 by B_0 and introduce $B_0 \to 0$ Replace 1 by B_1 and introduce $B_1 \to 1$	$B \to B_1BB / B_0S$ $B_1 \to 1$ $B_0 \to 0$

$G=(V_1,T_1,P_1,S)$

$V_1=\{S,A,B,B_o,B_1\}$

$T_1=\{0,1\}$

$P_1=\{$

$S \to B_oA \mid B_1B$

$A \to B_oAA \mid B_1S \mid 1$

$B \to B_1BB \mid B_oS \mid 0$

$B_o \to 0$

$B_1 \to 1$

$\}$

S is the start symbol

Finally

$G^1=(V^1,T^1,P^1,S)$ in CNF

$V_1=\{S,A,B,B_o,B_1,D_1,D_2\}$

$T_1=\{0,1\}$

$P_1=\{$

$S \to B_oA \mid B_1B$

$A \to B_1S \mid 1 \mid B_oD_1$

$B \to B_oS \mid 0 \mid B_1D_2$

$B_o \to 0$

$B_1 \to 1$

$D_1 \to AA$

$D_2 \to BB$

$\}$

S is the start symbol

Module - 5

9. a. Obtain a turning machine to accept the language **(08 Marks)**

$L = \{0^n 1^n \mid n \geq 1\}$

Ans. $M=(Q,\Sigma,\Gamma,\delta,q_o,B,F)$

Where $Q=\{q_o,q_1,q_2,q_3\}$

$\Sigma=\{o_1\}$

$\Gamma=\{0,1,x,y,B\}$

$q_o \in Q$ is the start state of machine

$B \in \Gamma$ is the blank symbol

$F=\{q_4\}$ is the final state

δ is shown below:

$\delta(q_o,0)=(q_1,X,R)$

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

$\delta(q_1,y)=(q_1,y,R)$

$\delta(q_1,1)=(q_2,Y,L)$

$\delta(q_2,Y)=(q_2Y,L)$

$\delta(q_2,0)=(q_2,0,L)$

$\delta(q_2,X)=(q_o,X,R)$

$\delta(q_o,Y)=(q_3,Y,R)$

$\delta(q_3,Y)=(q_3,Y,R)$

$\delta(q_3,B)=(q_4,B,R)$

S	Tape symbols (Γ)				
States	0	1	X	Y	B
q_o	(q,X,R)	–		(q_3,Y,R)	
q_1	$(q_1,0,R)$	(q_2,Y,L)		(q_1,Y,R)	
q_2	(q_2,o,L)	–	(q_o,X,R)	(q_2,Y,L)	
q_3				(q_3,Y,L)	(q_4,B,R)
q_4					

b. Explain

I. turing machine as multi-stack machines

II. semi-infinite tape **(08 Marks)**

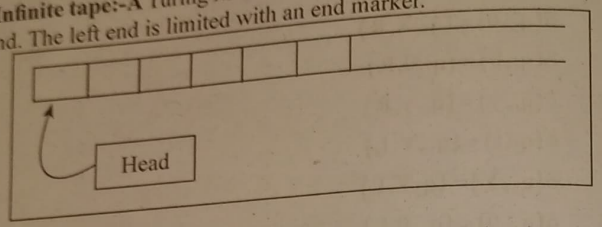
Ans. I. **Turing machine as multi-stack machines:-** Multi-stack/track Turing machines, a specific type of Multi-tape Turing machine, contain multiple tracks but just one tape head reads and writes on all tracks. Here, a single tape head reads n symbols from **n** tracks at one step. It accepts recursively enumerable languages like a normal single-track single-tape Turing Machine accepts.

A Multi-track Turing machine can be formally described as a 6-tuple $(Q, X, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- X is the tape alphabet
- Σ is ^me input alphabet
- δ is a relation on states and symbols where $\delta(Qi, [a_h\ a_2, a_3,....]) = (Q_j\ [b_1\ b_2, b_3,....]$, Leftshift or Rightshift)
- q_o is the initial state
- F is the set of final states

Note - For every single-track Turing Machine S, there is an equivalent multi-track Turing Machine M such that **L(S) = L(M).**

II. **Semi- Infinite tape:-A** Turing Machine with a semi-infinite tape has a left end but no right end. The left end is limited with an end marker.



It is a two-track tape -

Upper track - It represents the cells to the right of the initial head position.

Lower track - It represents the cells to the left of the initial head position in reverse order.

The infinite length input string is initially written on the tape in contiguous tape cells. The machine starts from the initial state qo and the head scans from the left end marker 'End'.

In each step, it reads the symbol on the tape under its head. It writes a new symbol on that tape cell and then it moves the head either into left or right one tape cell. A transition function determines the actions to be taken.

It has two special states called **accept state** and **reject state.** If at any point of time it enters into the accepted state, the input is accepted and if it enters into the reject state, the input is rejected by the TM. In some cases, it continues to run infinitely without being accepted or rejected for some certain input symbols.

Note - Turing machines with semi-infinite tape are equivalent to standard Turing machines.

OR

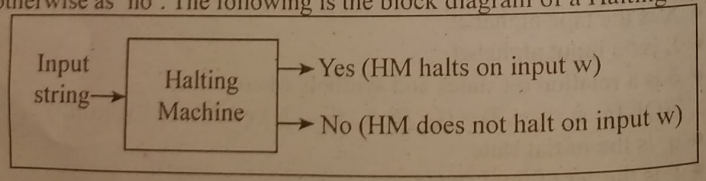
10. a. Explain Halting Problem **(08 marks)**

Ans. Halting Problem:

Input - A Turing machine and an input string w.

Problem - Does the Turing machine finish computing of the string win a finite number of steps? The answer must be either yes or no.

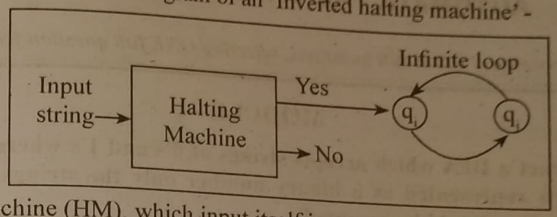
Proof - At first, we will assume that such a Turing machine exists to solve this problem and then we will show it is contradicting itself. We will call this Turing machine as a **Halting machine** that produces a 'yes' or 'no' in a finite amount of time. If the halting machine finishes in a finite amount of time, the output comes as 'yes', otherwise as 'no'. The following is the block diagram of a Halting machine -



Now we will design an **inverted halting machine (HM)'** as -

- If **H** returns YES, then loop forever.
- If H returns NO, then halt.

The following is the block diagram of an 'Inverted halting machine' -



Further, a machine $(HM)_2$ which input itself is constructed as follows -

- If $(HM)_2$ halts on input, loop forever.
- Else, halt.

Here, we have got a contradiction. Hence, the halting problem is **undecidable.**

b. Explain Linear bounded automata with respect to hiring machines (08 Marks)

Ans. A linear bounded automaton is a multi-track non-deterministic Turing machine with a tape of some bounded finite length.

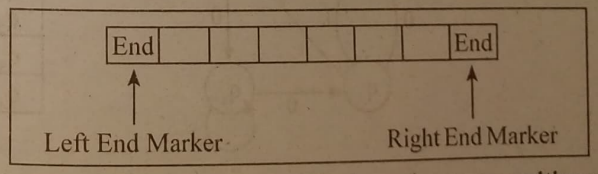
Length = function (Length of the initial input string, constant c) Here,

Memory information ≤ c * Input information

The computation is restricted to the constant bounded area. The input alphabet contains two special symbols which serve as left end markers and right end markers which mean the transitions neither move to the left of the left end marker nor to the right of the right end marker of the tape.

A linear bounded automaton can be defined as an 8-tuple (Q, X, X, Qo, ML, MR, δ, F) where -

- Q is a finite set of states
- X is the tape alphabet
- Σ is the input alphabet
- q_0 is the initial state
- M_L is the left end marker
- M_R is the right end marker where $M_R \wedge M_L$
- δ is a transition function which maps each pair (state, tape symbol) to (state, tape symbol, Constant 'c') where c can be 0 or +1 or -1
- F is the set of final states



A deterministic linear bounded automaton is always **context-sensitive** and the linear bounded automaton with empty language is **undecidable.**

Fifth Semester B.E. Degree Examination

CBCS - Model Question Paper - 2

AUTOMATA THEORY AND COMPUTABILITY

Time: 3 hrs. Max. Marks: 80

Note : *Answer any FIVE full questions, selecting ONE full question from each module.*

MODULE - 1

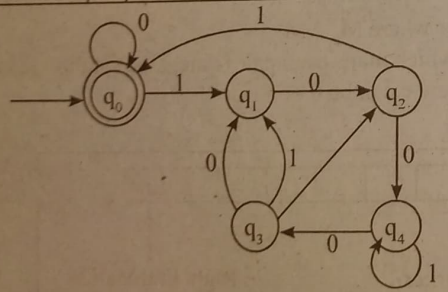
1. a. Construct a DFA which accepts strings of 0's and 1's where the value of each string is represented as a binary number only the strings representing zero modulo five should be accepted. **(08 Marks)**

Ans.

$\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$ with $r = 2$ and $k = 5$

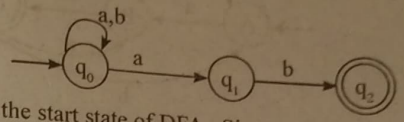
So, $j = (2i + d) \bmod 5$

Remainder	d	$(2*i+d) \bmod 5 = j$	$\delta(q_i, d) = q_j$
i = 0	0	$(2*0+0) \bmod 5 = 0$	$\delta(q_0, 0) = q_0$
	1	$(2*0+1) \bmod 5 = 1$	$\delta(q_0, 1) = q_1$
i = 1	0	$(2*1+0) \bmod 5 = 2$	$\delta(q_1, 0) = q_2$
	1	$(2*1+1) \bmod 5 = 3$	$\delta(q_1, 1) = q_3$
i = 2	0	$(2*2+0) \bmod 5 = 4$	$\delta(q_2, 0) = q_4$
	1	$(2*2+1) \bmod 5 = 0$	$\delta(q_2, 1) = q_0$
i = 3	0	$(2*3+0) \bmod 5 = 1$	$\delta(q_3, 0) = q_1$
	1	$(2*3+1) \bmod 5 = 2$	$\delta(q_3, 1) = q_2$
i = 4	0	$(2*4+0) \bmod 5 = 3$	$\delta(q_4, 0) = q_3$
	1	$(2*4+1) \bmod 5 = 4$	$\delta(q_4, 1) = q_4$



δ	0	1
$\rightarrow *q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

b. Convert the following NFA to its equivalent DFA using lazy evaluation **(08 Marks)**



Ans. **Step 1 :** Identifying the start state of DFA : Since q_0 is the start state of NFA, $[q_0]$ is the state of DFA.

Step 2 : Identify the alphabets of DFA : The input alphabets of NFA are the input alphabets of DFA. So $\Sigma = \{a,b\}$

Step 3 : Identify 2D which are the states of DFA : Start from the start state q_0 and find the transition as shown below :

For state q_0 :

Input symbol = a

$\delta_D(\{q_0\}, a) = \{q_0, q_1\}$

Input symbol = b

$\delta_D(\{q_0\}, b) = \{q_0\}$

For state $\{q_0, q_1\}$:

Input symbol = a

$\delta_D(\{q_0, q_1\} a) = \delta_N(\{q_0, q_1\}, a)$

$= \delta_N(\{q_0\}, a) \cup \delta_N(\{q_1\}, a)$

$= \{q_0, q_1\} \cup \phi$

$= \{q_0, q_1\}$

Input symbol = b

$\delta_D(\{q_0, q_1\}, b) = \delta_N(\{q_0, q_1\}, b)$

$\delta_D(\{q_0\}, b) \cup \delta_N(\{q_1\}, b)$

$= \{q_0\} \cup \{q_2\}$

$= \{q_0, q_2\}$

For state $\{q_0, q_2\}$:

Input symbol = a

$\delta_D(\{q_0, q_2\}, a) = \delta_N(\{q_0, q_2\}, a)$

$= \delta_N(\{q_0\}, a) \cup \delta_N(\{q_2\}, a)$

$= \{q_0, q_1\} \cup \phi$

$= \{q_0, q_1\}$

Input symbol = b

$\delta_D(\{q_0, q_2\}, b) = \delta_N(\{q_0, q_2\}, b)$

$= \delta_N(\{q_0\}, b) \cup \delta_N(\{q_2\}, b)$

$= \{q_0\} \cup \phi$

$= \{q_0\}$

Since, no new state is generated the procedure is terminated.

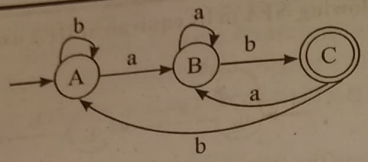
Step 4 : Identify the final states of DFA : Since q_2 is the final state of NFA in the above set, wherever q_2 is present as an element, the corresponding set is the final state of DFA so

$F_D = \{\{q_0, q_2\}\}$

	δ	a	b
$\rightarrow$	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
*	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

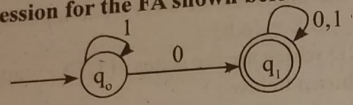
$\Rightarrow$ By renaming

	δ	a	b
$\rightarrow$	A	B	A
	B	B	C
*	C	B	A



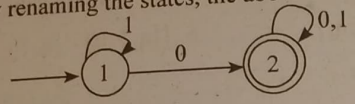
OR

2. a. Obtain a regular expression for the FA shown below :



What is the language corresponding to the regular expression. **(08 Marks)**

Ans. Let $q_0 = 1$ and $q_1 = 2$ by renaming the states, the above FA can be written as



Basis :

When k = 0

$R_{11}^{(0)} = \varepsilon + 1$

$R_{12}^{(0)} = 0$

$R_{21}^{(0)} = \phi$

$R_{22}^{(0)} = \varepsilon + 0 + 1$

Induction : The regular expression corresponding to the path from state i to state j through a state which is not higher than k is given by

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}\left[R_{kk}^{(k-1)}\right]^* R_{kj}^{(k-1)}$$

When k = 1

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)}\left[R_{11}^{(0)}\right]^* R_{11}^{(0)}$$

$$= (\varepsilon + 1) + (\varepsilon + 1)(\varepsilon + 1)^* (\varepsilon + 1)$$

$$= 1^*$$

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)}\left[R_{11}^{(0)}\right]^* R_{12}^{(0)}$$

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

$$= 0 + 1^* 0$$

$$= 1^* 0$$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}\left[R_{11}^{(0)}\right]^* R_{11}^{(0)}$$

$$= \phi + \phi(\varepsilon + 1)^* (\varepsilon + 1)^* 0$$

$$= \phi$$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}\left[R_{11}^{(0)}\right]^* R_{12}^{(0)}$$

$$= (\varepsilon + 0 + 1) + \phi(\varepsilon + 1)^* 0$$

$$= (\varepsilon + 0 + 1)$$

When K = 2

$$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)}\left[R_{22}^{(1)}\right]^* R_{21}^{(1)}$$

$$= 1^* + 1^* 0(\varepsilon + 0 + 1)^* \phi$$

$$= 1^*$$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}\left[R_{22}^{(1)}\right]^* R_{22}^{(1)}$$

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

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

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

$$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)}\left[R_{22}^{(1)}\right]^* R_{21}^{(1)}$$

$$= \phi(\varepsilon + 0 + 1)(\varepsilon + 0 + 1)^* \phi$$

$$= \phi$$

$$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)}\left[R_{22}^{(1)}\right]^* R_{22}^{(1)}$$

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

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

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

So, $R_{12}^2 = 1^* 0(0 + 1)^*$

So, the regular expression for the given DFA is $1^*0(0+1)^*$ which is the language consisting of any number of is followed by a zero and then followed by string of 0's and 1's.

b. List and explain applications of Regular Expressions. **(08 Marks)**

Ans. 1) Regular expression in unix
2) Pattern Matching
3) Lexical Analysis
4) Unix editor

1) Regular expression in unix : Regular expressions are extensively used in UNIX operating system. But certain short hand notations are used in UNIX platform using which complex regular expressions are avoided. For example, the symbol '.' stands for any character, the sequence [bcde...] stands for regular expression "a + b + c + d...", the operator 1 is used in place of +, the operator? means "Zero or one of "etc most of the commands are in voked invariably uses regular expression. For example grep (Global Regular Expression and Print) used to search for a pattern of string.

2) Pattern Matching : Refers to set of objects with some common properties. We can match an identifier or a decimal number or we can search for a string in the text.

3) Lexical Analysis : Regular expressions are extensively used in the design of lexical analyzer phase. This phase scans the source program and recognizer the tokens which are logically together. The UNIX commands such as lex accepts regular expressions as the input and produces the lexical analyzer generator. This generator takes a high level description of a lexical analyzer as the input and produces lexical analyzer.

4) Unix editor : In UNIX operating system, we can use the editor ed to search for a specific pattern in the text. For example, if the command specified is /abc*c/ then the editor searches for a string which starts with ab followed by zero or more c's and followed by the symbol C.

Module - 2

3. a. Show that L = $\{a^i b^j \mid i>j\}$ is not regular. **(08 Marks)**

Ans. Step 1 : Let L is regular and n be the number of states in FA.

Consider the string $x = a^{n+1}b^n$

Step 2 : Since $|x| = 2n + 1 \geq n$, we can split x into uvw such that $|uv| \leq n$ and $|v| \geq 1$ as shown below.

$$X = a^{n+1}b^n = \underset{u}{a^j}\,\underset{v}{a^k}\,\underset{w}{ab^n}$$

Where $|u| = j$ and $|v| = k \geq 1$ and so that $|uv| = |u| + |v| = j + j \leq n$

Step 3 : According to pumping lemma, uv w $\in$ L for $i \geq 0$

i..e, $a^j (a^k)^i ab^n \hat{I} L$ for $i^3 0$

Now, if we choose i = 0, number of a's in string u will bot be less than the number of b's in w which is contradiction to the assumption that number of a's are more than the number of b's.

So, the language L = $\{a^i b^j \mid i>j|\}$ is not regular

b. Show that if L_1 and L_2 are regular, then $L_1 \cup L_2$, $L_1 \bullet L_2$ and L_1^* are also regular. **(08 Marks)**

Ans. Theory : If L_1 and L_2 are regular, then $L_1 \cup L_2$, $L_1 \bullet L_2$ and L_1^* also denote the regular languages.

Proof : It is given that L_1 and L_2 are regular languages , so there exists regular expression R_1 and R_2 such that

$L_1 = L(R_1)$

$L_2 = L(R_2)$

By the definition of regular expressions, we have

1. $R_1 + R_2$ is a regular expression denoting the language $L_1 \cup L_2$.
2. $R_1 . R_2$ is a regular expression denoting the language $L_1 . L_2$
3. R_1^* is a regular expression denoting the language L_1^*

So the regular languages are closed under union, concatenation and star operations.

OR

4. a. Obtain the minimized DFA for the following. **(10 Marks)**

δ	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Ans. Vertical and horizontal marking

	B						
	C						
*	D	X	X	X			
	E				X		
	F				X		
	G				X		
	H				X		
		A	B	C	D	E	F	G

*

δ	a	b
(A,B)	(A,B)	(A,C)
(A,C)	(B,D)	(A,B)
(A,E)	(B,D)	(A,F)
(A,F)	(B,G)	(A,E)
(A,G)	(B,F)	(A,G)
(A,H)	(B,G)	(A,D)
(B,C)	(A,D)	(C,B)
(B,E)	(A,D)	(C,F)
(B,F)	(A,G)	(C,E)
(B,G)	(A,F)	(C,G)
(B,H)	(A,G)	(C,D)
(C,E)	(D,D)	(B,F)
(C,F)	(D,G)	(B,E)
(C,G)	(D,F)	(B,G)
(C,H)	(D,G)	(B,D)
(E,F)	(D,G)	(F,E)
(E,G)	(D,F)	(F,G)
(E,H)	(D,G)	(F,D)
(F,G)	(G,F)	(E,G)
(E,H)	(G,G)	(E,D)
(G,H)	(F,G)	(D,G)

	B							
	C	X	X					
*	D	X	X	X				
	E	X	X		X			
	F			X	X	X		
	G			X	X	X		
	H	X	X	X	X	X	X	X
		A	B	C	D	E	F	G

*

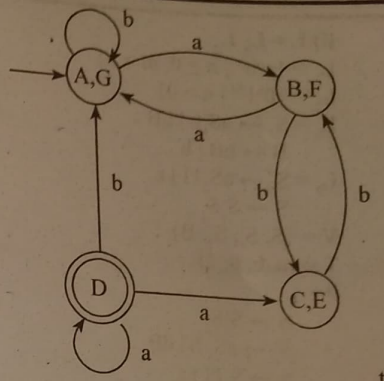
δ	a	b
(A,B)	(B,A)	(A,C)
(A,F)	(B,G)	(A,E)
(A,G)	(B,F)	(A,G)
(B,F)	(A,G)	(C,E)
(B,G)	(A,F)	(C,G)
(C,E)	(D,D)	(B,F)
(E,G)	(G,F)	(E,G)

	B	*						
	C	X	X					
*	D	X	X	X				
	E	X	X		X			
	F	X		X	X	X		
	G		X	X	X	X		
	H	X	X	X	X	X	X	X
		A	B	C	D	E	F	G

*

(A,G)(B,F) & (C,E) are indistinguishable
(D,H) are distinguishable

δ	a	b
→(A,G)	(B,F)	(A,G)
(B,F)	(A,G)	(C,E)
(C,E)	D	(B,F)
*D	D	(A,G)
H	(A,G)	D



t

b. What are the limitations of finite automation. **(06 Marks)**

Ans. 1. An FA has finite number of states and so it does not have the capacity to remember arbitrary long amount of information.

2. Since it does not have memory, FA cannot remember a long string. For example, to check for matching parenthesis, check whether the string is a palindrone or not etc, are not possible using FA.

3. Finite autoyata or finite state machine have troube recognizing various types of languages involving counting, calculating, storing the string.

