Theory of Computation

Course Title: Theory of Computation — Full Marks: 60 + 20 + 20
Course No: CSC257 — Pass Marks: 24 + 8 + 8
Nature of the Course: Theory + Lab — Credit Hrs: 3
Semester: IV

Course Objectives

This course presents a study of Finite State Machines and their languages. It covers the details of finite state automata, regular expressions, context free grammars. More, the course includes design of the Push-down automata and Turing Machines. The course also includes basics of undecidabilty and intractability.

Course Description

The main objective of the course is to introduce concepts of the models of computation and formal language approach to computation. The general objectives of this course are to, introduce concepts in automata theory and theory of computation, design different finite state machines and grammars and recognizers for different formal languages, identify different formal language classes and their relationships, determine the decidability and intractability of computational problems.

Course Details

Unit I: Basic Foundations **(3 Hrs.)**

1.1. Review of Set Theory, Logic, Functions, Proofs

1.2. Automata, Computability and Complexity: Complexity Theory, Computability Theory, Automata Theory

1.3. Basic concepts of Automata Theory: Alphabets, Power of Alphabet, Kleen Closure Alphabet, Positive Closure of Alphabet, Strings, Empty String, Substring of a string, Concatenation of strings, Languages, Empty Language

Unit II: Introduction to Finite Automata **(8 Hrs.)**

2.1 Introduction to Finite Automata, Introduction of Finite State Machine

2.2 Deterministic Finite Automata (DFA), Notations for DFA, Language of DFA, Extended Transition Function of DFA Non-Deterministic Finite Automaton (NFA), Notations for NFA, Language of NFA, Extended Transition

2.3 Equivalence of DFA and NFA, Subset-Construction

2.4 Method for reduction of NFA to DFA, Theorems for equivalence of Language accepted by DFA and NFA

2.5 Finite Automaton with Epsilon Transition (ε - NFA), Notations for ε - NFA, Epsilon Closure of a State, Extended Transition Function of ε – NFA, Removing Epsilon Transition using the concept of Epsilon Closure, Equivalence of NFA and ε –NFA, Equivalence of DFA and ε – NFA

2.6 Finite State Machines with output: Moore machine and Mealy Machines

Unit III: Regular Expressions **(6 Hrs.)**

3.1 Regular Expressions, Regular Operators, Regular Languages and their applications, Algebraic Rules for Regular Expressions

3.2 Equivalence of Regular Expression and Finite Automata, Reduction of Regular Expression to ε – NFA, Conversion of DFA to Regular Expression

3.3 Properties of Regular Languages, Pumping Lemma, Application of Pumping Lemma, Closure Properties of Regular Languages over (Union, Intersection, Complement) Minimization of Finite State Machines: Table Filling Algorithm

Unit IV: Context Free Grammar **(9 Hrs.)**

4.1 Introduction to Context Free Grammar (CFG), Components of CFG, Use of CFG, Context Free Language (CFL)

4.2 Types of derivations: Bottomup and Topdown approach, Leftmost and Rightmost, Language of a grammar

4.3 Parse tree and its construction, Ambiguous grammar, Use of parse tree to show ambiguity in grammar

4.4 Regular Grammars: Right Linear and Left Linear, Equivalence of regular grammar and finite automata

4.5 Simplification of CFG: Removal of Useless symbols, Nullable Symbols, and Unit Productions, Chomsky Normal Form (CNF), Greibach Normal Form (GNF), Backus- Naur Form (BNF)

4.6 Context Sensitive Grammar, Chomsky Hierarchy Pumping Lemma for CFL, Application of Pumping Lemma, Closure Properties of CFL

Unit V: Push Down Automata **(7 Hrs.)**

5.1 Introduction to Push Down Automata (PDA), Representation of PDA, Operations of
PDA, Move of a PDA, Instantaneous Description for PDA

5.2 Deterministic PDA, Non Deterministic PDA, Acceptance of strings by PDA, Language of PDA

5.3 Construction of PDA by Final State , Construction of PDA by Empty Stack,

5.4 Conversion of PDA by Final State to PDA accepting by Empty Stack and vice-versa, Conversion of CFG to PDA, Conversion of PDA to CFG

Unit VI: Turing Machines (10 Hrs.)

6.1 Introduction to Turing Machines (TM), Notations of Turing Machine, Language of a Turing Machine, Instantaneous Description for Turing Machine, Acceptance of a string by a Turing Machines

6.2 Turing Machine as a Language Recognizer, Turing Machine as a Computing Function, Turing Machine with Storage in its State, Turing Machine as a enumerator of stings of a language, Turing Machine as Subroutine

6.3 Turing Machine with Multiple Tracks, Turing Machine with Multiple Tapes, Equivalence of Multitape-TM and Multitrack-TM, Non-Deterministic Turing Machines, Restricted Turing Machines: With Semi-infinite Tape, Multistack Machines, Counter Machines

6.4 Curch Turing Thesis, Universal Turing Machine, Turing Machine and Computers, Encoding of Turing Machine, Enumerating Binary Strings, Codes of Turing Machine, Universal Turing Machine for encoding of Turing Machine