Module - 3

5. a. Obtain a grammer to generate the language **(10 Marks)**

i. $L=\{q^n b^{n-3} \mid \eta \geq 3\}$

ii. $L=L_1L_2$ where $L_1=\{a^n b^m \mid n \geq 0, m > n\}$ $L_2=\{0^n, {}^{2n} \mid n \geq 0\}$

Ans. i. $L = \{a^n b^{n-3} \mid n \geq 3\}$

It is clear from the above statement that the set of strings that can be generated by this language can be represented as

L = {aaa, aaaab, aaaaabb, ...}

$S \rightarrow aaaA$

$A \rightarrow aAb \mid \varepsilon$

So the final grammar that can be generated is

V = {S,A}

T = {a,b}

P = {

$S \rightarrow aaA$

$A \rightarrow aAb|\varepsilon$

}

S - is start symbol

ii) $L = L_1 L_2$

$L_1 = \{a^n b^m \mid n \geq 0, m > n\}$

$L_2 = \{0^n 1^{2n} \mid n \geq 0\}$

$L_1 = S_1 \rightarrow aS_1 b \mid bB$

$B \rightarrow bB \mid b$

$L_2 = S_2 \rightarrow aS_2 11 \mid \varepsilon$

$S \rightarrow S_1 S_2$

$V = \{S, S_1, S_2, B\}$

$T = \{a, b, 0, 1\}$

$P = \{$

$S \rightarrow S_1 S_2$

$S_1 \rightarrow aS_1 b \mid dB$

$S \rightarrow S_2 11 \mid \varepsilon$

$B \rightarrow bB \mid b$

$\}$

S_1 is start symbol.

b. Obtain the string ibtib taxa from the grammar shown below and verify whether the grammar is ambiguous or not **(06 Marks)**

$S \rightarrow iCts \mid ictSeS \mid a$

$C \rightarrow b$

Ans. $S \Rightarrow iCtS \Rightarrow ibtS$

$\Rightarrow ibtiCtSeS$

$\Rightarrow ibtibtSeS$

$\Rightarrow ibtibtaeS$

$\Rightarrow ibtibtaea$

$\Rightarrow iCtSeS$

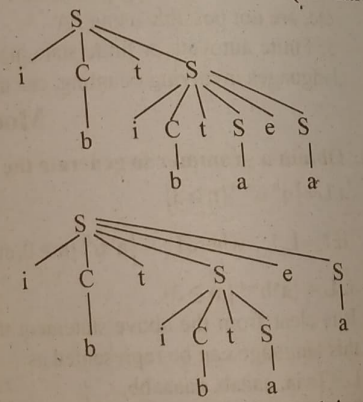
$\Rightarrow ibtSeS$

$\Rightarrow ibtiCtSeS$

$\Rightarrow ibtibtSeS$

$\Rightarrow ibtibtaeS$

$\Rightarrow ibtibtaea$



Since there are two different parse trees for the string 'i btibtaca' by applying atmost derivaton the given grammer is ambiguous.

6. a. For the grammer

$S \rightarrow aABC$

$A \rightarrow aB/a$

$B \rightarrow bA / b$

$C \rightarrow a$

Obtain the corresponding PDA **(08 Marks)**

Ans. Step 1 : Push the start symbol S on to the stack and change the state to $q_1, \delta(q_o, \varepsilon, Z_o) = (q_1, SZ_o)$

Step 2 :

Production	Transition
$S \rightarrow aABC$	$\delta(q_1,a,S) = (q_1, ABC)$
$A \rightarrow aB$	$\delta(q_1, a,A) = (q_1,B)$
$A \rightarrow a$	$\delta(q_1,a,A) = (q_1,\varepsilon)$
$B \rightarrow bA$	$\delta(q_1,b,B) = (q_1,A)$
$B \rightarrow b$	$\delta(q_1,b,B) = (q_1,\varepsilon)$
$C \rightarrow a$	$\delta(q_1,a,C) = (q_1, \varepsilon)$

Step 3 : Finally in state q_1, without consuming any input change the state to q_f which is an accepting state i.e., $\delta(q_1, \varepsilon, Z_o) = (q_f, Z_o)$

$Q = \{q_o, q_1, q_f\}$

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

$\Gamma = \{S, A, B, C, Z_o\}$

δ: is shown below

$\delta(q_o, \varepsilon, Z_o) = (q_1, SZ_o)$

$\delta(q_1, a, S) = (q_1, ABC)$

$\delta(q_1, a, A) = (q_1, B)$

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

$\delta(q_1, b, B) = (q_1, A)$

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

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

$\partial(q_1, \varepsilon, Z_o) = (q_f, Z_o)$

$q_o \in Q$ is the start state of machine

$Z_o \in \Gamma$ is the initial symbol on the stock

$F = \{q_f\}$ is the final state

b. Obtain a CFG that generates the language accepted by PDA **(08 Marks)**

$M = (\{q_o, q_1\}, \{a, b\}, \{A, z\}, \delta, q_o, Z, \{q_f\})$ with the transitions

$\delta(q_o, a, z) = (q_o, AZ)$

$\delta(q_o, b, A) = (q_o, AA)$

$\delta(q_o, a, A) = (q_f, \varepsilon)$

Ans. $\delta(q_o, a, A) = (q_1, \varepsilon)$

For δ of the form $\delta(q_i, a, z) = (q_j, \varepsilon)$	Resulting production
$\delta(q_o, a, A) = (q_1, \varepsilon)$	$(q_o A q_1) = a$

The transitions
$\delta(q_o, a, z) = (q_o, Az)$
$\delta(q_o, b, A) = (q_o, AA)$

For δ of the form	Resulting production
$\delta(q_1, a, z) = (q_1, AB)$	$(q_oZq_o) \rightarrow a(q_oAq_o)(q_oZq_o) \mid a(q_oAq_1)(q_1Zq_o)(q_oZq_1)$ $\rightarrow a(q_oAq_o)(q_oZq_1) \mid a(q_oAq_1)(q_1Zq_1)$
$\delta(q_o, a, z) = (q_o, AZ)$	$(q_oZq_o) \rightarrow b(q_oAq_o)(q_oAq_o) \mid b(q_oAq_1)(q_1Aq_o)$
$\delta(q_o, b, A) = (q_o, AA)$	$(q_oZq_1) \rightarrow b(q_oAq_o)(q_oAq_1) \mid b(q_oAq_1)(q_1Aq_1)$

Module - 4

7. a. Eliminate the useless symbols in the grammar.
S → aA| bB
A → aA | a
B → bB
D → ab | Ea
E → aC|d **(08 Marks)**

Ans.

Old variable	New variable	Productions
ϕ	A,D,E	A → a D → ab E → o|
A,D,E	A,D,E,S	S → aA A → aA D → Ea
A,D,E,S	A,D,E,S	

The resulting grammar $G_1 = (V_1, T_1, P_1, S)$ where
$V_1 = \{A, D, E, S\}$ S is start symbol
$T_1 = \{a, b, d\}$
$P_1 = \{$
$A \rightarrow a \mid aA$
$D \rightarrow ab \mid Ea$
$\in \rightarrow d$
$S \rightarrow aA$
$\}$

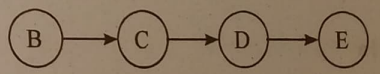
p^1	T^1	V^1
	–	S

S → aA	a	S,A
A → a|aA	a	S,A

$G^1 = (V^1, T^1, P^1, S)$
$V^1 = \{S, A\}$
$T^1 = \{a\}$
$P^1 = \{$
$S \rightarrow aA$
$A \rightarrow a \mid aA$
$\}$
S is start symbol

b. Eliminate all unit productions from the grammar.
S → AB
A → a
B → C | b
C → D
D → E | bC
E → d | Ab **(08 Marks)**

Ans. The non unit productions of the grammar G are shown below
S → AB
A → a
B → b
D → bC
E → d | Ab
Unit productions of the grammar G are
B → C
C → D
D → E



It is clear from above graph D => E, so all non unit production generated from E can also be generated from D.
The non unit production from E are
E → d | Ab
Also be obtained from D
D → d | Ab
The resulting produciton are
D → bC
D → d|Ab
Similarly

C →d|Ab
C →bC
C →d|Ab
B →b
B →d|Ab
B →bC
V^1 = {S,A,B,C,D,E}
T^1 = {a,b,d}
p^1 = {
S → AB
A → a
B → b | d | Ab | bC
C → bC | d | Ab
D → bC | d|Ab
E → d | Ab
}
S is the start symbol

OR

8. a. Convert the following grammar G to GNF **(08 Marks)**

$G = \{(A_1, A_2, A_3), (a, b), P, A_1\}$ where, p consists of the following

Productions :

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow A_1A_2 \mid a$

Ans. We observe $A_1 \rightarrow A_2A_3$ is the only production with A_1 on the hand side. Let us substitute A_2A_3 for A_1 in production $A_3 \rightarrow A_1A_2$

The resulting set of production is :

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow A_2A_3A_2 \mid a$

Similarly A_2 in the production rule for A_3 with A_3A_1 and b

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow A_3A_1A_3A_2 \mid bA_3A_2 \mid a$

$A_3 \rightarrow bA_3A_2B_3$

$A_3 \rightarrow aB_3$

$B_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2B_3$

Resulting

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a$

$B_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2B_3$

Now all the production for A_3 are all in GNF

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b$

$A_3 \rightarrow bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a$

$B_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2B_3$

An equivalent grammar in GNF thus can be written as

$A_1 \rightarrow bA_3A_2B_3A_1A_3 \mid aB_3A_3 \mid bA_3A_2A_1A_3 \mid A_1A_3 \mid bA_3$

$A_2 \rightarrow bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b$

$A_3 \rightarrow bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a$

$B_1 \rightarrow bA_3A_2B_3A_1A_3A_3A_2 \mid bA_3A_2B_3A_1A_3A_3A_2B_3$

$aB_3A_1A_3A_3A_2 \mid aB_3A_1A_3A_3A_2B_3 \mid bA_3A_2A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2B_3 \mid$

$aA_1A_3A_3A_2 \mid aA_1A_3A_3A_2B_3 \mid bA_3A_3A_2 \mid bA_3A_3A_2B_3$

b. Show that L = {w|w∈{a,b,c}}* where $n_a(w) = n_b(w) = n_c(w)$} is not context free. **(08 Marks)**

Ans. The language $L_1 = \{a^n b^n c^n \mid n \geq 0\}$ is obtained by the intersection Land the regular language represented by the regular a*b*c* i.e.,

$\{a^n b^n c^n \mid n \geq 0\} = \{a^* b^* c^* n \{w \mid w \in \{a,b,c\}\}^*$ where $\eta_d(w) = \eta_b(w) = \eta_c(w)\}$

We know that intersection of context free language and regular language is also a context free. But its already known that

$L_1 = \{a^n b^n c^n \mid n \geq 0\}$ is not context free. Since L_1 is not context free, it implies that the given language

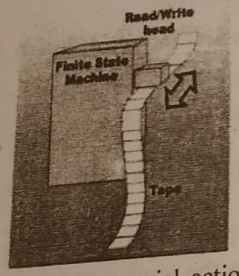
$L = \{w \mid w \in \{a,b,c\}^*$ where $\eta_a(w) = \eta_b(w) = \eta_c(w)\}$ is not context free and not context free grammer.

Module - 5

9. a. Explain the concept of turing machine in detail. **(08 Marks)**

Ans. A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the state of the Turing machine. After reading an input symbol, it is replaced with another symbol, its internal state is changed, and it moves from one cell to the right or left. If the TM reaches the final state, the input string is accepted, otherwise rejected. A TM can be formally described as a 7-tuple

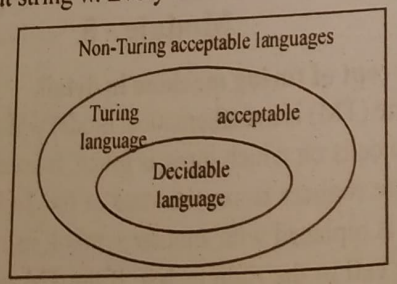
(Q, X, Σ, 8, qo, B, F) where -
Q is a finite set of states
X is the tape alphabet
Σ is the input alphabet
δ is a transition function; δ:Q x X → Q x X x {Left_shift, Right_shift}.
q_o is the initial state
B is the blank symbol
F is the set of final states
A Turing machine is a finite state machine that has an unlimited supply of paper tape that it can write on and read back. There are many formulations of a Turing machine, but essentially the machine reads a symbol from the tape, which is used as an input to the finite state machine. This takes the input symbol and according to it and the current state does three things:
1. It prints something on the tape
2. Moves the tape right or left by one cell
3. Changes to a new state



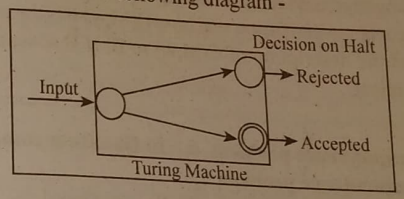
A Turing machine can also perform a special action - it can stop or halt - and surprisingly it is this behavior that attracts a great deal of attention.
For example, a Turing machine is said to recognize a sequence of symbols written on the tape if it is started on the tape and halts in a special state called a final state. What is interesting about this idea is that there are sequences that a Turing machine can recognize that a finite state machine can't.

b. Explain in detail language decidability in the context of theory of computation. (08 Marks)

Ans. A language is called **Decidable** or **Recursive** if there is a Turing machine which accepts and halts on every input string w. Every decidable language is Turing-Acceptable.



A decision problem P is decidable if the language L of all yes instances to P is decidable. For a decidable language, for each input string, the TM halts either at the accept or the reject state as depicted in the following diagram -



Example 1
Find out whether the following problem is decidable or not -
Is a number 'm' prime?
Solution
Prime numbers = {2,3, 5, 7, 11, 13, }
Divide the number 'm' by all the numbers between '2' and 'Vm' starting from '2'.
If any of these numbers produce a remainder zero, then it goes to the "Rejected state", otherwise
it goes to the "Accepted state". So, here the answer could be made by 'Yes' or 'No'.
Hence, it is a decidable problem.

10. a. State and prove the Rice theorem (08 Marks)

Ans. Theorem

Rice's theorem: Any nontrivial property about the language recognized by a Turing machine is undecidable.

A property about Turing machines can be represented as the language of all Turing machines, encoded as strings, that satisfy that property. The property P is about the language recognized by Turing machines if whenever L(M)=L(N) then P contains (the encoding of) M iff it contains (the encoding of) N. The property is non-trivial if there is at least one Turing machine that has the property, and at least one that hasn't.

Proof: Without limitation of generality we may assume that a Turing machine that recognizes the empty language does not have the property P. For if it does, just take the complement of P. The undecidability of that complement would immediately imply the undecidability of P.

In order to arrive at a contradiction, suppose P is decidable, i.e. there is a halting Turning machine B that recognizes the descriptions of Turing machines that satisfy P. Using B we can construct a Turning machine A that accepts the language {(M,w)| M is the description of a Turing machine that accepts the string w}. As the latter problem is undecidable this will show that B cannot exists and P must be undecidable as well.

Let MP be a Turing machine that satisfies P (as P is non-trivial there must be one).

Now A operates as follows:

On input (M,w), create a (description of a) Turing machine C(M,w) as follows:

On input x, let the Turing machine M run on the string w until it accepts (so if it doesn't accept C(M,w) will run forever).

Next run MP on x. Accept iff MP does.
Note that C(M,w) accepts the same language as MP if M accepts w; C(M,w) accepts the empty language if M does not accept w.
Thus if M accepts w the Turing machine C(M,w) has the property P, and otherwise it doesn't. Feed the description of C(M,w) to B. If B accepts, accept the input (M,w); if B rejects, reject.

b. Define
i) Post Correspondence problem & ii) Quantum computation **(08 Marks)**

Ans. i) Post Correspondence problem

The Post Correspondence Problem (PCP), introduced by Emil Post in 1946, is an undecidable decision problem. The PCP problem over an alphabet £ is stated as follows - Given the following two lists, M and N of non-empty strings over £ ~

$M = (x_1, x_2, x_3, \ldots, x_n)$
$N = (y_1, y_2, y_3, \ldots, y_n)$

We can say that there is a Post Correspondence Solution, if for some $i_1, i_2 \ldots\ldots i_k$ ik, where $1 < i_j < n$, the condition $X_{il} \ldots\ldots X_{ik} = Y_{il} \ldots\ldots Y_{ik}$

Find whether the lists
M = (abb, aa, aaa) and N = (bba, aaa, aa) have a Post Correspondence Solution?

	X_1	X_2	X_3
M	Abb	aa	aaa
N	Bba	aaa	aa

Here, $X_2X_1X_3$ = 'aaabbaaa' and $y_2y_1y_3$ = 'aaabbaaa'
We can see that $X_2X_1X_3 = y_2y_1y_3$ Hence, the solution is i = 2, j = 1, **and** k = 3

ii) Quantum computation

Quantum computing studies theoretical computation systems (quantum computers) that make direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data. Quantum computers are different from binary digital electronic computers based on transistors. Whereas common digital computing requires that the data be encoded into binary digits (bits), each of which is always in one of two definite states (0 or 1), quantum computation uses quantum bits, which can be in superpositions of states. A quantum Turing machine is a theoretical model of such a computer, and is also known as the universal quantum computer. The field of quantum computing was initiated by the work of Paul Benioff and Yuri Man in in 1980, Richard Feynman in 1982, and David Deutsch in 1985. A quantum computer with spins as quantum bits was also formulated for use as a quantum spacetime in 1968.

As of 2017, the development of actual quantum computers is still in its infancy, but experiments have been carried out in which quantum computational operations were executed on a very small number of quantum bits. Both practical and theoretical research continues, and many national governments and military agencies are funding quantum computing research in an effort to develop quantum computers for civilian, business, trade, environmental and national security purposes, such as cryptanalysis.

Fifth Semester B.E. Degree Examination
CBCS - Model Question Paper - 3
AUTOMATA THEORY AND COMPUTABILITY

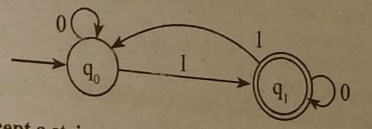
Time: 3 hrs. Max. Marks: 80

Note : Answer any FIVE full questions, selecting ONE full question from each module.

MODULE - 1

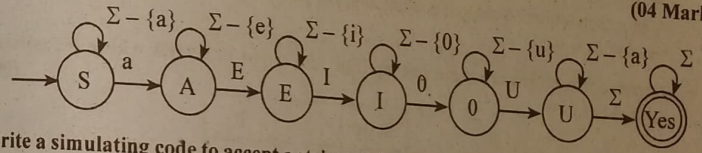
1. a. Design a DFSM M : Which will check for a odd parity over the binary string 0 and 1.

Ans. L = {w ∈ {0, 1} * : w has odd parity} **(04 Marks)**



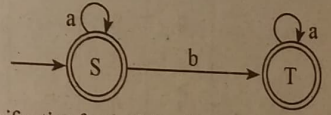
b. Design a DFSM to accept a string that forms vowels : L = {w ∈ {a – z}*}

Ans. **(04 Marks)**



c. Write a simulating code to accept a string of a's and b's where the machine has to reject if it encounter's more than one b. **(08 Marks)**

Ans. The DFSM for the string a's & b's which accept only one b is as follows.



We could view M as specification for the following program:
Until accept or reject do :
S : S = get-next-symbol
if S = end-of-file then accept
Else if s = a then go to S
Else if s = b then go to T
T : S = get-next-symbol
if S = end-of-file then accept
Else if s = a then go to T
Else if s = b then reject
End

OR

2. a. Define Canonical form for regular languages **(04 Marks)**

Ans. A canonical form for some set of objects C assigns exactly one representation to each class of "equivalent" object in C. Further, each such representation is distinct, so two objects in C share the same representation if they are "Equivalent" in the sense for which we define the form.

The ordered binary decision diagram (OBDD) is a canonical form for Boolean expression that makes if possible for model checkers to verify the correctness of very large concurrent systems and hardware circuits.

b. Explain the Moore Machine with mathematical notion. **(04 Marks)**

Ans. A Moore machine M is a seven - tuple (K, Σ, O, δ, D, S, A) where:

- K is a finite set of states
- Σ is a input alphabet
- O is an output alphabet
- $S \in K$ is the start state
- $A \subseteq K$ is the set of accepting states (although for some applications thieole signation is not important)
- δ is the transition function. It is the function from (K) to (0*)

A Moore machine M computes a function f(w) iff, when it reads the input string w, its output sequences is f(w).

c. Construct a minimum state automation equivalent to a DFA whose transition table is shown below : **(08 Marks)**

State	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_4	q_3
q_2	q_4	q_3
(q_3)	q_5	q_6
(q_4)	q_7	q_6
q_5	q_3	q_6
q_6	q_6	q_6
q_7	q_4	q_6

Ans.

$Q_1^0 = \{q_3, q_4\}$, $Q_2^0 = \{q_0, q_1, q_2, q_5, q_6, q_7\}$

$\pi_0 = \{\{q_3, q_4\}, \{q_0, q_1, q_2, q_5, q_6, q_7\}\}$

q_3 is 1 – equivalent to q_4. $So_1\{q_3, q_4\} \in \pi_1$

q_0 is not 1 – equivalent to q_1, q_2, q_5 but q_0 is 1 – equivalent to q_6.

Hence $\{q_0, q_6\} \in \pi_1$ .. q_1 is 1 – equivalent to q_2 but not 1 – eqivalent to q_5, q_6 or q_7.

$So_1\{q_5, q_7\} \in \pi_1$

q_5 is not 1 – equivalent to q_6 but to q_7. $So_1\{q_5, q_7\} \in \pi_1$

Hence $\pi_1 = \{\{q_3, q_4\}, \{q_0, q_6\}, \{q_1, q_2\}, \{q_5, q_7\}\}$

q_3 is 2 – equivalent to q_4. $So\{q_3, q_4\} \in \pi_2$

q_0 is not 2 – equivalent to q_6. $So\{q_0\}, \{q_6\} \in \pi_2$

q_1 is 2 – equivalent to q_2. $So\{q_1, q_2\} \in \pi_2$

q_5 is 2 – equivalent to q_7. $So_1\{q_5, q_7\} \in \pi_2$

Hence $\pi_2 = \{\{q_3, q_4\}, \{q_0\}, \{q_6\}, \{q_1, q_2\}, \{q_5, q_7\}\}$

q_3 is 3 – equivalent to q_4; q_4 is 3 – equivalent to q_2 and q_5 is 3 – equivalent to q_7. Hence $\pi_3 = \{\{q_0\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_5, q_7\}, \{q_6\}\}$ As $\pi_3 = \pi_2$ the minimum state automation is $M^1 = \left(Q^1, \{a, b\}, \delta_1^1[q_0], \{[q_3, q_4]\}\right)$

State	a	b
$[q_0]$	$[q_1, q_2]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_3, q_4]$	$[q_3, q_4]$
$[q_3, q_4]$	$[q_5, q_7]$	$[q_6]$
$[q_5, q_7]$	$[q_3, q_4]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_6]$

Module - 2

3. a. Obtain the regular expression for the following language

i) $L = \{a^n b^m \mid m \geq 1, n \geq 1, nm \geq 3\}$ **ii) $L = \{a^{2n} b^{2m} \mid n \geq 0, m \geq 0\}$** **(08 Marks)**

Ans. **i) $L = \{a^n b^m \mid m \geq 1, n \geq 1, nm \geq 3\}$**

Case 1 : since $nm \geq 3$, if $m = 1$ then $n \geq 3$ i.e., RE is given by aaaa*b

Case 2 : since $mn \geq 3$, if $n = 1$ then $m \geq 3$ i.e., RE is given by abbbb*

Case 3 : since $nm \geq 3$, if $m \geq 2$ and $n \geq 2$ i.e., RE is given by aaa* bbb*

So, final regular expression is

RE = aaaa*b + abbbb* +aaa* bbb*

ii) $L = \{a^{2n} b^{2m} \mid n \geq 0, m \geq 0\}$

For every $n \geq 0$, a^{2n} results in even number of a's and for every $m \geq 0$ b^{2m} results in even number of b's. The regular expression representing even number of a's and b's is given by

RE = (aa)*

RE = (bb)* so final regular expression is

RE = (aa)* (bb)*

b. Let $\Sigma = \{0, 1\}$ $\Gamma = \{0, 1, 2\}$ and $h(0) = 01$, $h(1) = 112$. What is h(010)? If $L = \{00.\ 010\}$ What is homomorphism image of L? **(04 Marks)**

Ans. $h(w) = h(a_1)\, h(a_2) \ldots h(a_n)$

So, $h(010) = h(0)\, h(1)\, h(0) = 0111201$

$L = \{00_1 010\} = L(h(00), h(010))$
$= L(h(0)(0), h(0)(h_1)h(0))$
$= L(0101, 0111201)$

Therefore $h(010) = 0111201$
$L(00, 010) = L(0101, 0111201)$

c. What are the various limitations of finite automata? **(04 Marks)**

Ans. 1) An FA has finite number of states and so it does not have the capacity to remember arbitrary long amount of information.
2) Since it does not have memory, FA can not remember a long string. For example : String is palindrome or not.
3) Finite automata or finite state machine have trouble recognizing various types of languages involving counting, calculating storing the string.