Unit VII: Undecidability and Intractability **(5 Hrs.)**

7.1 Computational Complexity, Time and Space complexity of A Turing Machine, Intractability

7.2 Complexity Classes, Problem and its types: Absract, Decision, Optimization

7.3 Reducibility, Turing Reducible, Circuit Satisfiability, Cook's Theorem,

7.4 Undecidability, Undecidable Problems: Post's Correspondence Problem, Halting Problem and its proof, Undecidable Problem about Turing Machines

Laboratory Works:

The laboratory work consists of design and implementation of finite state machines like DFA,
NFA, PDA, and Turing Machine. Students are highly recommended to construct Tokenizers/ Lexers over/for some language. Students are advised to use regex and Perl (for using regular expressions), or any other higher level language for the laboratory works.

Text Books:

1. **John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, Introduction to Automata Theory, Languages, and Computation, 3rd Edition, Pearson - Addison-Wesley.**

Reference Books:

1. Harry R. Lewis and Christos H. Papadimitriou, Elements of the Theory of Computation, 2nd Edition, Prentice Hall.
2. Michael Sipser, Introduction to the Theory of Computation, 3rd Edition, Thomson Course Technology
3. Efim Kinber, Carl Smith, Theory of Computing: A Gentle introduction, Prentice- Hall.
4. John Martin, Introduction to Languages and the Theory of Computation, 3rd Edition, Tata McGraw Hill.
5. Kenneth H. Rosen, Discrete Mathematics and its Applications to Computers Science, WCB/Mc-Graw Hill.

TRIBHUVAN UNIVERSITY

Institution of Science and Technology

Course Title: Theory of Computation
Course No: CSC257
Level: B. Sc CSIT Second Year/ Fourth Semester

Full Marks: 60
Pass Marks: 24
Time: 3 Hrs

T.U. MODEL QUESTION-ANSWERS SET I

Section A

Long Answer Questions

Attempt any two questions. **[2*10=20]**

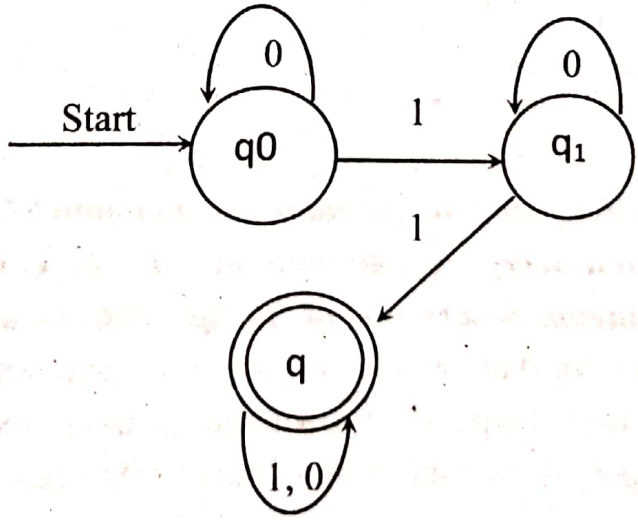
1. ***Define the extended transition function of DFA. Draw a DFA accepting language L= {1^n\|n=2, 3, 4…….}. Show acceptance of strings 1110011 and 1110 using extended transition function.***

[2+4+4]

Answer: The extended transition function of DFA, denoted by $\hat{d}$ is a transition function that takes two arguments as input, one is the state q of Q and another is a string w ∈ Σ*, and generates a state p ∈ Q. This state p is that the automaton reaches when starting in state q & processing the sequence of inputs w.

i.e. $(q, w) = p$

Example: The DFA accepting language L= {1^n\|n=2, 3, 4…….} is given below;



Showing acceptance of string 1110011 by using extended transaction function as,

$(q_0, 1110011)$

$= \delta(\hat{\delta}(q_0, 111001), 1)$

$= \delta(\delta(\hat{\delta}(q_0, 1110), 0), 1)$

$= \delta(\delta(\delta(\hat{\delta}(q_0, 1110), 0), 1), 1)$

$= \delta(\delta(\delta(\hat{\delta}(q_0, 111), 0), 0), 1), 1)$

$= \delta(\delta(\delta(\delta(\hat{\delta}(q_0, 11), 1), 0), 0), 1), 1)$

$= \delta(\delta(\delta(\delta(\delta(\hat{\delta}(q_0, 1), 1), 1), 0), 0), 1), 1)$

$= \delta(\delta(\delta(\delta(\delta(\delta(\hat{\delta}(q_0, \omega), 1), 1), 1), 0), 0), 1), 1)$

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

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

$= \delta(\delta(\delta(\delta(q_2, 1), 0), 0), 1), 1)$

$= \delta(\delta(\delta(q_2, 0), 0), 1), 1)$

$= \delta(\delta(q_2, 0), 1), 1)$

$= \delta(\delta(q_2, 1), 1)$

$= \delta(q_2, 1)$

$= q_2$ which is final state so accepted

Showing acceptance of string 1110 by using extended transaction function as,

$\hat{\delta}(q_0, 1110)$

$= \delta(\hat{\delta}(q_0, 111), 0)$

$= \delta(\delta(\hat{\delta}(q_0, 11), 1), 0)$

$= \delta(\delta(\delta(\hat{\delta}(q_0, 1), 1), 1), 0)$

$= \delta(\delta(\delta(\delta(\hat{\delta}(q_0, \varepsilon), 1), 1), 1), 0)$

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

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

$= \delta(\delta(q_2, 1), , 0)$

$= \delta(q_2, 0)$

$= q_2$ which is final state so accepted

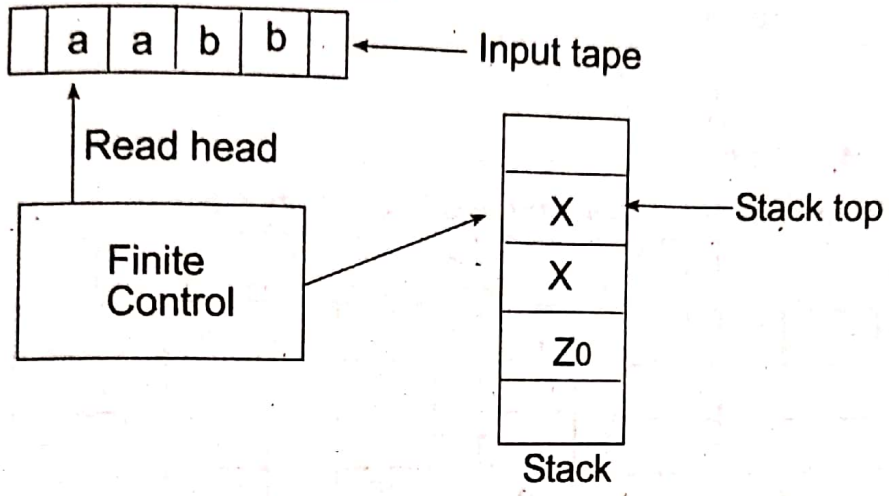
2. ***What is deterministic pushdown automaton? Configure a pushdown automaton accepting the language, L= {w C w^R\|w Є (a, b)*}. Show instantaneous description of strings abbCbba and baCba.*** *[2+4+4]*

Answer: Pushdown Automata is a finite automata with extra memory called stack which helps Pushdown automata to recognize Context Free Languages. A Pushdown Automata (PDA) can be formally defined as follows:

P = (Q, ∑, Γ, δ, q0, Z0, F) where,

- Q is the set of states
- ∑ is the set of input symbols
- Γ is the set of pushdown symbols (which can be pushed and popped from stack)
- q0 is the initial state

- Z0 is the initial pushdown symbol (which is initially present in stack)
- F is the set of final states
- δ is a transition function which maps Q x {Σ ∪ ∈} x Γ into Q x Γ*. In a given state, PDA will read input symbol and stack symbol (top of the stack) and move to a new state and change the symbol of stack.



Pushdown automaton accepting the language, L= {w C w^R | w ∈ (a, b)*}

By analysis of language it must satisfy following strings

{abcba, bacab, ababcbaba, ...}

Let push down automata be, P = (Q, ∑, Γ, δ, q_0, Z_0, F)

Where, Q= {q_0, q_1, qf}

∑ = {a, b}

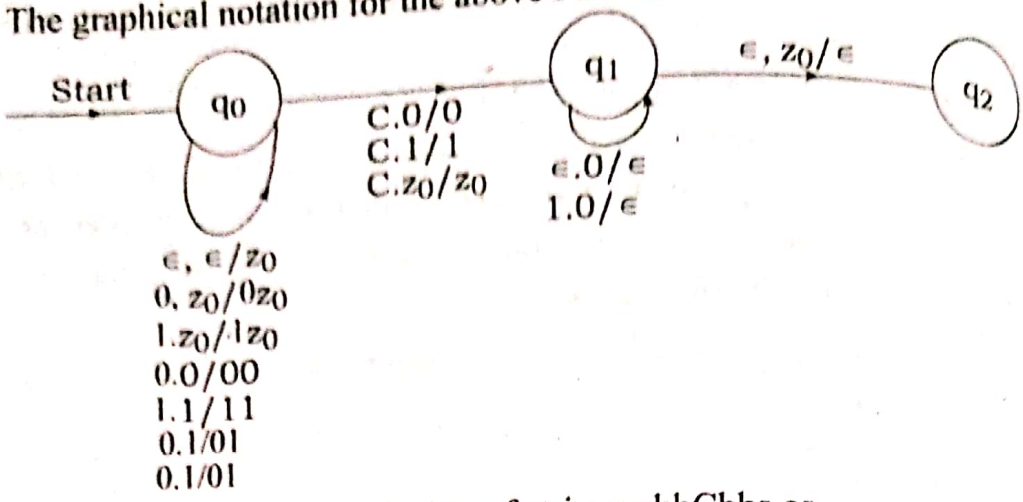
Γ = {a, b, z_0}

qf = Final state

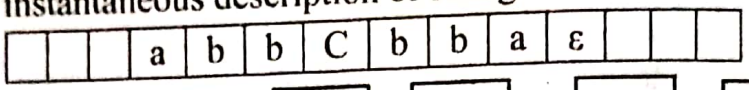
And transaction function δ can be defined as follows,