OR

4. a. State and prove that regular grammars Define exactly the regular languages. **(08 Marks)**

Ans. **Theorem :-** The class of languages that can be defined with regular grammars is exactly the regular languages.

Proof :- We first show that any language that can be defined with a regular grammar can be accepted by some FSM and so is regular. Then we must show that every regular language can be defined with a regular grammar. Both proofs are by construction.

Regular grammar → FSM : The following algorithm constructs an FSM M from a regular grammar $\Sigma = (V, \Sigma, R, S)$ and assures that $L(M) = L(G)$:

Grammar to FSM (G : regular grammar)

1. Create in M a separate state for each non terminal in V.
2. Make the state corresponding to S the start state.
3. If there are any rules in R of the form $x \rightarrow w_1$ for some $w \in \Sigma$ the create an additional state labeled #.
4. For each rule of the form $x \rightarrow wy$, add a transition from x to y labeled w.
5. For each rule of the form $x \rightarrow w$, add a transition from x to # labeled w.
6. For each rule of the form $x \rightarrow \varepsilon$, mark state X as accepting.
7. Mark state # as accepting.
8. If M is incomplete, M requires a dead state. Add a new static for every (q, i) pair for which no transition has already been definite create a transition from q to D labeled i. For every i in Σ_1 create a transition from D to D labeled i.

b. Show that the set $L = \{a^{i^2} \mid i \geq 1\}$ is not regular. **(08 Marks)**

Ans. **Step 1 :-** Suppose L is regular. let n be the number of states in the finite automaton accepting L.

Step 2 :- Let $w = a^{n^2}$. Then $|w| = n^2 > n$. By pumping Lemma, we can write w = xyz with $|xy| \leq n$ and $|y| > 0$.

Step 3 :- Consider $xy^2z \mid xy^2z| = |x| + 2|y| + |z| > |x| + |y|$ as $|y| > 0$. This means $n^2 = |xy^2z| = |x| + |y| + |z| < |xy^2z|$. As $|xy| \leq n$,

We have $|y| \leq n$. Therefore

$|xy^2z| = |x| + 2|y| + |z| \leq n^2 + n$

i.e, $n^2 < |xy^2z| \leq n^2 + n < n^2 + n + n + 1$.

Hence, $|xy^2z|$ strictly lies between n^2 and $(n + 1)^2$ but is not equal to any one of them. Thus $|xy^2z|$ is not a prefect square and so xy^2z L, But by pumping Lemma, $xy^2z \in L$, This is a contradiction.

Module - 3

5. a. Obtain a grammar to generate integer number and derive for +1965 from the productions **(08 Marks)**

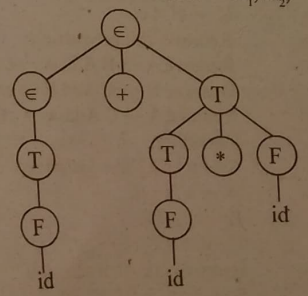
Ans. G = (V, T, P, S)
V = {D, S, N, I}
T = {+, –, 0, 1, 3, 4, 5, 6, 7, 8, 9}
P = {
I → N | SN (Generate signed / Unsigned number)
N → D | ND | DN (Generate one or more digits)
S → +|–|ε (Generate the sign)
D → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 (Generate digit)
}
S = I which is start symbol.
The signed number + 1965 can be derived as shown
I ⇒ SN
⇒ +N
⇒ +ND
⇒ +N5
⇒ +ND5
⇒ +N65
⇒ +ND65
⇒ +N965
⇒ +D965
⇒ +1965

b. Define parse tree. Obtain the parse tree for string id + id * is from the grammar **(08 Marks)**

E → E + T | T
T → T * F | F
F → id

Ans. Let G = (V, T, P, S) be a CFG, The tree is derivation tree (parse tree) with following properties.
1. The root has the labels.
2. Every vertex has a label which is in (V u T u ε)
3. Every leaf node has label from T and an interior vertex has a label from v.
4. If a vertex is label A and if $X_1, X_2, X_3, \ldots X_n$ are all children of A from left then $A \rightarrow X_1, X_2, X_3 \ldots X_n$ must be a production in P.

E ⇒ E + T
⇒ E + T * F
⇒ E + T * id
⇒ E + F * id
⇒ E + id * id
⇒ T + id * id
⇒ F + id * id
⇒ id + id * id



OR

6. a. Obtain a PDA to accept the language L(M) = {w|w ∈ (a + b) * $n_a(w) = n_b(w)$ by a final state, and show the I_D to reject aabbb. **(08 Marks)**

Ans. M (Q, Σ, Γ, δ, q_0, Z_0, F)

Q = {q_0, q_1}

Σ = {a, b}

Γ = {a, b, z_0}

δ : $\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, b, z_0) = (q_0, bz_0)$

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

$\delta(q_0, b, b) = (q_0, bb)$

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

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

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

$q_0 \in Q$ is the start state of the machine

$z_0 \in \Gamma$ is the initial symbol on the stack

F = {q_1} is the final state

Initial ID

$(q_0, aabbb, z_0) \leftarrow (q_0, abbb, az_0)$

$\leftarrow (q_0, bbb, aaz_0)$

$\leftarrow (q_0, bb, az_0)$

$\leftarrow (q_0, b, z_0)$

$\leftarrow (q_0, \varepsilon, bz_0)$

(Final configuration)

Since the transition $\delta(q_0, \varepsilon, b)$ is not defined the string aabbb is rejected by PDA.

b. Convert the grammar to chomsky normal form **(08 Marks)**

G = ({S, A, B, C, a, c}, {A, B, C}, R, S}), where

R = {S → a A C a

A → B | a

B → C | c

C → cC | ε}

Ans. S → aAca | aAa | aCa|aa

A → B|a

B → C|c

C → cC | c

Remove unit production

Remove A → B Add A → C|c

Remove B → C Add B → cC

Remove A → C Add A → cC

Therefore

S → aACa | aAa | aCa | aa

A → a | c | cC

B → c | cC

C → cC | C

Remove mixed production

$S \rightarrow T_aACT_a \mid T_aAT_a \mid T_aT_a$

A → a | c | T_cC

B → C | T_cC

C → T_cC | C

$T_a \rightarrow a$

$T_c \rightarrow C$

Remove long sequence

$S \rightarrow T_aS_1$ $S \rightarrow T_aS_3$ $S \rightarrow T_aS_4$ $S \rightarrow T_aT_a$

$S_1 \rightarrow AS_2$ $S_3 \rightarrow AT_a$ $S_4 \rightarrow CT_a$

$S_2 \rightarrow CT_a$

Finally

A → a | c | T_cC

B → C | T_cC

C → T_cC | C

$T_a \rightarrow a$

$T_c = C$

Module - 4

7. a. Show that L = {$a^nb^nc^n$ | n ≥ 1} is not context free but context sensitivity. **(08 Marks)**

Ans. **Step 1 :-** Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2 :- Let Z = $a^nb^nc^n$. then |Z| = 3n > n. write Z = uv wxy, where |vx| ≥ 1, i.e., at least one of V or x is not A.

Step 3 :- uvwxy = $a^nb^nc^n$. As | ≤ |Vx| ≤ n. v or x cannot contain all the three symbol a,b,c. So (i) v or x is of the form a^ib^i(or b^ic^i) for some i, j such taht i+j ≤ n. or (ii) v or x is a string formed by the repetition of only one symbol among a,b,c.

When v or x is of the form a^ib^j, $v^2 = a^ib^i$, a^ib^j (or $x^2 = a^ib^j$, a^ib^j). As V^2 is a substring of uv^2wx^2y, we cannot have uv^2wx^2y fo the form $a^mb^mc^m$. av^2wx^2y L.

When both v and x are formed by the repetition of a single symbol, the string uwy will contain the remaining symbol, say a_1. Also, a_1^n will be substring of uwy as a $_1$ does not occurrences of a_1. So uv^owx^o = uwy L

Thus for any choice of v or x, we get a contradiction. There for e, L is not context free, but context sensitive.

b. Prove that context free languages are Nonclosure under Intersection complement and difference. **(08 Marks)**

Ans. **Proof :-** The context free languages are not closed under intersection. The proof is by counter example Let :

$L_1 = \{a^nb^nc^m : n, m \geq 0\}$

$L_2 = \{a^mb^nc^m : n, m \geq 0\}$

Both L_1 and L_2 are context free since there exist straight forward context free grammars for them.

$L = L_1 \eta L_2$

$= \{a^n b^n c^n : n \geq 0\}$. If the CFL were closed under intersection L would have to be context free.

The CFL are not closed under complement given any sets L_1 and L_2.

$L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$

The CFL are closed under union so if they were also closed under complement, they would necessarily be closed under intersection. But it is not. Thus they are not closed under complement either.

The CFL are not closed under difference given any language L,

$\neg L = \Sigma^* - L$

Σ^* is context free, so, if the CFL were closed under difference, the complement of any context - free language would necessarily be context free. But it is not.
Therefore CFL are Non closure under Intersection complement and difference.

OR

8. a. **Consider the transition table for the Turning machine. Draw the computation sequence of the input string 00b.** **(08 Marks)**

State	Tape symbol		
	b	0	1
$\rightarrow q_1$	$1Lq_2$	$0Rq_1$	
q_2	bRq_3	$0Lq_2$	$1Lq_2$
q_3		bRq_4	bRq_5
q_4	$0Rq_5$	$0Rq_4$	$1Rq_5$
q_0	$0Lq_2$		

Ans. We describe the computation sequence if term of the contents of the tape and current state. If the string in the tape is $a_1a_2....a_ja_{j+1}...a_m$ and the TM in state q is to read a_{j+1}, the we write $a_1a_2...a_jq_{j+1}....a_m$.
For the input string oob, we get the following sequence
$q_100b \vdash 0q_10b \vdash 00q_1b \vdash 0q_201 \vdash q_2001 \vdash q_2b001 \vdash bq_3001 \vdash bbq_401 \vdash bb0q_41$
$\vdash bb01q_4b \vdash bb010q_5 \vdash bb01q_200 \vdash bbq_2100 \vdash bbq_20100 \dashv bq_2b0100 \vdash bq_30100$
$\vdash bbbq_2100 \vdash bbb_1q_400 \vdash bbb10q_40 \vdash bbb100q_4b \vdash bbb1000q_5b \vdash bbb1000v_200 \vdash$
$bbb10q_200 \vdash bbb1q_20000 \vdash bbbq_210000 \vdash bbq_2b10000 \vdash bbbq_310000 \vdash bbbbq_50000$

b. **Design a TM which can multiply two positive integers.** **(08 Marks)**

Ans. The input (m,n),m, n being given, the positive integers are represented by $o^m|o^n$. M starts with $o^m|o^n$ in its tape. At the end of the computation, o^{mn} surrounded by b's is obtained as the output.
The major steps in the construction are as follows:
1. $o^m|o^n|$ is placed on the tape
2. The leftmost 0 is erased.
3. A block of n 0's is copied onto the right end.
4. Step 2 and 3 are repeated m times and 10^m, 10^{mn} is obtained on the tape.
5. The prefix 10^m1 of $1\ 0^m10^{mn}$ is erased, leaving the product mn as the output.
For every 0 in 0^m, 0^n is a doled onto the right end. this requires repetition of step 3 we defined a subroutine called copy for step 3.

State	Tape symbol			
	0	1	2	b
q_1	q_20R	q_41L	–	–
q_2	q_20R	q_21R	–	q_30L
q_3	q_30L	q_31L	q_12R	–
q_4	–	q_51R	q_40L	–
q_5	–	–	–	–

Module - 5

9. a. **Does the pcp with two lists x = (b, bab^3, ba) and y(b^3, ba,a) have a solution?** **(04 Marks)**

Ans. We have to determine whether or not there exists a sequence of substrings of x such that string formed by this sequence and the string formed by the sequence of corresponding substrings of y are identical. The required sequence is given by $i_1 = 2, i_2 = 1, i_3 = 1, i_4 = 3$ i.e., (2,1,1,3) and m = 4
The corresponding strings are :

bab^3	b	b	ba	=	ba	b^3	b^3	a
x_2	x_1	x_1	x_3		y_2	y_1	y_1	y_3

Thus pcp has a solution.

b. **Prove that pcp with two lists x = (01, 1, 1) y = $(0)^2$ 10, 1^1)** **(04 Marks)**

Ans. For each substring $x_i \in x$ and $y_i \in y$ we have $|x_i| < |y_i|$ for all i. Hence the string generated by a sequence of the substring of x is shorter than the string generated by the sequence of corresponding substring of y. Therefore, the pcp has no solution.

c. **Let $f(n) = 4n^3 + 5n^2 + 7n + 3$. Prove that $f(n) = 0(n^3)$** **(08 Marks)**

Ans. Let $f(n) = 4n^3 + 5n^2 + 7n + 3$
In order to prove that $f(n) = 0(n^3)$, take c = 5 and $N_0 = 10$, then
$f(n) = 4n^3 + 5n^2\ 7n + 3 \leq 5n^3$ for $n \geq 10$
when n = 10,
$5n^2 + 7n + 3 = 573 < 10^3$ for $n > 10$, $5n^2 + 7n + 3 < n^3$
Then $f(n) = 0(n^3)$
Therefore the function $f(n) = 4n^3 + 5n^2 + 7n + 3$ is $f(n) = 0(n^3)$, the order of the growth of the function is in cubic order.

OR

10. a. **Write short notes on:**
i. Growth rate of algorithm
ii. Classes of P and NP
iii. NP- complete problem
iv. Subroutine in TM **(16 Marks)**

Ans. **i) Growth rate of algorithm:** Algorithms analysis is all about understanding growth rates. That is as the amount of data gets bigger, how much more resource will my algorithm require? Typically, we describe the resource growth rate of a piece of code in terms of a function. The algorithm may have different kind of growth rate, following are few growth rate
1. Constant growth rate
2. Logarithmic growth rate
3. Linear growth rate
4. Log linear
5. Quadratic growth rate
6. Cubic growth rate
7. Exponential growth rate

ii) Classes of P and NP

An algorithm is said to be polynomially bounded if its worst-case complexity is bounded by a polynomial function of the input size. A problem is said to be polynomially bounded if there is a polynomially bounded algorithm for it.

P is the class of all decision problems that are polynomially bounded. The implication is that a decision problem X e P can be solved in polynomial time on a deterministic computation model (such as a deterministic Turing machine).

NP represents the class of decision problems which can be solved in polynomial time by a non-deterministic model of computation. That is, a decision problem X e NP can be solved in polynomial-time on a non-deterministic computation model (such as a non-deterministic Turing machine). A non-deterministic model can make the right guesses on every move and race towards the solution much faster than a deterministic model.

iii) NP-Complete problem

This means that the problem can be solved in Polynomial time using a Non-deterministic Turing machine (like a regular Turing machine but also including a non-deterministic "choice" function). Basically, a solution has to be testable in poly time. If that's the case, and a known NP problem can be solved using the given problem with modified input (an NP problem can be reduced to the given problem) then the problem is NP complete.

The main thing to take away from an NP-complete problem is that it cannot be solved in polynomial time in any known way. NP-Hard/NP-Complete is a way of showing that certain classes of problems are not solvable in realistic time.

iv) Subroutine in TM:

TM program for the subroutine is written. This will have an initial state and a return state. After reaching the return state, there is a temporary halt. For using a subroutine, new states are introduced. When there is a need for calling the subroutine, moves are effected to enter the initial state for the subroutine and return to the main program of TM.

Fifth Semester B.E. Degree Examination, CBCS - Dec 2017 / Jan 2018

Automata Theory & Compatibility

Time: 3 hrs. Max. Marks: 80

Note : *Answer any FIVE full questions, selecting ONE full question from each module.*

Module - 1

1. a. **Define the following terms with examples:**
(i) Alphabet (ii) Power of an alphabet
(iii) Concatenation (iv) Languages **(04 Marks)**

Ans. **i. Alphabet :** A language consists of various symbol from which the words, statements etc, can be obtained. These symbols are called Alphabets. The symbol Σ denotes the set of alphabets of a language.

Ex : Σ = {a,b,.....z, A,B,.....,Z,0.....9,#,C,),}....etc}

ii Power of an alphabet : If Σ is an alphabet, we can epress the set of all strings of a certain length from that alphabet by using the exponential notation.

Ex: Σ = {0,1} the

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

iii. Concatenation : The concatenation of two strings u and v is the string obtained by writing the letters of string u followed by the letters of string v.

$u = a_1a_2a_3.....a_n \quad v = b_1b_2b_3.....b_n$

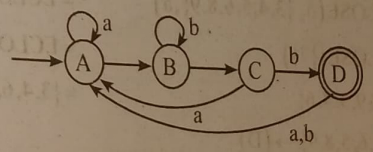
$uv = a_1a_2a_3.....a_nb_1b_2b_3.....b_n$

iv. Language : A language can be defined as a set of strings obtained from Σ* where Σ is set of alphabets of a particular language.

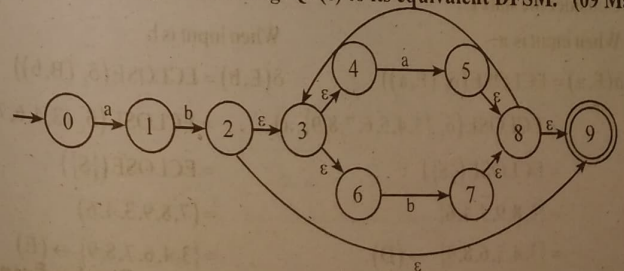
Ex : {ε,01,10,0011,1010,0101,0011,.....}

b. Draw a DFA to accept strings of a's and b's ending with 'bab'. **(03 Marks)**

Ans.



c. Convert the following NDFSM Fig. Ql (c) to its equivalent DFSM. **(09 Marks)**



Ans. Consider the state A :

When input is a :

$$\delta(A,a) = ECLOSE(\delta_E(A,a))$$
$$= ECLOSE(\delta_E(0,a))$$
$$= \{1\} \rightarrow (B)$$

When input is b :

$$\delta(A,b) = ECLOSE(\delta_E(A,b))$$
$$= ECLOSe(\delta_E(0,b))$$
$$= \{\phi\}$$

Consider the state B:

When input is a:

$$\delta(B,a) = ECLOSE(\delta_E(B,a))$$
$$= ECLOSE(\delta_E(1,a))$$
$$= \phi$$

When input is b

$$\delta(B,b) = ECLOSE(\delta_E(B,b))$$
$$= ECLOSE(\delta_E(1,b))$$
$$= ECLOSE(\{2\})$$
$$= \{2,3,4,6,9\} \rightarrow (6)$$

Consider the state C :

When input is a :

$$\delta(c,a) = ECLOSE(\delta_E(c,a))$$
$$= ECLOSE(\delta_E\{2,3,4,6,9\},a)$$
$$= ECLOSE(5)$$
$$= \{5,8,9,3,4,6\}$$
$$= \{3,4,5,6,8,9\} \rightarrow (D)$$

When input is b :

$$\delta(A,b) = ECLOSE(\delta E(A,b))$$
$$= ECLOS(\delta_E\{2,3,4,6,9\},b)$$
$$= ECLOSE(\{7\})$$
$$= \{7,8,9,3,4,6\}$$
$$= \{3,4,6,7,8,9\} \rightarrow (E)$$

Consider the state D :

When input is a:

$$\delta(D,a) = ECLOSE(\delta_E(D,a))$$
$$= ECLOSE(\delta_E\{3,4,5,6,8,9\},a)$$
$$= ECLOSE(\{7\})$$
$$= \{7,8,9,3,4,6\}$$
$$= \{3,4,6,5,8,9\} \rightarrow (D)$$

When input is b

$$\delta(B,b) = ECLOSE(\delta_E(B,b))$$
$$= ECLOSE(\delta E\{3,4,6,8,5,9\},b)$$
$$= ECLOSE(7,8,9,3,4,6)$$
$$= \{3,4,6,7,8,9\} \rightarrow (E)$$

Consider the state E :

When input is a:

$$\delta(E,a) = ECLOSE(\delta_E(E,a))$$
$$= ECLOSE(\delta_E\{3,4,5,6,7,8,9\},a)$$
$$= ECLOSE(\{S\})$$
$$= \{5,8,9,3,4,6\}$$
$$= \{3,4,5,6,8,9\} \rightarrow (D)$$

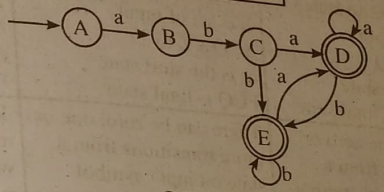
When input is b

$$\delta(E,b) = ECLOSE(\delta_E(B,b))$$
$$= ECLOSE(\delta_E\{3,4,6,7,8,5,9\},b)$$
$$= ECLOSE(\{S\})$$
$$= (7,8,9,3,4,6)$$
$$= \{3,4,6,7,8,9\} \rightarrow (E)$$

Since no new state, will stop

Since no new state, will stop

δ	a	b
A	B	ϕ
B	ϕ	C
*C	D	E
*D	D	E
*E	D	E

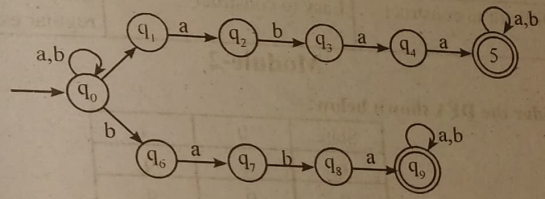


OR

2. a. Draw a DFSM to accept the language,

$1 = \{\omega \in \{a, b\}^* : \forall x, y \in \{a, b\}^* ((\omega = x\,abbaay) \vee (\omega = x\,babay))\}$ (03 Marks)

Ans.



b. Define distinguishable and indistinguishable states. Minimize the following DFSM,

S	0	1
A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

(i) Draw the table of distinguishable and indistinguishable state for the automata.
(ii) Construct minimum state equivalent of automata. **(09 Marks)**

Ans. Refer Q.no.4(a) of MQP - 2.

c. Write differences between DFA, NFA and e -NFA. **(04 Marks)**

Ans.

DFA	NFA	ε - NFA
1. The DFA is S-tuple $M = (Q,\Sigma,\delta,q_0,F)$ where Q is finite states Σ is set of input $\delta : Q \times \Sigma$ to Q q_0 is the start state FCQ is set of final state	An NFA is 5 - tuple $(\Sigma, Q, \delta, q_0, F) = M$ Where Q is firite states Σ is set of input $\delta : Q \times \Sigma$ to 2^Q q_0 is the sturt state FCQ is final state	Am ε - NFA is 5 - tuple $M = (Q,\Sigma,\delta,q_0,F)$ where Q is finite states Σ is set of input $\delta : Q \times (\Sigma \cup Q)$ to Z^Q q_0 is th start state $F \leq Q$ is final state
2. There can be zero or one transition from a state on an input symbol	There can be zero, one or more transitions from a state on input symbol	There can be zero , one or more transition from state with or without any input symbol
3. More number of transition	Less number of transition	Relatively more transition when compared with NFA
4. Difficult to construct	Easy to construct	Easy to construct using regular expression.