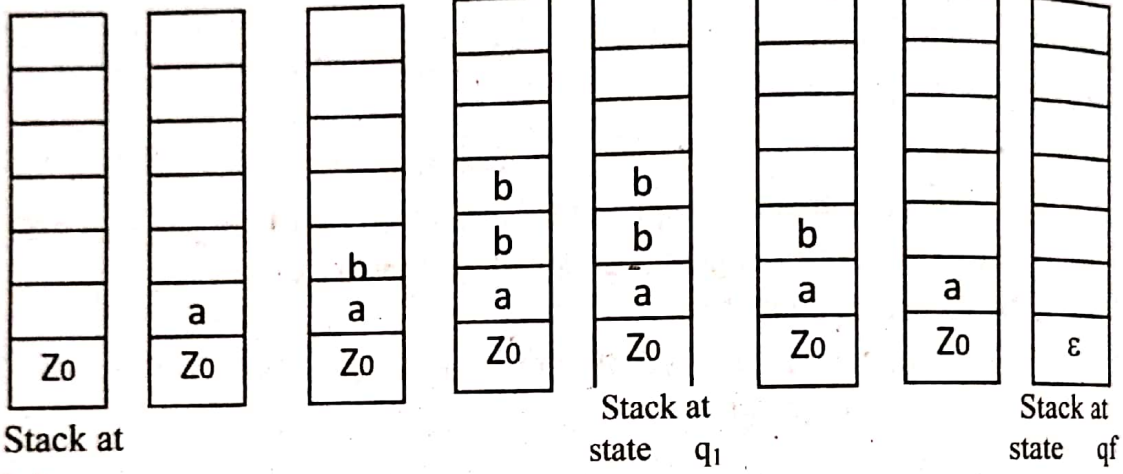
1. δ(q_0, ε , ε) = (q0, Z0)
2. δ(q_0, a, Z_0) = (q0, aZ0)
3. δ(q0, b, Z0) = (q0, bZ0)
4. δ(q0, a, a) = (q0, aa)
5. δ(q0, b, b) = (q0, bb)
6. δ(q0, a, b) = (q0, ab)
7. δ(q0, b, a) = (q0, ba)
8. δ(q0, c, a) = (q1, a) //Change the state
9. δ(q0, c, b) = (q1, b) // change the state
10. δ(q0, c, z0) = (q1, z0) //change the state
11. δ(q1, a, a) = (q1, ε)
12. δ(q1, b, b) = (q1, ε)
13. δ(q1, ε, z0) = (qf, ε)

The graphical notation for the above PDA is:



Show instantaneous description of strings abbCbba as





Stack at state q_0 | Stack at state q_1 | Stack at state qf

$(q_0, abbcbba, z_0)$ $= (q_0, bbcbba, az_0)$
$= (q_0, bcbba, baz_0)$
$= (q_0, cbba, bbaz_0)$
$= (q_1, bba, bbaz_0)$
$= (q_1, ba, baz_0)$
$= (q_1, a, az_0)$
$= (q_1, \varepsilon, z_0)$
$= (qf, \varepsilon, \varepsilon)$ accepted

Show instantaneous description of strings baCba

$(q_0, bacba, z_0)$ $= (q_0, acba, bz_0)$
$= (q_0, cba, abz_0)$
$= (q_1, ba, abz_0)$
$= (q_1, ba, abz_0)$ not accepted

3. ***How a Turing Machine works? Construct a Turing Machine accepting the language, L= {(")"}. Also show the transition diagram of the machine. Illustrate whether a string (( )) is accepted by the Turing Machine or not.*** *[2+6+2]*

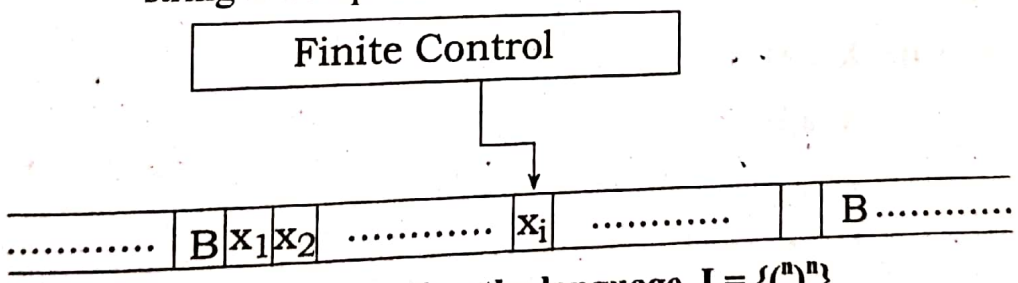
Answer: Turing Machine was invented by Alan Turing in 1936 and it is used to accept Recursive Enumerable Languages (generated by Type-0 Grammar). A Turing machine consists of a tape of infinite length on

which read and writes operation can be performed. The tape consists of infinite cells on which each cell either contains input symbol or a special symbol called blank. It also consists of a head pointer which points to cell currently being read and it can move in both directions. A TM is expressed as a 7-tuple M = (Q, Σ, Γ, δ, q0, B, F) where:

- Q = the finite set of states of the finite control
- Σ = the finite set of input symbols
- Γ = the complete set of tape symbols Σ is always subset of Γ.
- q_0 = the start state; q0 ε Q
- B = the blank symbol; B ε Γ but B does not belong to Σ.
- F = the set of final or accepting states; F is subset of Q
- δ = the transition function defined by

Q × Γ→ Q × Γ × (R, L, S); where R, L, S is the direction of movement of head left, or right or stationary. I.e. d(q, x) = d(p, Y, D); which means T M in state q and current tape symbol x, moves to next state P, replacing tape symbol x with Y and move the head either direction or remains at same cell of input tape.

- q_0 is the initial state
- F is the set of final states. If any state of F is reached, input string is accepted.



Turing Machine accepting the language, L= {(^{n})^n}

We are going to design a Turing machine which accepts a set of strings in which every string starts with '(' followed by any number of ')' as '('. No '(' is encountered after first ')' is read. It will not accepts the strings like {ε, (, ((, ((())), ()),),)), ((() and so on }

Let required Turing machine is:

M = (Q, Σ, Γ, δ, q_0, B, F)

Q= {q_0, q_1, q_2, q_3, qf}

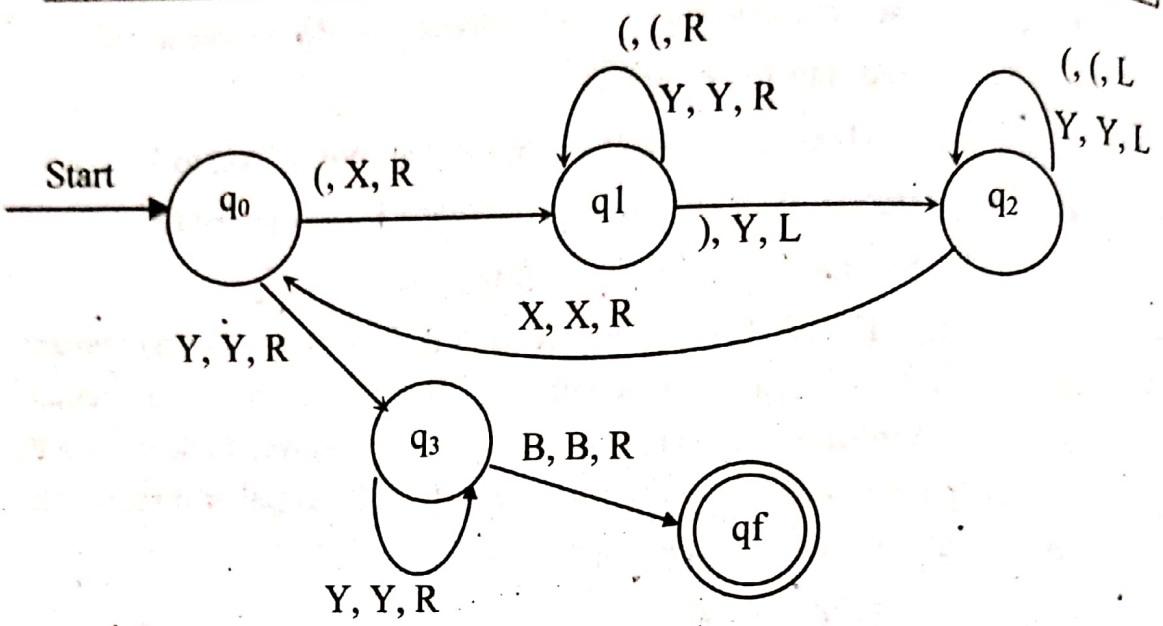
Σ = {(,)}

Γ = {(,), X, Y}

A

and δ can be defined as

	(	)	X	Y	B
q_0	(q_1, X, R)			(q_3, Y, R)	
q_1	$(q_1, (, R)$	(q_2, Y, L)		(q_1, Y, R)	
q_2	$(q_2, (, L)$		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	(qf, B, R)
qf					Accept



Illustrate whether a string (( )) is accepted by the Turing Machine or not:

q_0 (( )) = X q_1 ())

= X (q_1))

= X (q_1))

= X q_2(Y)

= q_2 X (Y)

= X q_0 (Y)

= X X q_1Y)

= X X Y q_1)

= X X q_2YY

= X q_2 X YY

= X X q_0 YY

= X X Y q_3Y

= X X Y Y q_3 B

= qf Accepted

Section B

Short Answer Questions

Attempt any eight questions. [8×5=40]

4. When a grammar is said to be in CNF? Convert following grammar to CNF; [1+4]

S→ 1A | 0B | ϵ
A→ 1AA | 0S | 0
B→ 0BB | 1 |A
C→CA | CS

Answer: A context free grammar G = (V, T, P, S) is said to be in Chomsky's Normal Form (CNF) if every production in G are in one of the two forms;

A→ BC and

A→ a where {A, B, C} ε V and a ε T

Thus a grammar in CNF is one which should not have;

- Є-production
- Unit production
- Useless symbols.

Second part:

Convert following grammar to CNF

S→ 1A | 0B | ϵ
A→ 1AA | 0S | 0
B→ 0BB | 1 |A
C→CA | CS

At first remove ε-production as,

S→ 1A | 0B
A→ 1AA | 0S | 0
B→ 0BB | 1 |A
C→CA | CS |C

Now remove unit production as,

S→ 1A | 0B
A→ 1AA | 0S | 0
B→ 0BB | 1 |1AA |0S | 0
C→CA | CS

Also remove useless symbols as,

Since C is unreachable from starting symbol S so last production is useless,

S→ 1A | 0B
A→ 1AA | 0S | 0
B→ 0BB | 1 |1AA |0S | 0

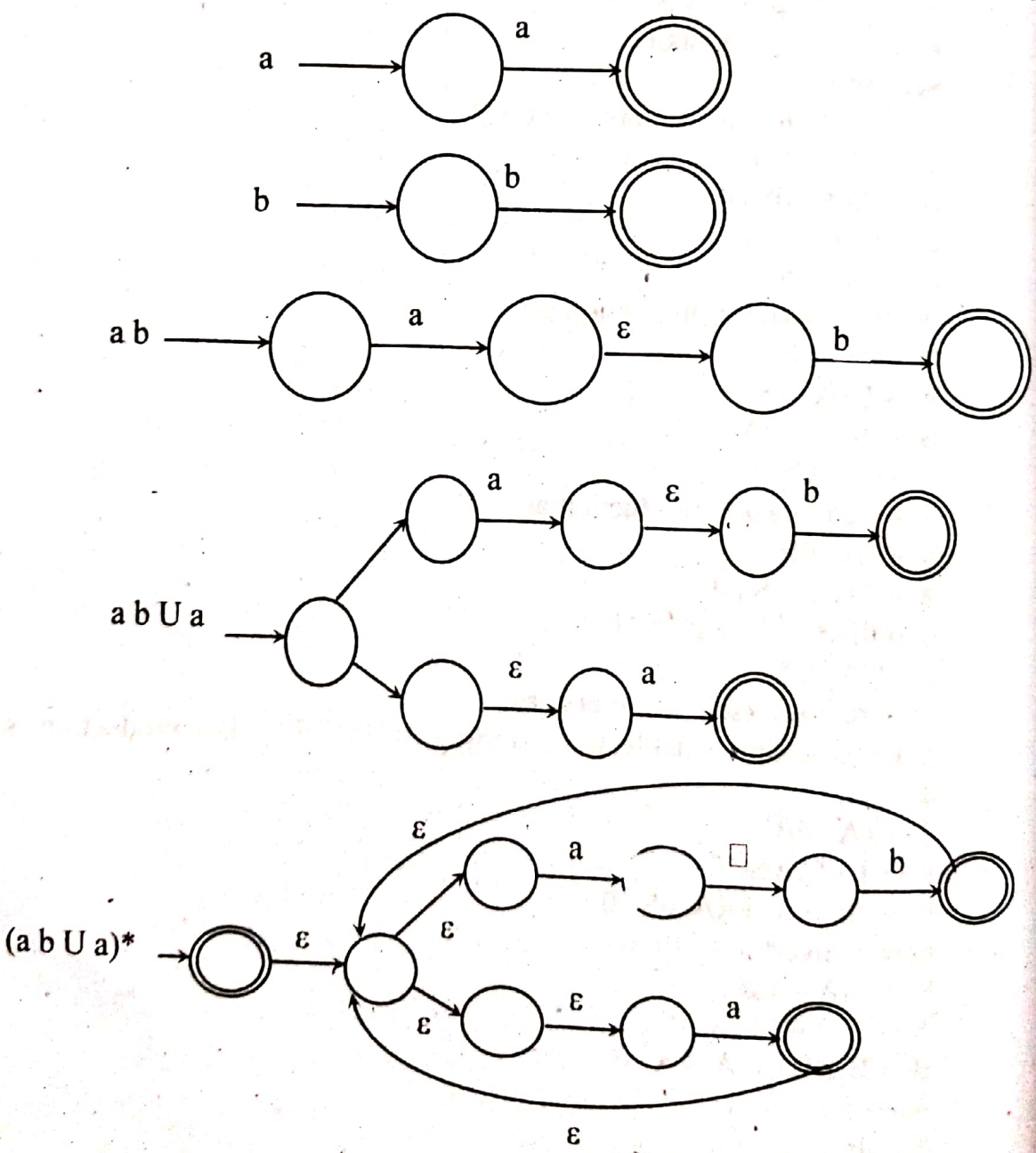
Now convert to CNF as

S→ A_1A | B_1B
A→ A_1AA | B_1S | 0
B→ B_1BB | 1 |A_1AA | 0
A_1→ 1
B_1→0

5. ***Define epsilon NFA. Configure equivalent epsilon NFA for the regular expression (ab U a)*.*** *[1+4]*

Answer: Nondeterministic finite automaton with ε-moves (NFA-ε) is a further generalization to NFA. This automaton replaces the transition function with the one that allows the empty string ε as a possible input. The transitions without consuming an input symbol are called ε-transitions. In the state diagrams, they are usually labeled with the Greek letter ε. ε-transitions provide a convenient way of modeling the systems whose current states are not precisely known: i.e., if we are modeling a system and it is not clear whether the current state (after processing some input string) should be q or q', then we can add an ε-transition between these two states, thus putting the automaton in both states simultaneously.