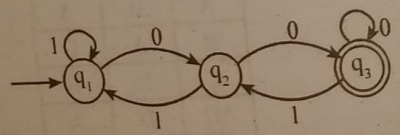
Module-2

3. a. Consider the DFA shown below:

State	0	1
q_1	q_2	q_1
q_2	q_3	q_1
* q_3	q_3	q_2

Obtain the regular expressions $R_{ij}^{(0)}$, $R_{ij}^{(1)}$ and simplify the regular expressions as much as possible. **(09 Marks)**

Ans.



Basic when k = 0

$R_{11}^{(0)} = \varepsilon + 1$ $R_{23}^{(0)} = 0$

$R_{12}^{(0)} = 0$ $R_{31}^{(0)} = \phi$

$R_{13}^{(0)} = \phi$ $R_{32}^{(0)} = 1$

$R_{21}^{(0)} = 1$ $R_{33}^{(0)} = \varepsilon + 0$

$R_{22}^{(0)} = \phi + \varepsilon = \varepsilon$

Induction: $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}\left[R_{kk}^{(k-1)}\right]^* R_{kj}^{(k-1)}$

When k =1

$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)}\left[R_{11}^{(0)}\right]^* R_{11}^{(0)}$

$= (\varepsilon+1)(\varepsilon+1)(\varepsilon+1)^*(\varepsilon+1)$

$= (\varepsilon+1) + (\varepsilon+1)1^*(\varepsilon+1)$

$= (\varepsilon+1) + 1^*$

$= 1^*$

$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)}\left[R_{11}^{(0)}\right]^* R_{13}^{(0)}$

$= 0 + 1(\varepsilon+1)^*\phi$

$= 0$

$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)}\left[R_{11}^{(0)}\right]^* R_{12}^{(0)}$

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

$= 0 + 1^* 0$

$= 1^* 0$

$R_{31}^{(1)} = R_{31}^{(0)} + R_{31}^{(0)}\left[R_{11}^{(0)}\right]^* R_{11}^{(0)}$

$= \phi + \phi(\varepsilon+1)^*(\varepsilon+1)$

$= \phi$

$R_{13}^{(1)} = R_{13}^{(0)} + R_{11}^{(0)}\left[R_{11}^{(0)}\right]^* R_{13}^{(0)}$

$= \phi + (\varepsilon+1)^*(\varepsilon+1)\phi$

$= \phi$

$R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)}\left[R_{11}^{(0)}\right]^* R_{12}^{(0)}$

$= 1 + \phi(\varepsilon+1)^* 0$

$= 1$

$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}\left[R_{11}^{(0)}\right]^* R_{11}^{(0)}$

$= 1 + 1(\varepsilon+1)^*(\varepsilon+1)$

$= 1 + 11^* = 11^*$

$R_{33}^{(1)} = R_{33}^{(0)} + R_{31}^{(0)}\left[R_{11}^{(0)}\right]^* R_{13}^{(0)}$

$= (\varepsilon+0) + \phi(\varepsilon+0)^*\phi$

$= (\varepsilon+0)$

$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}\left[R_{11}^{(0)}\right]^* R_{12}^{(0)}$

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

$= \varepsilon + 11^* 0$

When k = 2

$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)}\left[R_{22}^{(1)}\right]^* R_{21}^{(1)}$

$= 1^* + 1^*0(\varepsilon + 11^*0)^*11^*$

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

$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}\left[R_{22}^{(1)}\right]^* R_{22}^{(1)}$

$= 1^*0 + 1^*0(\varepsilon + 11^*0)^*(\varepsilon + 11^*0)$

$= 1^*0 + 1^*0(11^*0)^*(\varepsilon + 11^*0)$

$R_{13}^{(2)} = R_{13}^{(1)} + R_{12}^{(1)}\left[R_{22}^{(1)}\right]^* R_{23}^{(1)}$

$= \phi + 1^*0(\varepsilon + 11^*0)^*0$

$= (0 + \varepsilon) + 1(11^*0)^*0$

$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)}\left[R_{22}^{(1)}\right]^* R_{21}^{(1)}$

$= 11^* + (\varepsilon + 11^*0)(\varepsilon + 11^*0)^*11^*$

$= 11^* + (\varepsilon + 11^*0)(11^*0)11^*$

$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)}\left[R_{22}^{(1)}\right]^* R_{22}^{(1)}$

$= (\varepsilon + 11^*0) + (\varepsilon + 11^*0)(\varepsilon + 11^*0)^*(\varepsilon + 11^*0)$

$= (\varepsilon + 11^*0) + (\varepsilon + 11^*0)(11^*0)(\varepsilon + 11^*0)$

$R_{31}^{(2)} = R_{31}^{(1)} + R_{32}^{(1)}\left[R_{22}^{(1)}\right]^* R_{21}^{(1)}$

$= \phi + 1(\varepsilon + 11^*0)^*11^*$

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

$R_{32}^{(2)} = R_{32}^{(1)} + R_{32}^{(1)}\left[R_{22}^{(1)}\right]^* R_{22}^{(1)}$

$= 1 + 1(\varepsilon + 11^*0)^*(\varepsilon + 11^*0)$

$= 1 + 1(11^*0)^*(\varepsilon + 11^*0)$

$R_{33}^{(2)} = R_{33}^{(1)} + R_{32}^{(1)}\left[R_{22}^{(1)}\right]^* R_{23}^{(1)}$

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

Final RE can be calculated as

$R_{13}^{(3)} = R_{13}^{(2)} + R_{13}^{(2)}\left[R_{33}^{(2)}\right]^* R_{33}^{(2)}$

$= 1^*0(11^*0)^*0 + 1^*0(11^*0)^*0\left[(0 + \varepsilon) + 1(11^*0)^*0\right]^*(0 + \varepsilon) + 1(11^*0)^*0$

b. Give Regular expressions for the following languages on /_= {a,b,c}
(i) all strings containing exactly one a
(ii) all strings containing no more than 3 a's.
(iii) all strings that contain at least one occurance of each symbol in V .
(03 Marks)

Ans.

(i) $R \in = (b + c)^*a(b + c)^*$

(ii) $R \in = (b + c)^*(\sigma + a)(b + c)^*(\varepsilon + a)(b + c)^*$

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

3. c. Let L be the language accepted by the following finite state machine. (04 Marks)

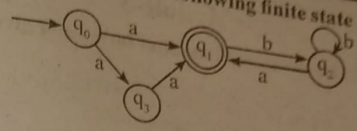


Fig. Q3 (c;

Indicate for each of the following regular expressions, whether it correctly describes L:
(i) (a U ba) bb* a
(ii) (ε U b) a (bb* a)*
(iii) ba U ab*a
(iv) (a U ba) (bb*a)*

Ans. i. NO
ii. YES
iii. NO
iv. YES

OR

4. a. Prove that the following language in not regular :
$L = \{0^n 1^n \mid \eta > 0\}$

Ans. Step 1 : Let L i.e, regular and η be the number of states **(05 Marks)**
$x = 0^n b^n$

Step 2 : Since $|x| = 2\eta > \eta$ we can split x into uvw such that $|uv| \leq \eta$ and $|V| \geq 1$ as

$x = \overbrace{a\,a\,a\,a\,a\,a}^{n}\ \underbrace{a}_{v}\ \overbrace{b\,b\,b\,b\,b\,b\,b\,b}^{n}$

u w

Step 3 : According to pumping lemma $uv^1\omega \in L$ for i = 0,1,2.....
When i = 0 'V' doesn't exist so $L = \{0^n 1^n \mid n > 0\}$ is not regular

b. If L_1 and L_2 are regular languages then prove that L_1UL_2, $L_1 . L_2$ and L_1^* are regular languages.
Ans. Refer Q.no.3(b) of MQP - 2. **(05 Marks)**

c. Is the following grammar is ambiguous? (06 Marks)
S → i C ts|ict ses|a
C → b

Ans. Refer Q.no.5(b) of MQP - 2. **(06 Marks)**

Module-3

5. a. Define Grammar, Derivation, Sentential forms and give one example for each. (03 Marks)

Ans. A grammer G is 4 tuple or quadruple G = (V,T,P,S) where 'V' is variable , T is terminals, P is production and 'S' is start symbol.

Ex: $S \to \varepsilon$, $S \to aS$

$A \Rightarrow \alpha\ \beta\ r$, the process of obtaining strings of terminal s and / or non - terminals from the start symbol by applying some or all productions is called derivation.

$E \Rightarrow E + E$, $E \Rightarrow id + E$, $E \Rightarrow id + id$

Let G = (V,T,P,S) be a grammar. The string ω obtained from the grammar G such that $S \Rightarrow w$ is called sentence of grammar G. Here, w is the string of terminals.

b. What is CNF? Obtain the following grammar in CNF (09 Marks)

$S \to ASB \mid \varepsilon$
$A \to aAS \mid a$
$B \to SbS \mid A \mid bb$

Ans. Let G = (V,T,P,S) be a CFG. The grammr G is said to be in CNF if all productions are of the form.

$A \to BC$ or $A \to a$

Eliminate ε - production

0v	nv	Production
ϕ	$S \to \varepsilon$	$S \to \varepsilon$
S	S	–

V = {A,B} are nullable variables

Production	Resulting production (p^1)
S → A S B	S → AB
A → aAS	A → aA | a
B → SbS	B → SbS| bS|Sb| b|A|bb

Given Production	Action	
S → AB	Already in CNF	S → AB
A → aA	Replace a by A_0 $A_0 \to A$	$A \to A_0 A$ $A_0 \to a$ $A \to a$
B → SbS|bS|Sb|A	Replace b by B_0 $B_0 \to b$	$B \to SB_0 S \mid B_0 S \mid SB_0 \mid a$

Replace BoS with B_1

$B \to SB_1$

$B_1 \to B_0 S$

$G^1 = (V^1, T, P^1, S)$ 'S' is start symbol

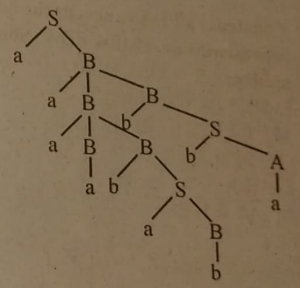
$V = (S, A, B, B_0, B_1, A_0)$

P = {
$S \to AB$
$A \to A_0 A$
$A_0 \to a$
$A \to a$
$B \to SB_1 \mid B_0 S \mid SB_0 \mid a \mid$
$B_0 \to b$
$B_1 \to B_0 S$
}

c. Let G be the grammar,
$S \to aB \mid bA$
$A \to a \mid aS \mid bAA$
$B \to b \mid bS \mid aBB$
For the string qaabbabbba find a
(i) Left most derivation.
(ii) Right most derivation.
(iii) Parse tree.

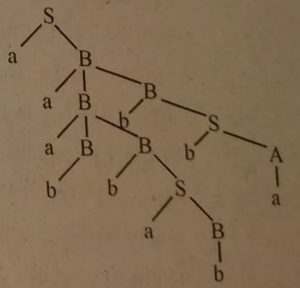
Ans. i. Let most derivation **(04 Marks)**

$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aaaBBB$
$\Rightarrow aaaabbSB$
$\Rightarrow aaaabbaBB$
$\Rightarrow aaaabbabB$
$\Rightarrow aaaabbabbS$
$\Rightarrow aaaabbabbbA$
$\Rightarrow aaaabbabbba$



ii. Right most derivation

$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aaaBbS$
$\Rightarrow aaaBbbA$
$\Rightarrow aaaBbba$
$\Rightarrow aaaaBBbba$
$\Rightarrow aaaaBbSbba$
$\Rightarrow aaaBbaBbba$
$\Rightarrow aaaaBbabbba$
$\Rightarrow aaaabbabbba$



OR

6. a. **Explain the following terms:**
(i) Pushdown automata (PDA).
(ii) Languages of a PDA.
(iii) Instantaneous description of a PDA. **(03 Marks)**

Ans. **i. Pushdown Automata (PDA) :** A PDA is a seven tuple
$M = (Q, \Sigma, |, \delta, q_0, Z_0, F)$
Q is set of finite states
Σ is set of input alphabets
| is set of stack alphabets
δ is transition $Q \times (\Sigma \cup \varepsilon) \times | - Q \times |^*$
$q_0 \varepsilon Q$ is start state
$Z_0 \varepsilon |$ is initial symbol on stack
$F \leq Q$ is set of final state
ii. Language of PDA : The language LCM accepted by a final state is defined as
$L(M) = \{w \mid (q_0, w, Z_0) \mid^* (P, \varepsilon, \alpha)\}$
iii. Instantaneous description : Let $M = (Q, \Sigma, |, \delta, q_0, Z_0, F)$ be a PDA. An ID (instantaneous description) is defined as 3-tuple or a triple (q, w, α)

b. **Construct a PDA to accept the language $L = \{ww^R | w \varepsilon \{a,b\}^*\}$. Draw the graphical representation of this PDA. Show the moves made by this PDA for the string aabbaa.** **(10 Marks)**

Ans. $L(M) = \{ww^R | w \varepsilon \{a,b)^*\}$
$M = (Q, \Sigma, |, \delta, q_0, Z_0, F\}$
$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a,b\}$
$| = \{a,b,Z_0\}$
δ : {

$$(q_0, \varepsilon, Z_0) = (q_{11} Z_0)$$
$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$
$$\delta(q_0, b, a) = \{(q_0, aa), (q, \varepsilon)\}$$
$$\delta(q_0, a, b) = (q_0, ba)$$
$$\delta(q_0, a, b) = (q_0, ab)$$
$$\delta(q_0, a, b) = (q_0, ab)$$
$$\delta(q_0, b, b) = \{(q_0, bb), (q, \varepsilon)\}$$
$$\delta(q_1, a, a) = (q_1, \varepsilon)$$
$$\delta(q_1, b, b) = (q_1, \varepsilon)$$
$$\delta(q_1, \varepsilon, Z_0) = (q_0, Z_0)$$

}

$q_0 \in Q$ is start state
$Z_0 \in P$ is initial stack symbol
$F = \{q_2\}$ is final state

$(q_0, aabbaa, z_0)$
$\downarrow$
$(q_0, abbaa, az_0) \rightarrow (q_0, aabbaa, z_0) \rightarrow (q_2, aabbaa, z_0)$
$\downarrow$
$(q_0, bbaa, aaz_0) \rightarrow (q_1, bbaa, z_0) \rightarrow (q_2, bbaa, z_0)$
$\downarrow$
$(q_0, baa, baaz_0) \rightarrow (q_0, aa, bbaaz_0) \rightarrow (q_0, a, abbaaz_0)$
$\downarrow$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\downarrow$
$(q_1, aa, aaz_0) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (q, \varepsilon, bbaaz_0)$
$\downarrow$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\downarrow$
$(q_1, a, az_0) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (q_0, \varepsilon, abbaaz_0)$
$\downarrow$
$(q_1, \varepsilon, z_0) \rightarrow (q_2, \varepsilon, z_0)$

c. **Convert the following CFG to PDA**
$S \rightarrow aABB | aAA$
$A \rightarrow aBB | a$
$B \rightarrow bBB | A$
$C \rightarrow a$

Ans. $Q = \{q_0, q_1, q_2\}$ **(03 Marks)**
$\Sigma = \{a, b\}$
$\Gamma = \{S, A, B, C_1, Z_0\}$
δ : {

$$\delta(q_0, \varepsilon, Z_0) = (q_1, SZ_0)$$
$$\delta(q_1, a, S) = (q_1, ABB)$$
$$\delta(q_1, a, S) = (q_1, AA)$$
$$\delta(q_1, a, A) = (q_1, BB)$$
$$\delta(q_1, a, A) = (q_1, \varepsilon)$$
$$\delta(q_1, b, B) = (q_1, BB)$$
$$\delta(q_1, a, B) = (q_1, BB)$$
$$\delta(q_1, a, B) = (q_1, \varepsilon)$$
$$\delta(q_1, a, c) = (q_1, \varepsilon)$$
$$\delta(q_1, \varepsilon, Z_0) = (q_f, Z_0)$$

}

$q_0 \in Q$ is start state
$Z_0 \in T$ is stack symbol
$F = \{q_f\}$ is final state

Module-4

7. a **If L_1 and L_2 are context free languages then prove that L,UL_2, $L_1 L_2$ and L_1^*, are context free languages.** **(04 Marks)**

Ans.

(i) $G_1 = (V_1, T_1, P_1, S_1)$

$G_2 = (V_2, T_2, P_2, S_2)$

$G_3 = (V_1 \cup V_2 \cup S_3, T_1 \cup T_2, P_3, S_3)$

S_3 is a start state G_3 and $S_3 \in (V_1 \cup V_2)$

$P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1 / S_2\}$

$L_3 = L_1 \cup L_2$

(ii) $G_4 = (V_1 \cup V_2 \cup S_4, T_1 \cup T_2, P_4, S_4)$

S_4 is a start symbol for the grammar G_4 and $S_4 \in (V_1 \cup V_2)$

$P_4 = P_1 \cup P_2 \cup \{S_4 \to S_1 S_2\}$

$\therefore L_3 = L_1 . L_2$

(iii) $G_5 = (V, US_5, T_1, P_5, S_5)$

S_5 is a the start symbol of Grammar G_5

$P_5 = P_1 \cup \{S_5 \to S_1 S_5 | \varepsilon\}$

$T L_5 = L_5^*$

b. **Give a decision procedure to answer each of the following questions:**
(i) Given a regular expression a and a PDA M, the language accepted by M a subset of the language generated by a?
(ii) Given a context-free Grammar G and two strings Si and S2, does G generate $S_1 S_2$?
(iii) Given a context free Grammar G, does G generate any even length strings.
(iv) Given a Regular Grammar G, is L(G) context-free? **(12 Marks)**

Ans. i. Observe that this is true if $\perp(M) \cap L(\alpha) = \phi$. So the following procedure answers the question :

1. From α, build a PDA M* so that $L(M^*) = L(\alpha)$
2. From M and M*, build a PDA M** that accepts $L(M) \cap L(M^*)$
3. If L(M**) is empty , return true else return false.

ii. 1. Convert G to chomsky normal forms2. Try all derivations in G of length up to $2|S_1 S_2|$. If any of them generates $S_1 S_2$ return True, else return false

iii. 1. Use CFG to PDA topolown (G) to build a PDA P hta taccepts L(G).
2. Build an FSM E that accepts all even length strings over the alphabet Σ_G.
3. Use insert PDA and FSM(P,E) to build a PDA P* that accepts $L(G) \cap L(E)$.
4. Return decioleCFLempty(P*)

iv. i. Return True (Since every regular language is context free)

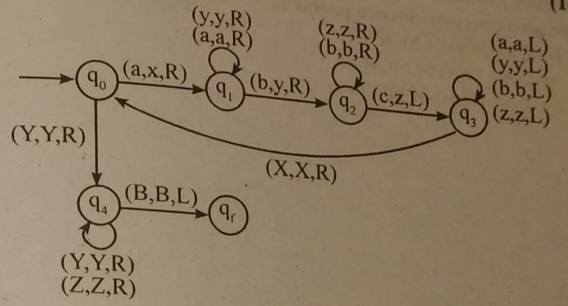
OR

8. a. **Explain with neat diagram, the working of a Turing Machine model. (05 Marks)**

Ans. Refer Q.no.9(a) of MQP - 2.

b. **Design a Turing machine to accept the language L= $\{a^n b^n c^n | n >= 1\}$. Draw the transition diagram. Show the moves made by this turing machine for the string aabbcc.** **(11 Marks)**

Ans.



a a b b c c
x a b b c c
x a b b c c
x a y b c c
x a y b c c
x a y b z c
x x y b z c
x x y y z c
x x y y z z

Module-5

9 **Write short notes on:**
a. Multi-tape turning machine.
b. Non-deterministic turning machine.
c. Linear Bounded automata. **(16 Marks)**

Ans. a. Refer Q.no. 9(b) of MQP - 1.

b. Non - deterministic turning machine : In a non - deterministic turning machine, for every state and symbol, there are a group of actions the TM can have. So here the transitions are not deterministic. The computation of a non - deterministic turning machine is a tree of configurations that can be reached from the start configuration. An input is accepted if there is at least one node of the tree which is an accept configuration, otherwise it is not accepted. If all branches of the computational tree

hact on all inputs, the non - deterministic turning machine is called a decideg and if for some input, all branches are rejected, the input is also rejected.

c. Linear bounded automata : Refer Q.no. 10(b) of MQP - 1.

OR

10. **Write short notes on:**
    a. Undecidable languages.
    b. Halting problem of turning machine.
    c. The post correspondence problem. **(16 Marks)**

Ans. a. Refer Q.no.9(b) of MQP - 2
b. Refer Q.no.10(a) of MQP - 1
c. Refer Q.no.10b(i) of MQP - 2.

Fifth Semester B.E. Degree Examination, CBCS - June / July 2018
Automata Theory & Compatibility

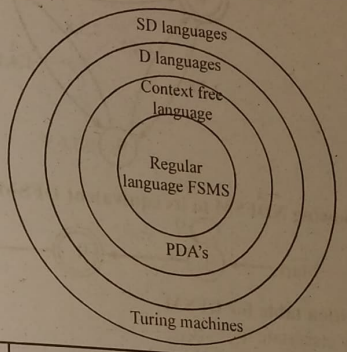
Time: 3 hrs. **Max. Marks: 80**

Note : *Answer any FIVE full questions, selecting ONE full question from each module.*

Module - 1

1. a. **With a neat diagram, explain a hierarchy of language classes in automata theory.** **(04 Marks)**

Ans.



Grammar	Language	Automaton
Type - 0	Recursively enumerable	Turing machine
Type - 7	Context sensitive	Linear bounded Non - deterministic Turing machine
Type - 2	Context free	Non determinstic pushdown automata
Type - 3	Regular	finite state automaton

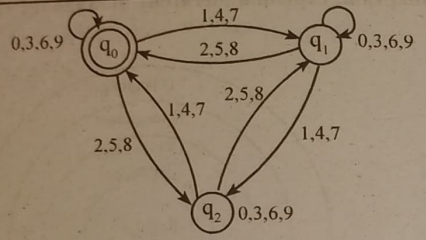
b. **Define deterministic FSM. Draw a DFSM to accept decimal strings which are divisible by 3.** **(06 Marks)**

Ans. Step 1 :- d = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} K = 3
Step 2 :- After dividing by 3, possible reminder are 0, 1, 2
Step 3 :- Compute transition

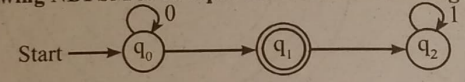
$$\delta(q_i, a) = q_j \quad \text{where } j = (r * i + d) \bmod K$$

wih r = 10 and K = 3
{0, 3, 6, 9} leaves 0 as reminder)
{1, 4, 7} leaves 1 as reminder)
{2, 5, 8} leaves 2 as reminder)

i = 0	0	(10 * 0 + 0) Mod 3 = 0	$\delta(q_0, 0) = q_0$	$\delta(q_0, \{0, 3, 6, 9\}) = q_0$
	1	(10 * 0 + 1) Mod 3 = 1	$\delta(q_0, 1) = q_1$	$\delta(q_0, \{1, 4, 7\}) = q_1$
	2	(10 * 0 + 2) Mod 3 = 2	$\delta(q_0, 2) = q_2$	$\delta(q_0, \{2, 5, 8\}) = q_2$
i = 1	0	(10 * 1 + 0) Mod 3 = 1	$\delta(q_1, 0) = q_1$	$\delta(q_1, \{0, 3, 6, 9\}) = q_1$
	1	(10 * 1 + 1) Mod 3 = 2	$\delta(q_1, 1) = q_2$	$\delta(q_1, \{1, 4, 7\}) = q_2$
	2	(10 * 1 + 2) Mod 3 = 0	$\delta(q_1, 2) = q_0$	$\delta(q_1, \{2, 5, 8\}) = q_0$
i = 2	0	(10 * 2 + 0) Mod 3 = 2	$\delta(q_2, 0) = q_2$	$\delta(q_2, \{0, 3, 6, 9\}) = q_2$
	1	(10 * 2 + 1) Mod 3 = 1	$\delta(q_2, 1) = q_0$	$\delta(q_2, \{1, 4, 7\}) = q_0$
	2	(10 * 2 + 2) Mod 3 = 0	$\delta(q_2, 2) = q_1$	$\delta(q_2, \{2, 5, 8\}) = q_1$



c. Convert the following NDFSM to its equivalent DFSM Rfer Fig 1.c.



Also write transition table for DFSM **(06 Marks)**

Ans. **Step 1 : -** Identify start state $Q_D = \{q_0\}$

Step 2 :- Identify alphabet $\Sigma = \{0, 1\}$

Step 3 :- Transitions

Input symbol = 0

$\delta_D = (\{q_0\}, 0) = \delta_N (\{q_0\}, 0) = \{q_0, q_1\}$

For state $\{q_0, q_1\}$

$\delta_D = (\{q_0, q_1\}, 0) = \delta_N (\{q_0, q_1\}, 0)$

$= \delta_N \{q_0, 0\} \cup \delta_N(q_1, 0)$

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

For state $\{q_1\}$

Input Symbol = 0

$\delta_D = (\{q_1\}, 0) = \delta_N (\{q_1\}, 0)$

For state $\{q_0, q_1, q_2\}$

Input Symbol = 0

$\delta_D = (\{q_0, q_1, q_2\}, 0) = \delta_N (\{q_0, q_1, q_2\}, 0)$

$= \delta_N \{q_0, 0\} \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_0, q_1\} \cup \{q_2\} \{\phi\} = \{q_0, q_1, q_2\}$

For state $\{q_1, q_2\}$

$\delta_D = (\{q_1, q_2\}, 0) = \delta_N (\{q_1, q_2\}, 0)$

$= \delta_N \{q_1, 0\} \cup \delta_N(q_2, 0)$

$= \{q_2\} \cup \phi = \{q_2\}$

For state $\{q_2\}$

Input Symbol = 0

$\delta_D = (\{q_2\}, 0) = \delta_N (\{q_2\}, 0) = \{\phi\}$

Input symbol = 1

$\delta_D = (\{q_0\}, 1) = \delta_N (\{q_0\}, 1) = \{q_1\}$

$\delta_D = (\{q_0, q_1\}, 1) = \delta_N (\{q_0, q_1\}, 1)$

$= \delta_N \{q_0, 1\} \cup \delta_N(q_1, 1\}$

$= \{q_1\} \cup \{q_2\} = \{q_1, q_2\}$

Input Symbol = 1

$\delta_D = (\{q_1\}, 1) = \delta_N (\{q_1\}, 1)$

Input Symbol = 1

$\delta_D = (\{q_0, q_1, q_2\}, 1) = \delta_N (\{q_0, q_1, q_2\}, 1)$

$= \delta_N \{q_0, 1) \cup \delta_N(q_1, 1\} \cup \delta_N(q_2, 1\}$

$= \{q_1\} \cup \{q_2\} \cup \{q_2\} = \{q_1, q_2\}$

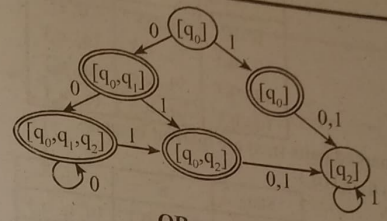
$\delta_D = (\{q_0, q_1\}, 1) = \delta_N (\{q_0, q_1\}, 1)$

$= \delta_N \{q_1, 1) \cup \delta_N(q_2, 1\}$

$= \{q_2\} \cup \{q_2\} = \{q_2\}$

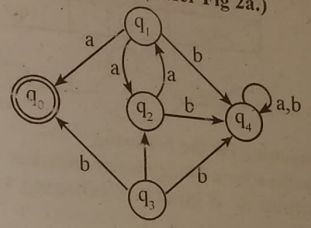
Input Symbol = 1

$\delta_D = (\{q_2\}, 1) = \delta_N (\{q_2\}, 1) = \{q_2\}$



OR

2. a. Minimize the following finite automata (Refer Fig 2a.) **(06 Marks)**



Ans. **Step 1 :-**

q_1				
q_2				
q_3				
q_4	X	X	X	X
	q_0	q_1	q_2	q_3

Step 2 :-

δ	a	b
(p, q)	(r, s)	(r, s)
(q_0, q_1)	(q_1, q_2)	(q_3, q_4)
(q_0, q_2)	(q_1, q_1)	(q_3, q_4)
(q_0, q_3)	(q_1, q_2)	(q_3, q_4)
(q_1, q_2)	(q_2, q_1)	(q_4, q_4)
(q_1, q_3)	(q_2, q_2)	(q_4, q_4)
(q_2, q_3)	(q_1, q_2)	(q_4, q_4)

q_1	X			
q_2	X			
q_3	X			
q_4	X	X	X	X
	q_0	q_1	q_2	q_3

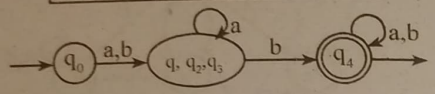
Step 3 :-

δ	a	b
(p, q)	(r, s)	(r, s)
(q_1, q_2)	(q_2, q_1)	(q_4, q_4)
(q_1, q_3)	(q_2, q_2)	(q_4, q_4)
(q_2, q_3)	(q_1, q_2)	(q_4, q_3)

None of the unmarked pairs (r, s) are marked in table.

Step 4 :- (q_0) (q_1 q_2 q_3) (q_4)

State	a	b
q_0	(q_1 q_2 q_3)	(q_1 q_2 q_3)
(q_1 q_2 q_3)	(q_1 q_2 q_3)	(q_4)
(q_4)	. (q_4)	(q_4)



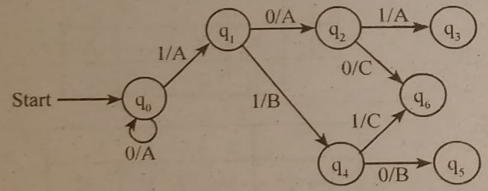
b. Construct a mealy machine for the following

i) Design a mealy machine for a binary input sequence. Such that if it has substring 101, the machine outputs A. if input has substring 110, the machine outputs. B otherwise it outputs C.

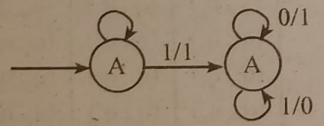
ii) Design a mealy machine that takes binary number as input and produces 2's complement of that number as input.

Assume the string is read from LSB to MSB and end carry is discarded. (06 Marks)

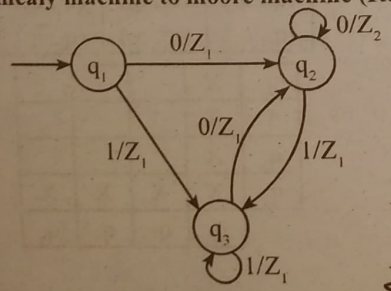
Ans. i)



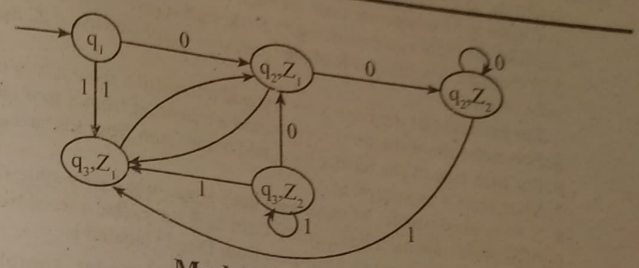
ii)



c. Convert the following mealy machine to moore machine (Refer fig 2.c.) (04 Marks)



Ans.



Module - 2

3. a. Define regular expression. Obtain a regular expression for the following language:

i) L = {$a^n b^m$ | m + n is even}

ii) L = {$a^n b^m$ | m ≥ 1 n ≥ 1, ṅm ≥ 3}

iii) L = { W : |W| mod 3 = 0 where W ∈ {a, b}*}

Ans. Definition :- Refer Q.No. 3.a. of MQP - 1 **(08 Marks)**

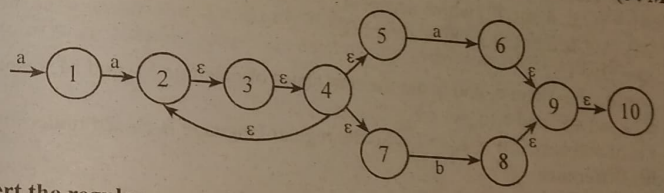
i) R∈ = ((a+b) (a + b) * or R∈ =(a a) * (b b) * + a(a a) * b(b b)*

ii) R∈ = aaaa*b + abbbb* + aaa* bbb*

iii) R∈ = ((a + b) (a + b) (a + b))*

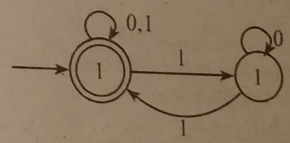
b. Design an NDFSM that accept the language L(aa*(a + b))

Ans. **(04 Marks)**



c. Convert the regular expression (0 + 1) * 1(0 + 1) to NDFSM

Ans. **(04 Marks)**



OR

4. a. If the regular grammars define exactly the regular language, then prove that the class of languages that can be defined with regular grammars is exactly the regular languages. **(04 Marks)**

Ans. We first show that any languages that can be defined with a regular grammar can be accepted by some FSM and so is regular. Then we must show that every regular language can be defined with a regular grammar. Both proofs are by construction.

Regular grammar → FSM : The following algorithm construct an FSM M from a regular grammar G = (V, Σ, R,, S) and assures that LCM) = LCG) :

grammar to FSM (G : regular grammar) =
1. Create in M a separate state for each non terminal in V.
2. Make the state corresponding to S the start state.
3. If there are any rules in R of the form $x \rightarrow w$, for some $w \in \Sigma$, then create an additional state labeled #.
4. For each rule of the form $x \rightarrow wy$, add a transition from z to y labeled w.
5. For each rule of the form $x \rightarrow w$, add a transition from x to # labeled w.
6. For each rule of the form $x \rightarrow \varepsilon$, add a transition from x as accepting.
7. Mark state # as accepting.
8. If M is in complete, M requires a dead state. Add a new state D. For every (q, i) pair for which no transition has already been defined, create a transition from q to D labeled i. for every i in Σ, create a transition from D to D labeled i.

b. Prove that the regular language are closed under complement, intersection, difference, reverse and letter substitution. **(08 Marks)**

Ans. **i) Under complement :-** Let M1 = (Q, Σ, δ, q_0, F) be a DFA which accepts the language
1. Since the language is accepted by a DFA, the language is regular. Noe let us define the machine M_2 = (Q, Σ, δ, q_0, Q - F) which accepts I. Note that there in no difference between M_1 and M_2 except the final states.
The non - final states of M_1 = are th final state of M_2 and final stat of M_1 are the non final states of M_2 so the language which is rejected by M_1 is accepted by M_2 and vice versa. Thus we have a machine M_2 which accepts all those strings denoted by I that are rejected by machine M_1. So regular language is closed under complement.

ii) Intersection :-
$M_1 = (Q, \Sigma, \delta_1, q_1, F_1)$ which accepts L_1
$M_2 = (Q, \Sigma, \delta_2, q_2, F_2)$ which accepts L_2
$Q = Q_1 \times Q_2$
$q = (q_1, q_2)$ where q_1 and q_2 are the start states of machine M_1 and M_2 respectively.
$\hat{\delta}_1(q_1, w) \in F_1$ and $\hat{\delta}_2(q_2, w) \in F_2$.
i.e.,.if and only if $w \in L_1 \cap L_2$. So the regular language is closed under intersection.

iii) Difference
$M_1 = (Q, \Sigma, \delta_1, q_1, F_1) \rightarrow L_1$
$M_2 = (Q, \Sigma, \delta_2, q_2, F_2) \rightarrow L_2$
$\hat{\delta}_1(q_1, q_2), w)$ is in F
$(\hat{\delta}_1(q_1, w) \in F_1$ and $(\hat{\delta}_2(q_2, w) \in F_2$
i.e., regular language is close under difference.

iv) Reversal and letter substitution
$L(E^R) = (L(E))^R$
Refer Q.No. 3b. of MQP - 2.

c. State and prove pumping lemma for regular language. **(04 Marks)**

Ans. Refer Q.No. 3b. of MQP - 1.

Module - 3

5. a. Define a context free grammar. Obtain the grammar to generate the language L= {W} $n_a(w) = n_b(w)$} **(04 Marks)**

Ans. $S \rightarrow \varepsilon$
$S \rightarrow a\ s\ b$

$S \rightarrow b\ s\ a$
$G = (V, J, P, S)$
$V = \{S\}$
$T = \{a, b\}$
$P = \{\delta \rightarrow \varepsilon$
$\delta \rightarrow a\ s\ b$
$\delta \rightarrow b\ s\ a$
$\}$
S is start symbol

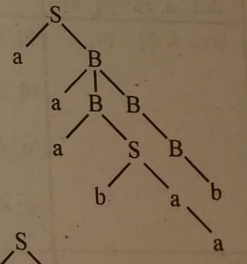
b. For the regular expression (011 + 1)* (01)* obtain the context free grammar. **(04 Marks)**

Ans. $G = (V, T, P, S)$
$V = \{S, A\}$
$T = \{0, 1\}$
$P = \{$
$S \rightarrow \varepsilon \mid 0\ 1\ 1\ A \mid 0\ 1\ S$
$A \rightarrow 1\ S \mid \varepsilon$
$\}$
's' is start symbol.

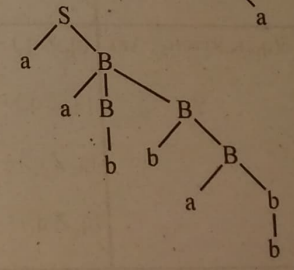
c. What is ambiguity? Show that the following grammar is ambigugous.
$S \rightarrow aB \mid bA$
$A \rightarrow as \mid bAA \mid a$
$B \rightarrow bs \mid aBB \mid b.$ **(08 Marks)**

Ans. A grammar G is ambiguous if and only if there exists atleast one string $w \in T$ * ϕορ which two or more different parse trees exist by applying either LMD or RMD.

$S \Rightarrow aB$
$S \Rightarrow aaBB$
$S \Rightarrow aabSB$
$S \Rightarrow aabbAB$
$S \Rightarrow aabbaB$
$S \Rightarrow aabbab$



$S \Rightarrow aB$
$S \Rightarrow aaBB$
$S \Rightarrow aabB$
$S \Rightarrow aabbS$
$S \Rightarrow aabbaB$
$S \Rightarrow aabbab$



Two parse tree for same string
So grammar is ambiguous.

OR

6. a. **Define PDA. Obtain to accept the language 1(M) = {w C w^R) | w ∈ (a, b)8}, where w^R is reverse of w by a final state.** **(08 Marks)**

Ans. Refer Q.No. 6.. of MQP - 1

b. **For the grammar**
S → aABB | aAA
A → aBB | a
B → bBB | A
C → a
Obtain the corresponding PDA. **(04 Marks)**

Ans. Refer Q.No. 6.c. of Dec 2017 / Jan 2018

c. **Obtain a CFG for the FDA shown below:**
$f(q_0, a, Z) = (q_0, AZ)$
$f(q_0, a, A) = (q_0, A)$
$f(q_0, b, A) = (q, \varepsilon)$
$f(q_1, \varepsilon, Z) = (q_2, \varepsilon)$. **(04 Marks)**

Ans.

For δ of the form $\delta(q_1, a, z) = (q_1, \varepsilon)$	Resulting production $(q_1 Z q_1) \rightarrow a$
$\delta(q_0, a, A) = (q_3, \varepsilon)$ $\delta(q_0, b, A) = (q_1, \varepsilon)$ $\delta(q_1, \varepsilon, z) = (q_2, \varepsilon)$	$(q_0 A q_3) \rightarrow a$ $(q_0 A q_1) \rightarrow b$ $(q_1 Z q_2) \rightarrow \varepsilon$

For δ of the form $\delta(q_1, a, z) = (q_1, \varepsilon)$	Resulting production $(q_1 Z q_1) \rightarrow a$
$\delta(q_0, a, A) = (q_3, \varepsilon)$	$(q_0 Z q_0) \rightarrow a(q_0 A q_0)(q_0 Z q_0) \mid a(q_0 A q_1)(q_1 Z q_0) \mid a(q_0 A q_2)(q_2 Z q_0) \mid a(q_0 A q_3)(q_3 Z q_0)$ $(q_0 Z q_1) \rightarrow a(q_0 A q_0)(q_0 Z q_1) \mid a(q_0 A q_1)(q_1 Z q_1) \mid a(q_0 A q_2)(q_2 Z q_1) \mid a(q_0 A q_3)(q_3 Z q_1)$ $(q_0 Z q_2) \rightarrow a(q_0 A q_0)(q_0 Z q_2) \mid a(q_0 A q_1)(q_1 Z q_2) \mid a(q_0 A q_2)(q_2 Z q_2) \mid a(q_0 A q_3)(q_3 Z q_2)$ $(q_0 Z q_3) \rightarrow a(q_0 A q_0)(q_0 Z q_3) \mid a(q_0 A q_1)(q_1 Z q_3) \mid a(q_0 A q_2)(q_2 Z q_3) \mid a(q_0 A q_3)(q_3 Z q_3)$
$\delta(q_3, \varepsilon, z) = (q_0, Az)$	$(q_3 Z q_0) \rightarrow (q_0 A q_0)(q_0 Z q_0) \mid (q_0 A q_1)(q_1 Z q_0) \mid (q_0 A q_2)(q_2 Z q_0) \mid (q_0 A q_3)(q_3 Z q_0)$ $(q_3 Z q_1) \rightarrow (q_0 A q_0)(q_0 Z q_1) \mid (q_0 A q_1)(q_1 Z q_1) \mid (q_0 A q_2)(q_2 Z q_1) \mid (q_0 A q_3)(q_3 Z q_1)$ $(q_3 Z q_2) \rightarrow (q_0 A q_0)(q_0 Z q_2) \mid (q_0 A q_1)(q_1 Z q_2) \mid (q_0 A q_2)(q_2 Z q_2) \mid (q_0 A q_3)(q_3 Z q_2)$ $(q_3 Z q_3) \rightarrow (q_0 A q_0)(q_0 Z q_3) \mid (q_0 A q_1)(q_1 Z q_3) \mid (q_0 A q_2)(q_2 Z q_3) \mid (q_0 A q_3)(q_3 Z q_3)$

Module-4

7. a. **Consider the grammar**
S → 0A|1B
A → OAA|1S|1
B → 1BB|0S|0
Obtain the grammar in CNF.

Ans. A → 1
B → 0 **(08 Marks)**

Given production	Action	Resulting production
S → 0A | 1B	Replace 0 by B_0 $B_0 \rightarrow 0$ Replace 1 by B_1 $B_1 \rightarrow 1$	S → B_0A | B_1B $B_0 \rightarrow 0$ $B_1 \rightarrow 1$
A → 0AA | 1S	Replace 0 by B_0 $B_0 \rightarrow 0$ Replace 1 by B_1 $B_1 \rightarrow 1$	A → B_0AA | B_1S $B_0 \rightarrow 0$ $B_1 \rightarrow 1$
B → 1BB | 0S	Replace 0 by B_0 $B_0 \rightarrow 0$ Replace 1 by B_1 $B_1 \rightarrow 1$	B → B_1BB | B_0S $B_1 \rightarrow 1$ $B_0 \rightarrow 0$

Consider $A \rightarrow B_0AA$ and $B \rightarrow B_1BB$

$A \rightarrow B_0AA \Rightarrow A \rightarrow B_0D_1$
$D_1 \rightarrow AA$

$B \rightarrow B_1BB \Rightarrow B \rightarrow B_1D_2$
$D_2 \rightarrow BB$

G' = (V', T, P', S) is in CNF where
$V_1 = \{S, A, B, B_0, B_1, D_1, D_2\}$
$T_1 = \{0, 1\}$
$P_1 = \{$

$S \rightarrow B_0A \mid B_1B$
$A \rightarrow B_1S \mid 1 \mid B_0D_1$
$B \rightarrow B_0S \mid 1 \mid B_1D_2$
$B_0 \rightarrow 0$
$B_1 \rightarrow 1$
$D_1 \rightarrow AA$
$D_2 \rightarrow BB$
$\}$

'S' is the start symbol

b. **Show that $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context free.** **(08 Marks)**

Ans. Refer Q.No. 8.b. of MQP - 2

OR

8. a. **With a neat diagram, explain the working of a basic Turing machine.** **(04 Marks)**

Ans. Refer Q.No. 9.a. of MQP - 2

b. **Obtain a Turing machine to accept the language $L = \{0^n1^n \mid n \geq 1\}$.** **(08 Marks)**

Ans. Refer Q.No. 9.a. of MQP - 1

c. **Briefly explain the techniques for TM construction.** **(04 Marks)**

Ans. Refer Q.No. 9.b.(i) of MQP - 1

Module-5

9. a. **Obtain a Turing machine to recognize the language $L = \{0^n1^n2^n \mid n \geq 1\}$.** **(08 Marks)**

Ans.

States	0	1	2	Z	Y	X	B
q_0	q_1, X, R				q_4, Y, R		
q_1	q_1, 0, R	q_2, Y, R			q_1, Y, R		
q_2		q_2, 1, R	q_3, Z, L	q_3, Z, R			
q_3	q_3, 0, L	q_3, 1, L		q_3, Z, L	q_3, Y, L	q_0, X, R	
q_4				q_5, Z, R	q_4, Y, R		
q_5				q_5, Z, R			q_6, B, R
q_6							

b. **Prove that $HALT_{TM}$ ={(M,W)| the Turing machine M halts on input W} is undecidable.** **(04 Marks)**

Ans. Refer Q.No. 10.a. of MQP - 1

c. **With example, explain the quantum computation.** **(04 Marks)**

Ans. Refer Q.No. 10.b.(ii) of MQP - 2

OR

10 **Write a short note on:**

a. **Multiple Turing machine**

b. **Non deterministic Turing machine**

c. **The model of linear bounded automaton**

d. **The post correspondence problem.** **(16 Marks)**

Ans. a) Refer Q.No. 9.b.(i) of MQP - 1

b) Refer Q.No. 9.b. of Dec 2017 / Jan 2018

c) Refer Q.No. 10.b. of MQP - 1

d) Refer Q.No. 10.b. of MQP - 2

Fifth Semester B.E. Degree Examination, CBCS - Dec 2018 / Jan 2019

Automata Theory and Compatibility

Time: 3 hrs. Max. Marks: 80

Note : Answer any FIVE full questions, selecting ONE full question from each module.