Epsilon NFA for the regular expression (ab U a)*



6. ***Differentiate Kleene Closure from Positive Closure. For $\sum = \{0, 1\}$, compute $\sum^*$ and $\sum 2$.***

Answer: The Kleene closure denoted by $\sum^*$, is a unary operator on a set of symbols or strings, $\sum$, that gives the infinite set of all possible strings of all possible lengths over $\sum$ including ε.

Representation – $\sum^* = \sum 0 \cup \sum 1 \cup \sum 2 \cup \ldots\ldots$ where $\sum p$ is the set of all possible strings of length p.

Example – If $\sum = \{a, b\}$, $\sum^* = \{\varepsilon, a, b, aa, ab, ba, bb\ldots\}$

The positive closure denoted by $\sum+$ is the infinite set of all possible strings of all possible lengths over $\sum$ excluding ε.

Representation – $\sum+ = \sum 1 \cup \sum 2 \cup \sum 3 \cup \ldots\ldots$

$\sum+ = \sum^* - \{\varepsilon\}$

Example: If $\sum = \{a, b\}$, $\sum+ = \{a, b, aa, ab, ba, bb...\}$

For $\sum = \{0, 1\}$, compute $\sum^*$ and $\sum 2$

$\sum^* = \{\varepsilon, 0, 1, 00, 11, 10, 01, 000, 111, 101, 010, 1010\ldots\}$

$\sum 2 = \{00, 11, 10, 01\}$

7. ***Write the regular expression over {0, 1} for strings*** *[2.5+2.5]*

- ***Not ending with 0.***
- ***Of length at least 3 that ends with 00.***

Answer: Not ending with 0.

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

Of length at least 3 that ends with 00.

RE: $(0+1)^+ 00$

8. ***What is undecidable problem? Define Post's Correspondence Problem with an example.*** *[1+4]*

Answer: In computability theory and computational complexity theory, an undecidable problem is a decision problem for which it is proved to be impossible to construct an algorithm that always leads to a correct yes-or-no answer. The halting problem is an example: it can be proven that there is no algorithm that correctly determines whether arbitrary programs eventually halt when run.

The Post Correspondence Problem (PCP)

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

- $M = (x_1, x_2, x_3, \ldots\ldots, xn)$
- $N = (y_1, y_2, y_3, \ldots\ldots, yn)$

We can say that there is a Post Correspondence Solution, if for some $i_1, i_2 \ldots ik$, where $1 \leq ij \leq n$, the condition $xi1 \ldots\ldots xik = yi_1 \ldots\ldots yik$ satisfies.

Example 1: Find whether the lists M = (abb, aa, aaa) and N = (bba, aaa, aa) have a Post Correspondence Solution?

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.

Example 2: Find whether the lists M = (ab, bab, bbaaa) and N = (a, ba, bab) have a Post Correspondence Solution?

Solution

	x_1	x_2	x_3
M	ab	bab	bbaaa
N	a	ba	bab

In this case, there is no solution because$|x_2x_1x_3| \neq |y_2y_1y_3|$ (Lengths are not same)

Hence, it can be said that this Post Correspondence Problem is undecidable.

9. ***How pumping lemma can be used to prove that any language is not a regular language? Show that language, L = {0^n 1^r\|n ≥0} is not a regular language.*** *[4+1]*

Answer: Method to prove that a language L is not regular

- At first, we have to assume that L is regular.
- So, the pumping lemma should hold for L.
- Use the pumping lemma to obtain a contradiction:

✓ Select w such that $|w| \geq c$

✓ Select y such that $|y| \geq 1$

✓ Select x such that $|xy| \leq c$

✓ Assign the remaining string to z.

✓ Select k such that the resulting string is not in L.

Hence L is not regular.

Pumping Lemma is to be applied to show that certain languages are not regular. It should never be used to show a language is regular.

- If L is regular, it satisfies Pumping Lemma.
- If L does not satisfy Pumping Lemma, it is non-regular.

Let L be a regular language. Then there exists a constant 'c' such that for every string w in L:

$|w| \geq c$

We can break w into three strings, w = xyz, such that:

- $|y| > 0$
- $|xy| \leq c$
- For all $k \geq 0$, the string xy^kz is also in L.