Module - 1

1. a. **Define the following with example :**
i) String ii) Language iii) Alphabet iv) DFSM. **(08 Marks)**

Ans. i) The sequence of symbols obtained from the alphabets of a language is calle a string. Formally, a string is defined as a finite sequence of symbols from the alphabet Σ. Sequence of symbols from the alphabet Σ.

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

ii) A language can be defined as a set of strings obtained from Σ^* where Σ is set of alphabet of a particular language. In others words, a language is subset of Σ^* which is denoted by $L \subseteq \Sigma^*$.

Ex : {ε , 0, 1, 01, 10, 1100, 0011 ....}

iii) A language consist of various symbols from which the words, statements etc., can be obtained. These symbols are called alphabets.

Ex : Σ = {a , b , ...,z,A,B,C,....,Z,#,{, }, (,).....0 , ...1}

iv) Deterministic Finite Automata (DFSM) is 5 - tuple or quintuple indicating five components $M = (Q, \Sigma, \delta, q_0, F)$

Where M is the name of machine

Q is non - empty finite set of states

Σ is non - empty finite set of input alphabets

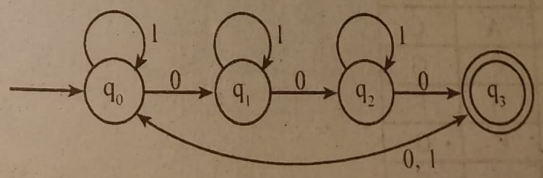
δ is transition function $Q \times \Sigma \rightarrow Q$

$q_0 \in Q$ is start state

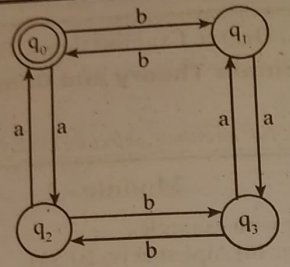
$F \subseteq Q$ is accepting or final states

b. **Design a DFSM to accept each of the following languages :**
i) $L = \{W \in \{0, 1\}^* : W$ has 001 as a substring$\}$
ii) $L = \{W \in \{a, b\}^* : W$ has even number of a's and even number of b's$\}$.
(08 Marks)

Ans. i)

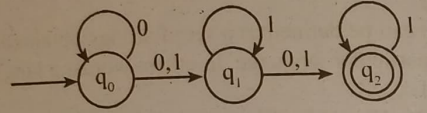


ii)



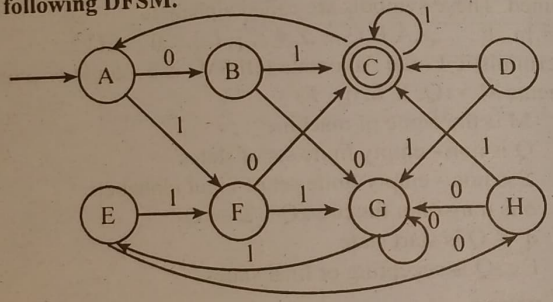
OR

2. a. Define NDFSM. Convert the following NDFSM to its equivalent DFSM. **(08 Marks)**



Ans. Refer Q.no.1(c) of June/July 2018.

b. Minimize the following DFSM. **(08 Marks)**



Ans.

	0	1
→A	B	F
B	G	C
*C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

Step 1 :

B							
C	X	X					
D		X					
E		X					
F		X					
G		X					
H		X					
	A	B	C	D	E	F	G

δ	a	b
(A,B)	(B,G)	(F,C)
(A,D)	(B,C)	(F,G)
(A,E)	(B,H)	(F,F)
A,F)	(B,C)	(F,G)
(A,G)	(B,G)	(F,E)
(A,H)	(B,G)	(F,C)
(B,D)	(G,C)	(C,G)
(B,E)	(G,H)	(C,F)
(B,F)	(G,C)	(C,G)
(B,G)	(G,C)	(C,E)
(B,H)	(G,G)	(C,C)
(D,E)	(C,H)	(G,F)
(D,F)	(C,C)	(G,G)
(D,G)	(C,G)	(G,E)
(D,H)	(C,G)	(G,C)
(E,F)	(H,C)	(F,G)
(E,G)	(H,G)	(F,E)
(E,H)	(H,G)	(F,C)
(F,G)	(C,G)	(G,E)
(F,H)	(G,G)	(E,C)
(G,H)	(G,G)	(E,C)

Step 2 :

B	X						
C	X	X					
D	X	X	X				
E		X	X	X			
F	X	X	X		X		
G		X	X	X		X	
H	X	X	X	X	X	X	X
	A	B	C	D	E	F	G

(A,E)	(B,H)	(F,F)
(A,G)	(B,G)	(F,E)
(B,H)	(G,G)	(C,C)
(D.F)	(C,C)	(G,G)
(E,G)	(H,G)	(F,E)

Indistinguishable pairs : (A,E) , (,H) & (D,F)
Distinguishable pairs : C & G

Minimize DFA :

Step 1 :

(A,E) , (B,H) & (D,F) Indistinguishable pair C,G distinguishable pair.

Step 2 :

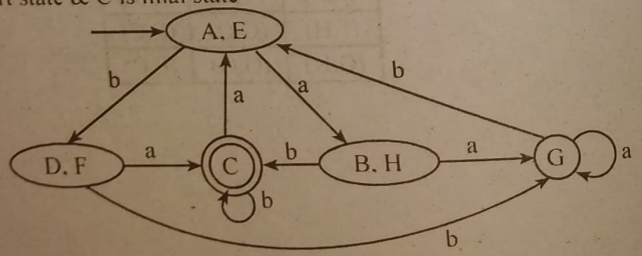
States in minimized DFA
(A,E), (B,H), C; (D,F) , G

Step 3 :

δ	0	1
(A,E)	(B,H)	(D,F)
(B,H)	G	C
C	(A,E)	C
(D,F)	C	G
G	G	(A.E)

Step 4 :

(A.E) is start state & C is final state

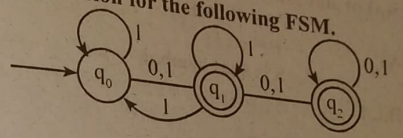


Module - 2

3. a. **Define Regular expression and write Regular expression for the following language.**
i) $L = \{a^{2n} b^{2m} \mid n \geq 0, m \geq 0\}$
ii) $L = (a^n b \mid m \geq 1, n \geq 1, nm \geq 3\}$.

Ans. Refer Q.no. 3(a) of June/ July 2018 for definition and (ii) **(08 Marks)**
i. R . E = (aa)* (bb)*

b. **Obtain the Regular expression for the following FSM.** **(08 Marks)**



Ans. Since q_2 is
R . E = 1 01* 1(0 + 1)*

OR

4. a. **Define a Regular grammar. Design regular grammars for the following languages.**
i) Strings of a's and b's with at least one a.
ii) Strings of a's and b's having strings without ending with ab.
iii) Strings of 0's and 1 's with three consecutive 0's.

Ans. i) A grammer G is 4 - tople G = (V,T,P,S) where **(08 Marks)**
V is set of variables or non - terminals
T is set of terminals
P is set of production
S is start symbol
i. V = {S,A}
T = {Q}
P = { $S \rightarrow aS$
$S \rightarrow \varepsilon$}
}
S is start symbol
ii. $S \rightarrow aA \mid bS$
$A \rightarrow aA \mid bB$
$B \rightarrow aA \mid bS \mid \varepsilon$
iii. V = {S}
T = {0,1}
P = { $S \rightarrow A\ 000A$
$A \rightarrow 0A \mid 1A \mid \varepsilon$
}
S is the start symbol

b. State and prove pumping theorem for regular languages. **(08 Marks)**

Ans. Refer Q.no. 4(c) of June / July 2018.

Module-3

5. a. Define context free grammar. Design a context free grammar for the languages. **(08 Marks)**
i) $L = \{0^m 1^m 2^n \mid m \geq 0, n \geq 0\}$ **ii) $L = \{a^i b^j \mid i \neq j, i \geq 0, j \geq 0\}$**
iii) $L = \{a^n b^{n-3} \mid n \geq 3\}$.

Ans. Refer Q.no. 5(a) of June/July 2018

i. $S \rightarrow AB$
$A \rightarrow 01 /)A1$
$B \rightarrow \varepsilon \mid 2B$
$V = \{S,A,B\}$
$T = \{0,1\}$
S is start symbol

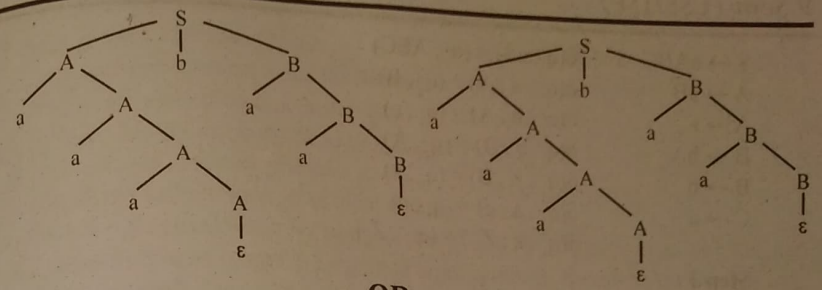
ii. $V = \{S,A,B,C\}$
$T = \{a, b\}$
$P = \{$
$S \rightarrow aSb$
$S \rightarrow A$
$S \rightarrow B$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$
$\}$
S is start symbol

iii. $V = \{S,A\}$
$T = \{a,b\}$
$P = \{$
$S \rightarrow a\,a\,a\,A$
$A \rightarrow aA\,b \mid \varepsilon$
$\}$
S is start symbol

b. Consider the grammar G with production.
$S \rightarrow AbB$
$A \rightarrow aA \mid \in$
$B \rightarrow aB \mid bB \mid \in$
Obtain leftmost derivation , rightmost derivation and parse tree for the string aaabab. **(08 Marks)**

Ans.

$S \Rightarrow A\beta B$	$\Sigma \Rightarrow A\beta B$
$\Rightarrow AbaB$	$\Rightarrow aAbB$
$\Rightarrow AbabB$	$\Rightarrow aaAbB$
$\Rightarrow Abab$	$\Rightarrow aaaAbB$
$\Rightarrow aAbab$	$\Rightarrow aaabB$
$\Rightarrow aaAbab$	$\Rightarrow aaabaB$
$\Rightarrow aaaAbab$	$\Rightarrow aaababB$
$\Rightarrow aaabab$	$\Rightarrow aaabab$

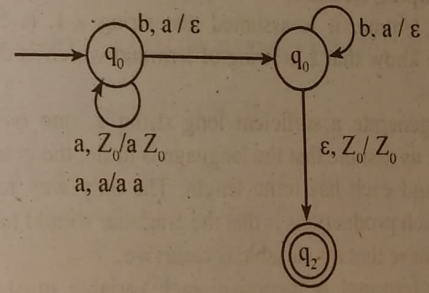


OR

6. a. Define a PDA. Obtain a PDA to accept $L = \{a^n b^n \mid W \in \{a, b\}^*\}$. Draw the transition diagram. **(08 Marks)**

Ans. Refer Q.no. 6(a) of June / July 2018

$Q = \{q_0, q_7, q_2\}$
$\Sigma = \{a,b\}$
$\Gamma = \{a, Z_o\}$
$\delta : \{$
$\delta(q_o, a, z_o) = (q_o, aZ_o)$
$\delta(q_o, a, a) = (q_o, aa)$
$\delta(q_o, b, a) = (q_1, \varepsilon)$
$\delta(q_1, b, a) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, Z_o) = (q_o, Z_o)$
$q_0 \in Q$ is the start state of machine
$Z_o \in \Gamma$ is the initial symbol on the stack
$F = \{q_2\}$ is the final state



b. Convert the following grammar into equivalent PDA.
$S \rightarrow aABC$
$A \rightarrow aB|a$
$B \rightarrow bA|b$
$C \rightarrow a.$ **(08 Marks)**

Ans. **Step 1 :**
Push start symbol
$\delta(q_o, \varepsilon, Z_o) = (q, SZ_o)$
Step 2 :

$S \rightarrow aABC$ $\quad \delta(q_1, a, S) = (q_1, ABC)$
$A \rightarrow aB$ $\quad \delta(q_1, a, A) = (q_1, B)$
$A \rightarrow a$ $\quad \delta(q_1, a, A) = (q_1, \varepsilon)$
$B \rightarrow bA$ $\quad \delta(q_1, b, B) = (q_1, A)$
$B \rightarrow b$ $\quad \delta(q_1, b, B) = (q_1, \varepsilon)$
$C \rightarrow a$ $\quad \delta(q_1, a, c) = (q_1, \varepsilon)$,
$\delta(q_1, \varepsilon, Z_o) = (q_1, Z_o)$

Step 3 :
$M = (Q, \Sigma, \Gamma, \delta, q_o, Z_o, F)$
$Q = \{q_0, q, q_f\}$
$\Sigma = \{a,b\}$
$\Gamma = \{S,A,B,C,Z_o\}$
δ is transition in step 2
$q_o \in Q$ is the start symbol
$Z_o \in \Gamma$ is initial stack symbol
$F = \{q_f\}$ is final state

Module-4

7. a. State and prove pumping lemma for context free languages. Show that $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context free. **(10 Marks)**

Ans. Statement : Let L be the context free language and is infinite. Let Z be sufficiently long string and z L so that | z | n where n is some positive integer. If the string z can be decomposed into combination of strings z = uvwxy.
Such that |vwx| n, |vx| 1, then uv'w'y L for i=0,1.2,.....

Proof of Pumping Lemma:
By pumping lemma, it is assumed that string z L is finite and is context free language. We know that z is string of terminal which is derived by applying series of productions.

Case 1: To generate a sufficient long string z, one or more variables must be recursive. Let us assume that the language is finite, the grammar has a finite number of variables and each has finite length. The only way to derive sufficiently long string using such productions is that the grammar should have one or more recursive variables. Assume that no variable is recursive.
Since no non terminal is recursive, each variable must be defined. Since those variables are also non recursive, they to be defined in terms of terminal and other variables and so on.
From this we conclude that there is a limit length of the string that is generated from the start symbol S. this contradicts our assumption that the language is finite.
Therefore, the assumption that one or more variable are non recursive is incorrect.
This means that this means that one or more variable are non recursive and hence the proof.

Case 2: The string z L implies that after applying some / all production some number of times, we get finally string of terminal and the derivation stops.

Let z L is sufficiently long string and so the derivation must have involved recursive use of some non terminal A and the derivation must have the form:
Note that any derivation should start from the start symbol S.
A DFA is a 5-tuple or quintuple $M=(Q, \Sigma, q_o, F)$
Q is non-empty, finite set of states.
1 is non-empty, finite state set of input alphabet.
is transition function, which is mapping from Q. X Σ to Q. for this transition function the parameters to be passed are state and input symbols. Based on the current state and input symbols, the machine may enter into another state. qo Q is the start state.
F Q is a set of accepting or final state. Note: for each input symbol a, from a given state there is exactly one transition and we are sure to which state the machine enters. So the machine is called Deterministic Machine
Lanbncn = $\{a^n b^n c^n \mid n > 0\}$ is not a CFL
Proof. Suppose $La^n b^n c^n$ is context-free. Let p be the pumping length.

- Consider $z = a^p b^p c^p \in La^n b^n c^n$.
- Since $|z| > p$, there are u, v, w, x, y such that $z = uvwxy$, $|vwx| \leq p$, $|vx| > 0$and $uv^iwx^iy \in L$ for all $i > 0$.
- Since $|vwx| \leq p$, vwx cannot contain all three of the symbols a, b, c, because there are p bs. So vwx either does not have any as or does not have any bs or does not have any cs. Suppose, (w log) vwx does have any as. Then $uv^0wx^0y = uwy$ contains more as than either bs or cs. Hence $uwy \notin L$.

b. Explain Turing machine model. **(06 Marks)**

Ans. Refer Q.no. 8(b) of Dec 2017/ Jan 2018.

OR

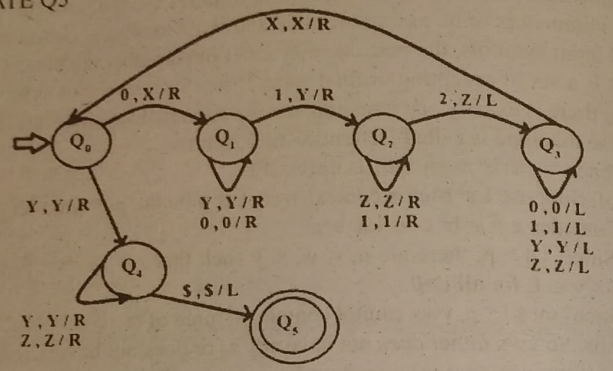
8. a. Design a Turing machine to accept the language $L = \{0^n 1^n 2^n \mid n \geq 1\}$.(08 Marks)

Ans. Step-1:
Replace 0 by X and move right, Go to state Q.1.
Step-2:
Replace 0 by 0 and move right, Remain on same state
Replace Y by Y and move right, Remain on same state
Replace 1 by Y and move right, go to state Q2.
Step-3:
Replace 1 by 1 and move right, Remain on same state
Replace Z by Z and move right, Remain on same state
Replace 2 by Z and move right, go to state Q3.
Step-4:
Replace 1 by 1 and move left, Remain on same state
Replace 0 by 0 and move left, Remain on same state
Replace Z by Z and move left, Remain on same state
Replace Y by Y and move left, Remain on same state
Replace X by X and move right, go to state QO.
Step-5 :

If symbol is Y replace it by Y and move right and Go to state Q4
Else go to step 1

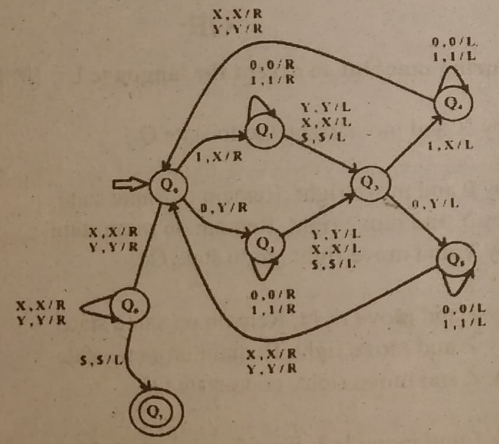
Step-6:

Replace Z by Z and move right, Remain on same state
Replace Y by Y and move right, Remain on same state
If symbol is $ replace it by $ and move left, STRING IS ACCEPTED, GO TO FINAL STATE Q5



b. Design a Turing machine to accept strings of a's and b's ending with ab or ba. (08 Marks)

Ans.



Module-5

9. a. Explain the following :
i) Non deterministic Turing machine ii) Multi - tape Turing machine.(06 Marks)

Ans. REfer Q.no. 10 (a),(b) of June / July 2018.

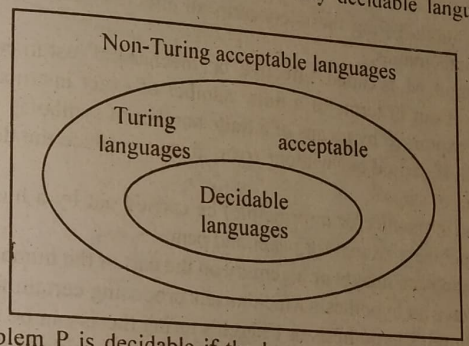
b. Define the following :
i) Recursively enumerable language ii) Decidable language.

Ans. **Recursive Enumerable (RE) or Type -0 Language** **(06 Marks)**

RE languages or type-0 languages are generated by type-0 grammars. An RE language can be accepted or recognized by Turing machine which means it will enter into final state for the strings f language and may or may not enter into rejecting state for the strings which are not part of the language. It means TM can loop forever for the strings which are not a part of the language. RE languages are also called as Turing recognizable languages.

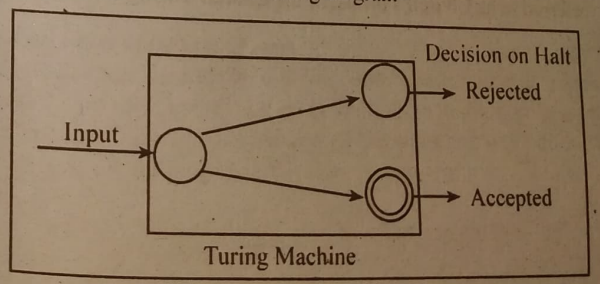
Decidable language

A language is called Decidable or Recursive if there is a Turing machine which accepts and halts on every input string w. Every decidable language is Turing-Acceptable.



A decision problem P is decidable if the language L of all yes instances to P is decidable.

For a decidable language, for each input string, the TM halts either at the accept or the reject state as depicted in the following diagram -



c. What is Post correspondence problem? **(04 Marks)**

Ans. Refer Q.no. 10(d) of June/ July 2018.

OR

10. a. What is Halting problem of Turing machine? **(06 Marks)**

Ans. Refer Q.no. 10(b) of Dec 2017 / Jan 2018.

b. Define the following : i) Quantum computer ii) Class NP. **(06 Marks)**

Ans. **i) Quantum Computer :** Refer Q. no 9 c of June/July 2018

ii) Class NP : The class NP consists of those problems that are verifiable in polynomial time. NP is the class of decision problems for which it is easy to check the correctness of a claimed answer, with the aid of a little extra information. Hence, we aren't asking for a way to find a solution, but only to verify that an alleged solution really is correct. Every problem in this class can be solved in exponential time using exhaustive search.

c. Explain Church Turing Thesis. **(04 Marks)**

Ans.

- The Church-Turing thesis concerns an effective or mechanical method in logic and mathematics.
- A method, M, is called 'effective' or 'mechanical'just in case:
- M is set out in terms of a finite number of exact instructions (each instruction being expressed by means of a finite number of symbols)
- M will, if carried out without error, always produce the desired result in a finite number of steps
- M can (in practice or in principle) be carried out by a human being unaided by any machinery except for paper and pencil
- M demands no insight or ingenuity on the part of the human being carrying it out.
- They gave an hypothesis which means proposing certain facts.
- The Church's hypothesis or Church's turing thesis can be stated as:
- The assumption that the intuitive notion of computable functions can be identified with partial recursive functions.
- This statement was first formulated by Alonzo Church in the 1930s and is usually referred to as Church's thesis, or the Church-Turing thesis.

Fifth Semester B.E. Degree Examination, CBCS - June / July 2019

Automata Theory and Compatibility

Time: 3 hrs. Max. Marks: 80

Note : Answer any FIVE full questions, selecting ONE full question from each module.

Module - 1

1. a. Define the following : i) string ii) alphabet iii) language. **(06 Marks)**

Ans. Refer Q.No. 1.a. of Dec 2018 / Jan 2019

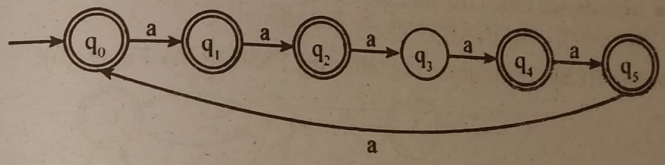
b. Design a deterministic finite State machine for the following language over Σ = {a, b}.

i) L = {W| |W| mod 3,> |W| mod 2}

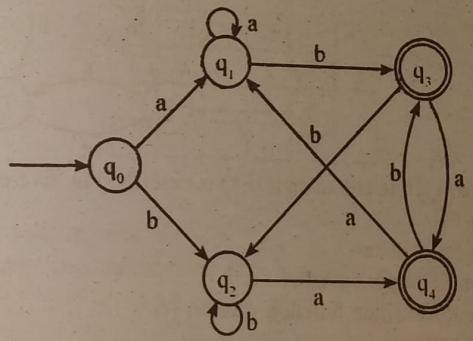
ii) L = {w | W ends either with ab or ba}. **(10 Marks)**

Ans.

i)



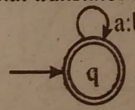
ii)



OR

2. a. Write a note on finite state transducers. **(07 Marks)**

Ans. A finite state transducer essentially is a finite state automaton that works on two (or more) tapes. The most common way to think about transducer is as a king of "Translating machine". They read from one of the tapes and write on to the other. This for instance, is a transducer that translates aS into bS.



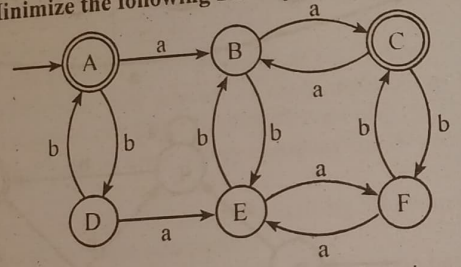
a : but the arc means that in this transition the transducer reads a from the first tape and writes b onto the second.

Transducers can however, be used in other modes that the translation made as well in the generation mode Transducers write on both tapes and in the recognition mode they read from both tapes. Further more, the direction of translation can be turned around i.e., a : b can not only be read as read a from the first tape and write b onto the second tape, but also "Read b from the second tape and write a on to the first tape.

So, the above transducer behaves as follows in the different modes.

- Generation mode : it write a string of aS on one tape and a string bS on the other tape. Both strings how the same length.
- Recognition mode : it accepts when the word on the first tape consists of exactly as many aS as the word on the second tape consist of bS.
- Translation mode (left to right) : it reads aS from the first tape and writes an b for every a that it reads onto the second tape.
- Translation mode (right to left) : it reads bS from the second tape and write on a for every f that ir reads onto the first tape.

b. Define DFSM? Minimize the following FSM. [Refer Fig.Q2(b)] **(09 Marks)**



Ans. A deterministic finite automata (DFA) is described by five element tuple:

$(Q, \Sigma, \delta, q_0, F)$

Q is a finite set of states

Σ is a non empty input alphabet

δ is a series of transition function

q_0 is starting state

F is final state

		a	b
→	*A	B	D
	B	C	E
	*C	B	F
	D	E	A
	E	F	B
	F	E	C

Step 1 :-

	B	X				
*	C	X	X			
	D	X		X		
	E	X		X		
	F	X		X		
		A	B	C	D	E
			*		*	

Step 2 :-

	a	b
(B, D)	(C, E)	(E, A)
(B, E)	(C, F)	(E, B)
(B, F)	(C, E)	(E, C)
(D, E)	(E, F)	(A, B)
(D, F)	(E, E)	(A, C)
(E, F)	(F, E)	(B, C)

B	X				
C	X	X			
D	X	X	X		
E	X	X	X	X	
F	X	X	X	X	X
	A	B	C	D	E

Distinguish pairs

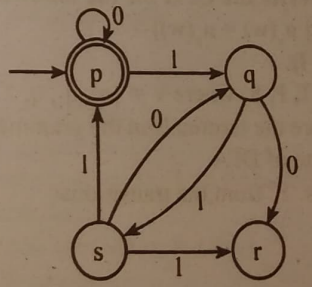
A, B, C, D, E, F

Since there is no in distinguishable pairs

Given DFA cannot be minimized its already in minimized state.

Module-2

3. a. Write the equivalent Regular Expression for the given Finite state machine. [Refer Fig.Q3(a)] **(08 Marks)**



Ans. Since the problem given doesn't have any transition for 'r' for the input 'o' it cant be solved.
By assuming the transition 'r' for the input 'o' goes to 'q' we get following R.E
$= 1 * 0(11 * 0) * 0 + 1 * 0 (11 * 0) * 0 [(0 + \varepsilon) + 1(11 * 0)*0]* (0 + \varepsilon) + 1(11 * 0)*0$

b. Write the Regular Expression for the following language.
i) {w ∈ {a, b}* with atmost one a}
ii) {w ∈ {a, b}* does not end with ba}
iii) {w ∈ {0, 1}* has substring 001}
iv) {w ∈ {0, 1 }* |W| is even}. **(08 Marks)**

Ans. i) (b + a) b * (ε + a)
ii) (a + b) * (aa | ab | bb) |a|b| ε
iii) (0 + 1) * 001 (0 + 1)*
iv) ((a + b) (a + b))*

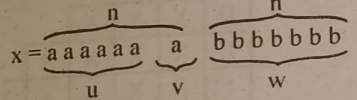
OR

4. a. State and prove the pumping theorem for regular language. **(08 Marks)**

Ans. Refer Q.No. 4.c. of June / July 2018

b. Show that the language L = {aⁿbⁿ | n > 0} is not regular. **(08 Marks)**

Ans. **Step 1 :-** Let L is regular and n be the number of states in FA.
Consider the string $x = a^n b^n$



Step 2 :- Since $1 \times 1 = 2n \geq n$, we can split x into uvw such that $|uv| \leq n$ and $|v| \geq 1$ as shown below.
Where $|u| = n - 1$ and $|v| = 1$ so that $|uv| = |u| + |v| = n - 1 + 1 = n$ and $|w| = n$.
According pumping lemma $uv^i w \in L$ for i = 0, 1, 2.....
Step 3 :- if i = 0, v does not appear and number of a's will be less than b's which is contradict to state of pumping lemma. Hence given language is not regular.

Module-3

5. a. Define grammar. Write the CFG for the following language.
i) L = {w ∈ {a, b}*| $n_a(w) = n_b(w)$}
ii) L = {$a^i b^j$ | i = j + 1}. **(08Marks)**

Ans. A grammar G = (V, T, P, S) where V = {q_0, q_1, q_2 .....} is states of DFA
T is input alphabet are the terminals in the grammar
S = q_0 is the start state of DFA
'P' is the productions 'P' from the transitions

i) V = {S}
T = {a, b}
P = {
S → ε
S → a S b
S → b S a
}
s is start symbol

ii) V = {S}
T = {a, b}
P = {
S → OA
A → OA I
A → ε
}
s is start symbol

b. What is inherent ambiguity? Show that the language given is inherently amtriguous?
L = {$a^n b^n c^m$ | n, m ≥ 0} ∪ {$a^n b^m c^n$ | n, m ≥ 0} **(08 Marks)**

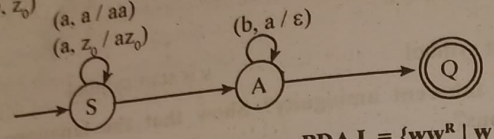
Ans. In many cases, when confronted with an ambiguous grammar 'G' it is possible to construct a new grmmar and that generates L(G) and that has less (or no) ambiguity unfortunately it is not always possible to do this. There exist context free language for which no unambiguous grammar exits. We call such languages inherently ambiguous.
L = {$a^i b^j c^k$: i, j, k ≥ 0, i = j or j = k}.
An alternative way to describe it is L = {$a^n b^n c^m$ | n. m ≥ 0} ∪ {$a^n b^m c^n$ | n, m ≥ 0}
Every string in L has either (or both) the same number of a's and b's or the same number of b's and c's. L is inherently ambiguous one grammar that describes it is
G = {S, S_1, S_2, A, B, a, b, c}, (a, b, c}, R, S} where
R = {S → S_1 | S_2
S_1 → S_1 c | A
A → a A b | ε
S_2 → aS_2 | B
B → b B c | ε}
Now consider the strings in $A^n B^n C^n$ = {$a^n b^n c^n$: n ≥ 0} They have two distinct derivations one through S_1 and the other through S_2 it is possible to move that L is inherently ambiguous : Given any grammar G that generates L there is at least one string with two derivation in G.

OR

6. a. Define PDA? Design PDA for the language L = {$a^n b^m a^n$ | n, m ≥ 0}. (06 Marks)

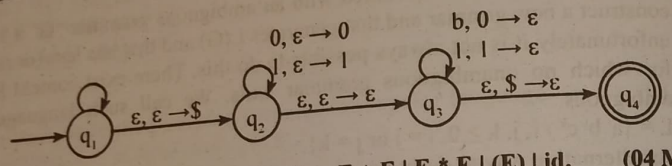
Ans. A Push Down Automata is a seven tuple
M = (Q, Σ, ⌈ , δ, q_0, z_0, F)
where Q is set of finite states
Σ is set of input alphabets
⌈ is set of stack alphabets
δ - transition form Q (Σ ∪ ε) × ⌈ to finite Suv set of Q × P*
q_0 ∈ Q is the start state
z_0 ∈ P is the initial symbol on the stack
F ≤ Q is set of final states

$\delta(S, a, z_0) = (A, a\, z_0)$
$\delta(S, a, a) = (S, a\, a)$
$\delta(A, b, a) = A$
$\delta(A, a, az_0) = (A, \varepsilon)$
$\delta(A, \varepsilon, z_0) = (Q, z_0)$



b. Convert the following language from CFG to PDA L = {ww^R | w ∈ {0, 1 }*}. **(06 Marks)**

Ans.



c. Convert the following CFG to CNF E → E + E | E * E | (E) | id. **(04 Marks)**

Ans. $E \rightarrow E\,E^1 \mid (E)$
$E^1 \rightarrow +E \mid *E$
$E \rightarrow io \mid$
$V = \{E\}$
$T = \{id_1 + E\}$
$P = \{$
$E \rightarrow EE^1 \mid (E)$
$E^1 \rightarrow +E \mid *E$
$E \rightarrow id$
$\}$

Module-4

7. a. Prove that the language L = {a^n b^n c^n | n ≥ 0} is not context free. **(08 Marks)**

Ans. Suppose this language is contact free; then it has a context free grammar. Let K be the constant associated with this grammar by the pumping Lemma. Consider the string $a^Kb^Kc^K$, which is in L and has length greater than K.
By the pumping Lemma this must be representable as uvxyz, such that all uv^ixy^iz are also in L. This is impossible, since

- either v and y cannot contain a mixture of letters from {a, b, c}; otherwise they would be in the wrong order for uv^2xy^2z
- if v or y contain just 'a's, 'b's or 'c's, then uv^2xy^2z cannot maintain the balance between the three letters (it can, of course maintain the balance between two) Q ε D

b. Prove that CFL are not closed under intersection, complement or difference?

Ans. Refer Q.No. 7.a. of Dec 2017 / Jan 2018 **(08 Marks)**

OR

8. a. Design a Turing machine to accept L = {a^n b^n c^n | n ≥ 0}.
Ans. Refer Q.No. 8.b. of Dec 2017 / Jan 2018 **(08 Marks)**

b. Define a turning machine. Explain the working of a turning machine. **(05 Marks)**
Ans. Refer Q.No. 8.a. of Dec 2017 / Jan 2018

c. Write a note on multitape machine.
Ans. Refer Q.No. 9.a. of Dec 2017 / Jan 2018 **(03 Marks)**

Module-5

9. Write a short notes on :
a. Growth rate of function **(05 Marks)**
b. Church-turning thesis **(06 Marks)**
c. Linear bounded automata. **(05 Marks)**

Ans. a. Growth rate of function : One of the most important problems in computer science is to get the best measure of the growth rates of algorithms, best being those algorithms whose run times grow the slowest as a function of the size of their input. Efficiency can mean survival of a company. For example, a sort of measure 0(2 n) on a database of millions of customers may take several days to run, whereas one of measure 0(n · log n) may take only a few minutes!
However, the big O estimate, does not necessarily give the best measure of the growth rate of a function. One can say that the growth rate of a sequential search is O (n^2), but one knows the number of comparisons is approximately proportional to n, n the number of input items. We would like to say that sequential search is O(n) (it is), but the notion of big O is not precise enough. Therefore, in this section we define theta Θ notation to more precisely measure the growth rate of functions and big omega Ω notation. The most important of these is O . We also define o and ω notation.
b. Church-turning thesis : Refer Q.No. 10.c. of Dec 2018 / Jan 2019
c. Linear bounded automata : Refer Q.No. 10.c. of June / July 2019

OR

10. Write a short notes on :
a. Post correspondence problem **(05 Marks)**
b. Halting problem in turning machine **(05 Marks)**
c. Various types of turning machine. **(06 Marks)**

Ans. a. Post correspondence problem : Refer Q.No. 10.c. of Dec 2017 / Jan 2018
b. Halting problem in turning machine : Refer Q.No. 10.b. of Dec 2017 / Jan 2018

c. Various types of turning machine :

1) Multiple track
2) Shift over Turing Machine
3) Nondeterministic
4) Two way Turing Machine
5) Multitape Turing Machine
6) Multidimensional Turing Machine
7) Composite Turing Machine
8) Universal Turing Machine

Refer Q. no 9.a. of Dec 2018/Jan 2019

Fifth Semester B.E. Degree Examination, CBCS - Dec 2019 / Jan 2020

Automata Theory and Computability

Time: 3 hrs. Max. Marks: 100

Note : Answer any FIVE full questions, selecting ONE full question from each module.

Module-1

1. a. **Explain with example,**
(i) Strings (ii) Language (iii) Function on string (06 Marks)

Ans. **i) Strings**

A string is a finite sequence, possibly empty, of characters drawn from some alphabet Σ.

- ε is the empty string
- Σ* is the set of all possible strings over an alphabet Σ.

Examples Σ = {a, b}

Strings ={ε.a, b, ab, aa, bb, ba, aaaa……}

ii) language

A language is a (finite or infinite) set of strings over a (finite) alphabet Σ.

Examples: Let Σ = {a, b}

Some languages over Σ:

∅ = { } // the empty language, no str

{ε} // language contains only the empty

{ε, a, aa, aaa, aaaa, aaaaa} //language have in a`s

iii) Functions on Strings

Length: is the number of characters in string s.

- |s| is the length of string s

Example:

|ε| = 0

|1001101| = 7

Concatenation : the concatenation of 2 strings s and t is the string formed by appending t to s; written as s||t or more commonly, st

Example:

If x = good and y = bye, then xy = goodbye and

yx = byegood

Replication: For each string w and each natural number k,

the string w^k is: $w^0 = \varepsilon$

$w^{k+1} = w^k w$

Examples: a^3 = aaa

Reverse: For each string w, w^R is defined as:

if |w| = 0 then $w^R = w = \varepsilon$

if |w| = 1 then w^R = w if |w| > 1 then: ∃ a ∈ Σ (∃ u ∈ Σ* (w = ua)) So define w^R = a u^R OR if |w| > 1 then: ∃ a ∈ Σ & ∃ u ∈ Σ* ∋ w = ua So define w^R = a u^R

Example: s =goodbye

$(s)^R$ = eybdoog

b. Discuss standard operations on Languages with example. **(04 Marks)**

Ans. Operation on language

- Union
- Intersection
- Complement
- Difference
- Concatenation
- Kleene star

Example : Σ = {a,b}

L_1 = {w∈{a, b}*: string with an even number of a's}

L_2 = {w∈{a, b}*: string with no b's}

So L_1 = {ε, b, aa, bb, aab, abab,… }

L_2 = {ε, a, aa, aaa, aaaa,… }

- Union

$L_1 \cup L_2$ = {ε, a, b, aa, bb,aaa, abab, aaaa,… }

- Intersection

$L_1 \cap L_2$ = {ε, aa, aaaa, aaaaaa,… }

- Difference

$L_2 - L_1$ ={a, aaa, aaaaa, aaaaaaa,… }

- Complement

$\neg L_2$ = {b, ab, abaa, bbaaaàa, aababaaaa,… } = {w ∈ {a, b}*: all strings that contain at least one b}

- Concatenation

If L_1 and L_2 are languages over Σ:

$L_1 L_2$ ={w ∈ Σ* : ∃s ∈ L_1(∃t ∈ L_2(w = st))}

L_1 = {cat, dog}

L_2 = {apple, pear}

$L_1 L_2$ ={catapple, catpear, dogapple, dogpear}

$L_2 L_1$ ={applecat, appledog, pearcat, peardog}

- Kleene Star

L* – language consisting of 0 or more concatenations of strings from L

Examples : L = {dog, cat, fish}

L* = {ε, dog, cat, fish, dogdog, dogcat, dogfish, fishcatfish, fishdogdogfishcat, ...}

c. Construct DFSM for the following languages :

(i) L = {ω ∈ {a, b}* | ω contains no more than one b},

(ii) L = {ω ∈ {a,b}* | ω contains Even number of a's and odd number of b's}

Give the transition Table and show that aabaa is accepted. **(10 Marks)**

Ans. (i) M = (k, Σ, δ, q_0, F)

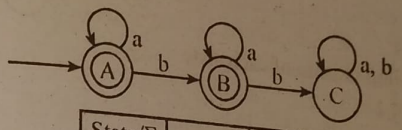
k = {A, B, C}

Σ = {a, b}

F = {A, B}

q_0 = A

string = aabaa



State/Σ	a	b
(A)	A	B
(B)	B	C
C	C	C

Computation

(A, aabaa) → (A, abaa) → (A, baa) → (B, aa) → (B, a) → (B, ε)

Machine has **halted** and B is final state. Hence string is accepted.

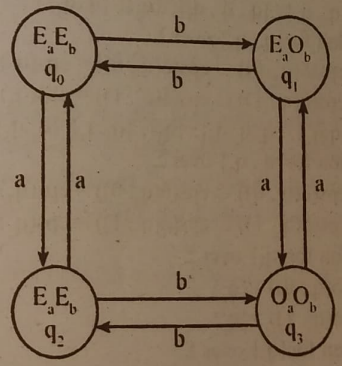
(ii) M = (k, Σ, δ, q_0, F)

k = {q_0, q_1, q_2, q_3}

Σ = {a, b}

F = {q_1}

string = aabaa



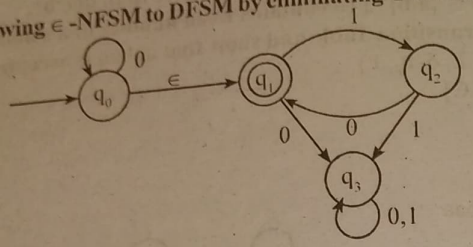
Computation

(q_0, aabaa) → (q_2, abaa) → (q_0, baa) → (q_1, aa) → (q_3, a) → (q_1, ε)

Machine has halted and q_1 is final state. Hence string is accepted.

OR

2. a. Convert the following ∈ -NFSM to DFSM by eliminating ∈-transition. (10 Marks)



Ans. Compute epsilon closure of each state

$eps(q_0)=\{q_0, q_1\}$
$eps(q_1)=\{q_1\}$
$eps(q_2)=\{q_2\}$
$eps(q_3)=\{q_3\}$
Start state = $eps(q_0) = \{q_0,q_1\}$
Active state = $(\{q_0, q_1\})$
Compute transition for $\{q_0, q_1\}$ over Σ
Consider state $\{q_0, q_1\}$ and find δ' on each symbol on $\sum = \{0,1\}$
$\delta'(\{q_0, q_1\},0) = eps(\delta(q_0, 0)) \cup eps(\delta(q_1, 0)) = eps(q_0) \cup eps(q_3) = \{q_0,q_1,q_3\}$
$\delta'(\{q_0, q_1\},1) = eps(\delta(q_0, 1)) \cup eps(\delta(q_1, 1)) = eps(\emptyset) \cup eps(q_2) = \{q_2\}$
Active state = $(\{q_0, q_1\}, \{q_0, q_1, q_3\}, \{q_2\})$
Compute transition for $\{q_0, q_1, q_3\}$ over Σ
$\delta'(\{q_0, q_1, q_3\},0) = eps(\delta(q_0, 0)) \cup eps(\delta(q_1, 0)) \cup eps(\delta(q_3, 0)) = \{q_0, q_1, q_3\}$
$\delta'(\{q_0, q_1, q_3\},1) = eps(\delta(q ,1)) \cup eps(\delta (q_1, 1)) \cup eps(\delta(q_3, 1)) = \{q_2, q_3\}$
Active state = $(\{q_0, q_1\}, \{q_0, q_1, q_3\}, \{q_2\}, \{q_2,q_3\})$
Compute transition for $\{q_2, q_3\}$ over Σ
$\delta'(\{q_2, q_3\},0) = eps(\delta(q_2, 0)) \cup eps(\delta(q_3, 0)) = eps(q_1) \cup eps(q_3) = \{q_1, q_3\}$
$\delta'(\{q_2, q_3\},1) = eps(\delta(q_2, 1)) \cup eps(\delta(q_3, 1)) = eps(q_3) \cup eps(q_3) = \{q_3\}$
Active state = $(\{q_0q_1\}, \{q_0,q_1,q_3\}, \{q_2\}, \{q_2,q_3\}\ \{q_1,q_3\}, \{q_3\})$
Compute transition for $\{q_1, q_3\}$ over Σ
$\delta'(\{q_1, q_3\},0) = eps(\delta(q_1, 0)) \cup eps(\delta(q_3, 0)) = eps(q_3) \cup eps(q_3) = \{q_3\}$
$\delta'(\{q_1, q_3\},1) = eps(\delta(q_1, 1)) \cup eps(\delta(q_3, 1)) = eps(q_2) \cup eps(q_3) = \{q_2q_3\}$
Compute transition for $\{q_3\}$ over Σ
$\delta'(\{q_3\},0) = eps(\delta(q_3, 0)) = \{q_3\}$
$\delta'(\{q_3\},1) = eps(\delta(q_3, 1)) = \{q_3\}$
Compute transition for $\{q_2\}$ over Σ
$\delta'(\{q_2\},0) = eps(\delta(q_2, 0)) = \{q_1\}$
$\delta'(\{q_2\},1) = eps(\delta(q_2, 1)) = \{q_3\}$

Active state=$(\{q_0q_1\}, \{q_0,q_1,q_3\}, \{q_2\}, \{q_2,q_3\}\ \{q_1,q_3\}, \{q_3\}, \{q_1\})$
Compute transition for $\{q_1\}$ over Σ
$\delta'(\{q_1\},0) = eps(\delta(q_1,0)) = \{q_3\}$
$\delta'(\{q_1\},1) = eps(\delta(q_1,1)) = \{q_2\}$
Active state=$(\{q_0q_1\}, \{q_0,q_1,q_3\}, \{q_3\}, \{q_2,q_3\}\ \{q_1,q_3\}, \{q_3\}, \{q_1\})$

		0	1
A	$\{q_0q_1\}$	$\{q_0,q_1,q_3\}$	$\{q_2\}$
B	$\{q_0,q_1,q_3\}$	$\{q_0,q_1,q_3\}$	$\{q_2,q_3\}$
C	$\{q_2\}$	$\{q_1\}$	$\{q_3\}$
D	$\{q_2,q_3\}$	$\{q_1,q_3\}$	$\{q_3\}$
E	$\{q_1,q_3\}$	$\{q_3\}$	$\{q_2,q_3\}$
F	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$
G	$\{q_1\}$	$\{q_3\}$	$\{q_2\}$

b. Define distinguishable and indistinguishable states. Minimize the number of states in DFSM. (10 Marks)

δ	0	1
→A	B	F
B	G	C
C	A	G
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

Ans. Refer Q.No. 2.b. from Dec 2018 / Jan 2019

Module - 2

3. a. Define Regular expression. Write RE for the following :
(i) Language of all strings of 0's and 1's that have odd number of 1's.
(ii) Language of all strings of 0's and 1's that has at least one pair of consecutive 0's.
(iii) The Language of all strings of 0's and 1's that have no pair's of consecutive 0's. (10 Marks)

Ans. Regular expression : Refer Q.No. 3.a. of MQP - 1
i) RE = 0*(10*10*)*10*
ii) RE = 1*(00)*0*1
iii) RE = $(1 \cup \varepsilon)(01)^*(0 \cup \varepsilon) + (0 \cup \varepsilon)(10)^*(1 \cup \varepsilon)$

b. Prove with an example that the class of language can be defined with regular Grammar is exactly the regular language. **(10 Marks)**

Ans. Theorem : The class of languages that can be defined with regular grammars is exactly the regular languages.

Proof : We first show that any language that can be defined with a regular grammar can be accepted by some FSM and so is regular. Then we must show that every regular language (i.e., every language that can be accepted by some FSM) can be defined with a regular grammar. Both proofs are by construction,

Regular grammar – FSM : The Following algorithm constructs an FSM M from a regular grammar G = {V. Σ, R, S) and assures that L (M) = L (G):

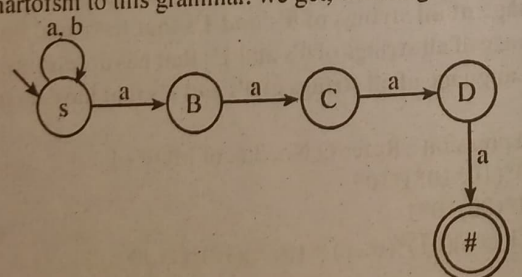
grammart of sn{G: regular grammar) =

1. Create in M a separate stale for each nonterminal in V.
2. Make the state corresponding to S the start state.
3. If there are any rules in R or the form X → w, for some to w ∈ Σ, then create an additional state Labeled #.
4. For each rule of the form X → wY. add a transition from X to Y labeled w.
5. For each rule of the form X → w, add a transition from X to # labeled w.
6. For each rule of the form X → ε, mark state X as accepting.
7. Mark state # as accepting.
8. If M is incomplete {i.e.. there are some (stale, input) pairs for which no transition is defined). M requires a dead state. Add a new state D, For every (q, i) pair for which no transition has already been defined, create a transition from q to D labeled i. For every i in Σ, create a transition from D to D labeled i.

Example : L = {w ∈ {a, b} *: w ends with the pattern aaaa}.

S → aS
S → bS
S → aB
B → aC
C → aD
D → a

Applying grammartofsm to this grammar. we get, omitting the dead state:



OR

4. a. Using Kleen's theorem, prove that any language that can be defined with a Regular expression can be accepted by some FSM. **(10 Marks)**

Ans. Theorem: Any language that can be defined with a regular expression can be accepted by some FSM and so is regular.

Proof: The proof is by construction . We will show that. given a regular expression a, we can construct an FSM M such that L (a) = L (M).

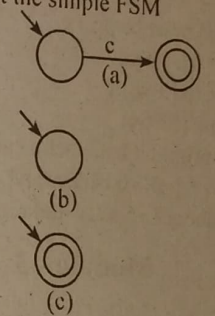
We first show that there exists an FSM that corresponds to each primitive regular expression:

If a is any c ∈ Σ. we construct for it the simple FSM

If a is Ø. We construct for it the simple FSM shown in.

Although it's not strictly necessary to consider ε since it has the same meaning as Ø*

So, if a is ε, we construct for it the simple FSM



- If α is the regular expression β ∪ γ and if both L((β) and L(γ) are regular, then we construct $M_3 = (K_3, \Sigma, \delta_3, s_3, A_3)$ such that L (M_3) = L(α) = L (β) ∪ L (γ). If necessary, rename the states of M_1 and M_2 so that $K_1 \cap K_2 = \emptyset$. Create a new start state, s_3, and connect it to the start states of M_1 and M_2 via ε-transitions. M_3 accepts iff either M_1 or M_2 accepts So $M_3 = (\{s_3\} \cup K_1 \cup K_2, \Sigma, \delta_3, s_3, A_1 \cup A_2)$, where $\delta_3 = \delta_1 \cup \delta_2 \cup \{((s_3, \varepsilon), s_1)\ ((s_3, \varepsilon), s_2)\}$.
- If α is the regular expression βγ and if both L(β) and L(γ) are regular, then we construct $M_3 = (K_3, \Sigma, \delta_3, s_3, A_3)$ such that L(M_3) = L(α) = L(β) L(γ). If necessary, rename the states of M_1 and M_2 so that $K_1 \cap K_2 = \emptyset$. We will build M_3 by connecting every accepting state of M_1 to the start state of M_2 via an e-transition. M_3 will start in the start state of M_1 and will accept iff M_2 does.
  So $M_3 = (K_1 \cup K_2, \Sigma, \delta_3, s_3, A_2)$.where $\delta_3 = \delta_1 \cup \delta_2 \cup \{((q, \varepsilon), s_2) : q \in A_1\}$,
- If α is the regular expression β* and if L (β) is regular, then we construct $M_2 = (K_2, \Sigma, \delta_2, s_2, A_2)$ such that L(M_2) = L(α) = L(β)*. We will create a new start state s_2 and make it accepting, thus assuring that M_2 accepts ε. (We need a new start state because it is possible that s_1 the start slate of M_1 is not an accepting state. If it isn't and if it is reachable via any input string other than ε, then simply making it an accepting state would cause M_2 to accept strings that are not in $(L(M_1))^*$.) We link the new s_2 to s_1, via an ε-transitions. Finally, we create e-transitions from

each of M_1's accepting states back to s_1. So $M_2 = (\{s_2\} \cup K_1, \Sigma, \delta_2, s_2, \{s_2\} \cup A_1)$, where $\delta_2 = \delta_1 \cup \{((s_2, \varepsilon), s_1)\} \cup \{((q, \varepsilon), s_1) : q \in A_1\}$.

b. State and prove pumping lemma for regular language and show that the language L = {a^p | P is a prime number} is not regular. **(10 Marks)**

Ans. Pumping lemma for Regular language refer Q. No. 3.b. from Model paper 1

Prime. = {a^n : n is prime}.

Let L be Prime a = (a^n : n is prime). We can use the Pumping Theorem to show that L is not regular. If it were, then there would exist some k such that any string w, where $|w| \geq k$, must satisfy the conditions of the theorem. We show one string w that does not. Let $w = a^j$, where j is the smallest prime number greater than k + 1. Since $|w| > k$, w must satisfy the conditions of the Pumping Theorem. So there must exist x, y, and z, such that $w = xyz$, $|xy| \leq k$ and $y \neq \varepsilon$. $y = a^p$ for some p. The Pumping Theorem further requires that $\forall q \geq 0$ ($xy^q z \in L$). So, $\forall q \geq 0$ ($a^{|x|+|z|+q|y|}$ must be in L). That means that $|x| + |z| + q \cdot |y|$ must be prime.

But suppose that $q = |x| + |z|$. Then:

$|x| + |z| + q.|y| = |x| + |z| + (|x| + |z|).y$

$= (|x| + |z|) - (1 + |y|)$,

which is composite (non-prime) if both factors are greater than 1. $(|x| + |z|) > 1$ because $|w| > k + 1$, and $|y| \leq k$. $(1 + |y|) > 0$ because $|y| > 0$. So, for at least that one value of q, the resulting string is not in L. So L is not regular.

Module - 3

5. a. Define context Free Grammar. Construct CFG for the following languages:

(i) Balanced parantheses.

(ii) L = {ω ∈ {a,b}* | co contains substring a, b} and derive two strings for each language along with parse tree. **(10 Marks)**

Ans. A context-free grammar G is a quadruple, (V, Σ, R, S), where:

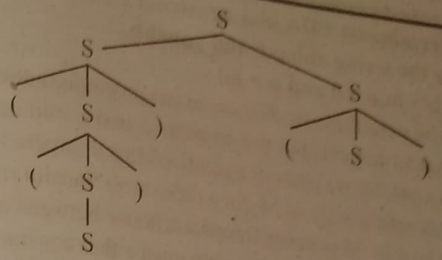
- V is the rule alphabet, which contains nonterminals and terminals.
- Σ (the set of terminals) is a subset of V,
- R (the set of rules) is a finite subset of $(V - \Sigma) \times V^*$,
- S (the start symbol) is an element of $V - \Sigma$.

i) Bal = {w ∈ {), (}*: the parentheses are balanced}

G = {{S,), (}, {), (} , R, S). where:

R = {S → (S), S → SS S → ε}.

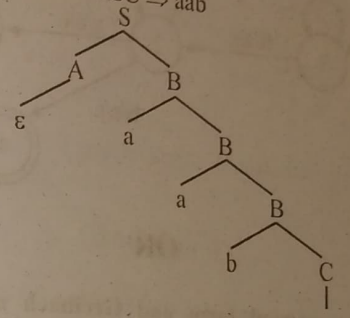
Example : S ⇒ (SS) ⇒ ((S)S) ⇒ (()S) ⇒ (()(S)) ⇒ (()()).



S ⇒ (S) ⇒ ((S)) ⇒ (())

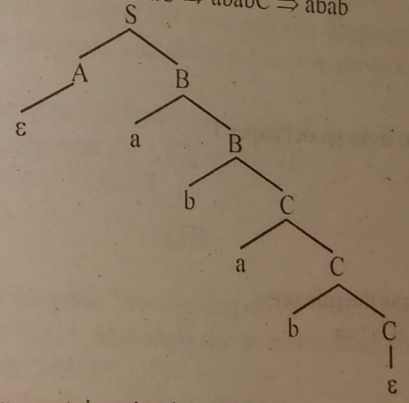
String = aab

S ⇒ AB ⇒ εB ⇒ aB ⇒ aaB ⇒ aabC ⇒ aab



String = abab

S ⇒ AB ⇒ εB ⇒ aB ⇒ abC ⇒ abaC ⇒ ababC ⇒ abab



ii) L = {w ∈ {a, b}*:w contain substring ab}

S → AB

A → bA|ε

B→aB|bC|b

C → aC|bC| ε

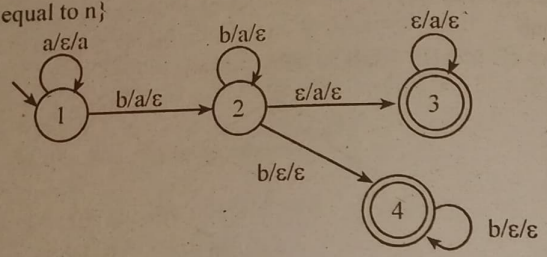
b. Explain deterministic PDA and construct DPDA for language given and give the trace for the string abbaab and aababb. **(10 Marks)**

$L = \{a^n b^m a^m b^n \mid m.n > 0 \text{ and } n = m\}$

Ans. A PDA M to be deterministic iff there exists no configuration of M in which M has a choice of what to do next. For this to be true. two conditions must hold:

1. Transition contains no pairs of transitions that compete with each other.
2. If q is an accepting state of M. then there is no transition ((q, ε,ε). (p. a)) for any p or a. In other words. M is never forced to choose between accepting and continuing.

Any transitions out of an accepting state must either consume input $L=\{a^n b^m a^m b^n \mid m,n >0$ and m not equal to n$\}$



OR

6. a. Discuss Chomsky normal form and Greibach normal form. Convert the following Grammar to Chomsky Normal form,

$S \rightarrow aACa$
$A \rightarrow B|a$
$B \rightarrow C$
$C \rightarrow cC \mid \in$ **(10 Marks)**

Ans. Refer Q.No. 8.a. from model paper 1

$S \rightarrow aACa$
$A \rightarrow B|a$
$B \rightarrow C$
$C \rightarrow cC \mid \varepsilon$

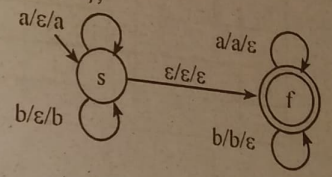
removeEps returns the rule set:

$s \rightarrow aAca \mid aAa \mid aCa \mid aa$
$A \rightarrow B|a$
$B \rightarrow C$
$C \rightarrow cC$

b. Explain Non deterministic PDA and construct an NPDA for the language. $L = \{\omega\omega^R \mid \omega \in \{a,b\}^*\}$
Give the transition diagram and show the trace for a string abaaba. (10 Marks)

Ans. a PDA may be designed to have multiple competing moves from a single configuration. Each node in the tree corresponds to a configuration of M and each path from the root to a leaf node may correspond to one computation that M might perform. Notice that the state the stack. and the remaining input can be different along different paths. As a result, it will not be possible to simulate all paths in parallel. the way we did for NDFSMs.

Pal Even = $\{ ww^R : w \in \{a, b\}^* \}$,



String = abaaba

$(s,abaaba,\varepsilon) \Rightarrow (s,baaba,a) \Rightarrow (s,aaba,ab) \Rightarrow (s,aba,aba) \Rightarrow (f,aba,aba) \Rightarrow (f,ba,ba)$
$\Rightarrow (f,a,a) \Rightarrow (f,\varepsilon,\varepsilon)$ Accepted

Module - 4

7. a. State pumping Lemma for context free language.
Ans. Refer Q.No. 7.a. from Dec18 / Jan 19 **(10 Marks)**

b. Define Turing Machine. Design TM to accept the language $L = \{a^n b^n c^n \mid n > 1\}$. Draw the transition diagram and show the moves made by TM for the string aabbcc. **(10 Marks)**

Ans. Turing machine Refer Q.No. 9a from Model paper 2
Turing machine for Language $L = \{ a^n b^n c^n : n > 0\}$.
Refer Q.No. 8a from Dec17 / Jan18

OR

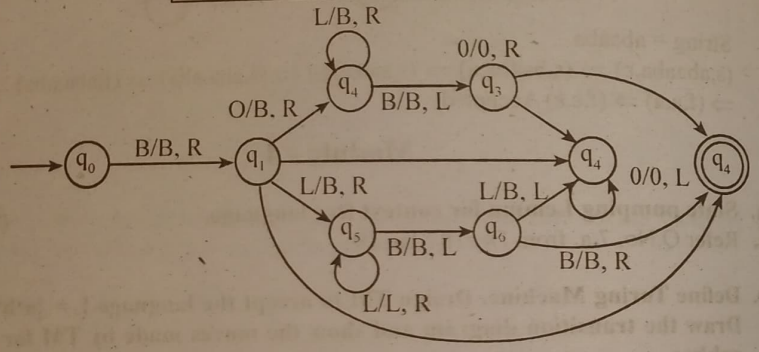
8. a. Explain with a neat diagram the working of TM and design a TM to accept all set of palidrom over {0,1}*. Also show the transition diagram and instantaneous description on string "10101". **(14 Marks)**

Ans. TM to accept strings of palindromer ove {0, 1} is

$M = (Q, \Sigma, V, S, q_0, B, A)$
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$
$\Sigma = \{0, 1\}$
$V = \{a, b, B\}$
q_0 start state
B is Blank character
A is $\{q_7\}$

Transition Table

S	V		
	O	L	B
q_0	–	–	q_1, B, R
q_1	q_1, B, R	q_5, B, R	q_7, B, R
q_2	q_2, O, R	q_2, 1, R	q_3, B, L
q_3	q_3, B, L	–	q_7, B, R
q_4	q_4, O, L	q_4, 1, L	q_1, B, R
q_5	q_5, O, R	q_5, b, R	q_6, B, L
q_6	–	q_6, B, L	q_7, B, R
q_7	–	–	–



10101B⇒B0101B⇒B0101B⇒B0101B⇒B0101B⇒B0101B
↑ ↑ ↑ ↑ ↑ ↑

B0101⇒B010BB⇒B010BB⇒B010BB⇒B010BB⇒B010BB⇒
↑ ↑ ↑ ↑ ↑ ↑

B010BB⇒BB10BB⇒BB10BB⇒BB10BB⇒BB10BB⇒BB1BBB⇒
↑ ↑ ↑ ↑ ↑ ↑

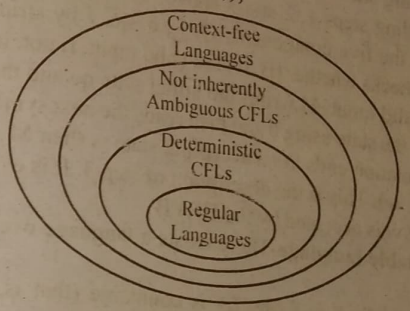
BB1BBB⇒BB1BB⇒BBBB0B
↑ ↑

b. Discuss the relationship between the deterministic context free language and the languages that are not inherently ambigus. (06 Marks)

Ans.

- There exist deterministic context-free languages that are not regular These languages are in the innermost donut in the figure. One example is $A^nB^n = \{a^*b^n: n \geq 0\}$.

- There exist languages that are not in the inner donut (i.e., they are not deterministic). But they are content-free and not inherently ambiguous. Two example* of languages in this second donut are:
  - PalEven = $\{ww^R : w \in \{a, b\}^*\}$. The grammar we showed for it in Example is unambiguous,
  - $\{a^nb^nc^md : n, m \geq 0\} \cup \{a^n b^m c^m e : n, m \geq 0\}$.
- There exist languages that are in the outer donut because they are inherently ambiguous. Two examples are:
  - $(a^ib^jc^k: i, j, k \geq 0$ and $((i = j)$ or $(j = k))\}$
  - $(a^ib^jc^k: i, j, k \geq 0$ and $((i \neq j)$ or $(j \neq k))\}$



Module - 5

9. a. With a neat diagram, explain variants of Turing Machines. (10 Marks)

Ans. Variants of TM

i) 1. Multitape TM: TM with more than one tape Refer Q.No. 9.b. (i) of model paper 1

ii) Non deterministic TM: Refer Q.No. 9.b. Dec 2017 / Jan 2018

b. Explain with example,

(i) Decidability (ii) Decidable languages (iii) Undecidable language. (10 Marks)

Ans. i) Decidability -

- Now these terms are also defined using Turing machines. When a Turing machine reaches a final state, it halts.
- We can also say that a Turing machine M halts when M reaches a state q and a current symbol 'a' to be scanned so that $\delta(q, a)$ is undefined.
- There are TMs that never halt on some inputs in any one of these ways, So we make a distinction between the languages accepted by a TM that halts on all input strings and a TM that never halts on some input strings.
- A language L proper subset of Σ^* is recursively enumerable if there exist turing machine M such thet L=T(M)
- A problem with two answers (Yes/No) is decidable if the corresponding language is recursive. In this case, the language L is also called decidable.
- A problem/language is undecidable if it is not decidable.

ii) Decidable language

we have to construct a TM that always halts and also accepts ADFA We describe the TM M using high level description. Note that a DFM B always ends in some state of B after n transitions for an input string of length n.

We define a TM M as follows:

1. Let B be a DFA and w an input string. (B, w) is an input for the Turing machine M.
2. Simulate B and input H' in the TM M.
3. If the simulation ends in an accepting state of B. then M accepts w. If it ends in a non accepting state of B, then M rejects w. We can discuss a few implementation details regarding steps 1. 2 and 3 above. The input (B, w) for M is represented by representing the five components Q, ∑, δ, qo, f by strings of ∑* and input string W ∈ ∑*. M checks whether (B, w) is a valid input. If not, it rejects (B, w) and halts. If (B, w) is a valid input, M writes the initial state qo and the leftmost input symbol of w. It updates the state using 0 and then reads the next symbol in w. This explains step 2. If the simulation ends in an accepting state w then M accepts (B, w). Otherwise, M rejects (B, w). This is the description of step 3. It is evident that M accepts (B,w) if and only if w is accepted by the DFA B

iii) UnDecidable language-There exists a language over 2: that is not recursively enumerable.

that L = T(M). As Σ is finite, Σ* is countable (that is, there exists a one-to-one correspondence between Σ* and #).

As a Turing machine M is a 7-tuple (Q, Σ, Φ, δ, q_0, b, F) and each member of the 7-tuple is a finite set, M can be encoded as a string. So the set I of all TMs is countable.

Let *L* be the set of all languages over Σ. Then a member of *L* is a subset of Σ* (Note that Σ* is infinite even though Z is finite). We show that *L* is uncountable (that is, an infinite set not in one-to correspondence with N).

We prove this by contradiction. If *L* were countable then *L* can be written as a sequence $\{L_1, L_2, L_3, \ldots\}$. We write Σ* as a sequence $\{w_1, w_2, w_3, \ldots\}$. So L_i can be represented as an infinite binary sequence $x_{i1} x_{i2} x_{i3} \ldots$ where

$$x_{ij} = \begin{cases} 1 & \text{if } w_j \in L_i \\ 0 & \text{otherwise} \end{cases}$$

Using this representation we write L_i as an infinite binary sequence,

$L_1 : x_{11}x_{12}x_{13} \ldots x_{1j} \ldots$

$L_2 : x_{21}x_{22}x_{23} \ldots x_{2j} \ldots$

:

$L_i : x_{i1}x_{i2}x_{i3} \ldots x_{ij} \ldots$

We define a subset L of Σ* by the binary sequence $y_1y_2y_3 \ldots$ where $y_i = 1 - X_{ii}$. If $x_{ii} = 0$, $y_i = 1$ and if $x_{ii} = 1$, $y_i = 0$. Thus according to our assumption the subset L of Σ* represented by the infinite binary sequence $y_1, y_2, y_3 \ldots$ should be L_k for some natural number k. But $L \neq L_k$. since $w_k \in L$, if and only if $w_k \notin L_k$. This contradicts

our assumption that *L* is countable. Therefore *L* is uncountable. As I is countable. *L* should have some members not corresponding to any TM in I. This proves me existence of a language over Σ mat is not recursively enumerable.

OR

10. a. Discuss Halting problem and post correspondence problem with respect to TM.

Ans. Halting problem Refer Q.No. 10.b. of Dec 17 / Jan 18 **(10 Marks)**

Post correspondence problem - Refer Q.No. 10.c. of Dec17 / Jan 18

b. Define non-deterministic TM and prove that there in a deterministic TM 'M' such that, T(M) = T(M_1)

Ans. Non deterministic turing machine M **(10 Marks)**

A nondeterministic Turing, machine is a 7-tuple {Q, Σ, P, δ, q^0, b, F) where

1. Q is a finite nonempty set of states
2. T is a finite nonempty set of tape symbols
3. b ∈ T is called the blank symbol
4. Σ is a nonempty subset of P. called the set of input symbols. We assume b ∉ Σ.
5. q_0 is the initial state
6. F ⊆ Q is the set of final states
7. δ is a partial function from Q × T into the power set of Q × T × {L, R}.

Theorem : If M is a nondeterministic TM, there is a deterministic TM Ml such that T(M) = T(M1)

Proof: We construct Ml as a multitape TM, Each symbol in the input string leads to a change in ID. Ml should be able to reach all IDs and stop when an ID containing a final state is reached- So the first tape is used to store IDs of M as a sequence and also the state of M. These IDs are separated by the symbol *

All IDs to the left of the current one have been explored already and so can be ignored subsequently. Note that the current ID is decided by the current input symbol of w