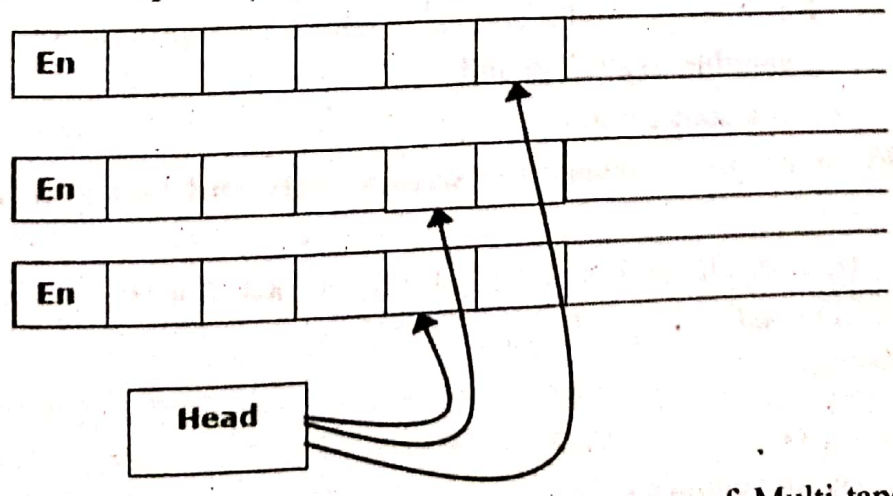
Show that language, $L = \{0^n 1^n | n \geq 0\}$ is not a regular language

Answer:

- At first, we assume that L is regular and n is the number of states.
- Let $w = 0^n1^n$. Thus $|w| = 2n \geq n$.
- By pumping lemma, let w = xyz, where $|xy| \leq n$.
- Let $x = 0^p$, $y = 0^q$, and $z = 0^r1^n$, where $p + q + r = n$, $p \neq 0$, $q \neq 0$, $r \neq 0$. Thus $|y| \neq 0$.
- Let k = 2. Then $xy^2z = 0^p0^{2q}0^r1^n$.
- Number of as = $(p + 2q + r) = (p + q + r) + q = n + q$
- Hence, $xy^2z = 0^{n+q}1^n$. Since $q \neq 0$, xy^2z is not of the form 0^n1^n.
- Thus, xy^2z is not in L. Hence L is not regular.

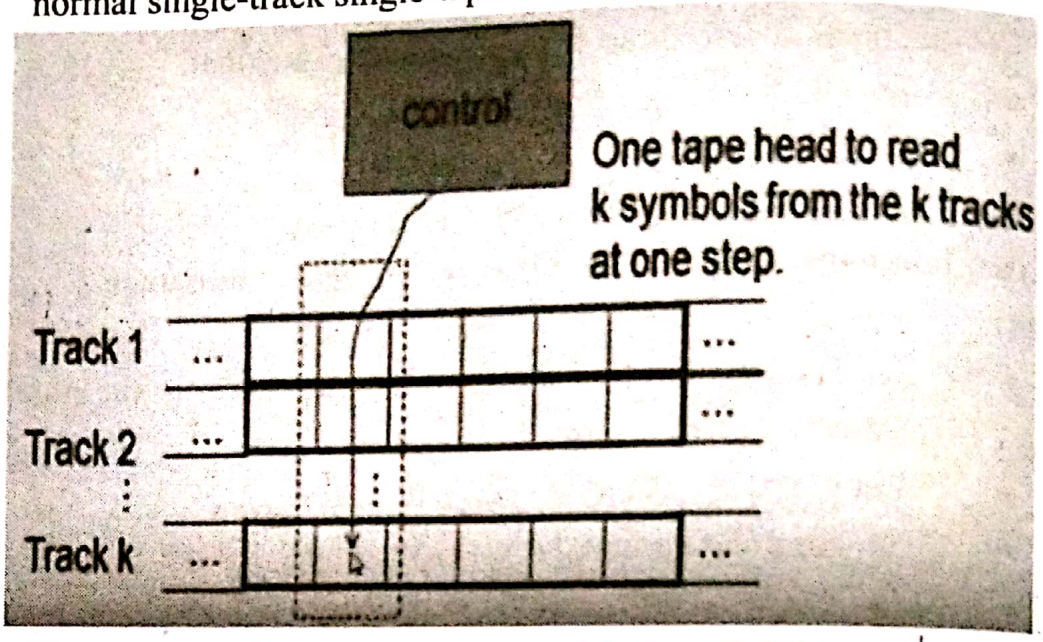
10. ***Discuss how Turing Machine with multiple tracks differs from a Turing Machine with multiple tapes.***

Answer: Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head. Each head can move independently of the other heads. Initially the input is on tape 1 and others are blank. At first, the first tape is occupied by the input and the other tapes are kept blank. Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads.



Multi-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.



11. How context free grammars are defined? Write a context free grammar over {0, 1}, where the strings start and end with the same symbol. *[2+3]*

Answer:

Context-free grammars (CFGs) are used to describe context-free languages. A context-free grammar is a set of recursive rules used to generate patterns of strings. A context-free grammar can describe all regular languages and more, but they cannot describe all possible languages.

A context-free grammar can be described by a four-element tuple (V, T, P, S) where

- V is a finite set of variables (which are non-terminal);
- T is a finite set of terminal symbols;
- P is a set of production rules where each production rule maps a variable to a string and
- S is a start symbol.

CFG over {0, 1}, where the strings start and end with the same symbol

L = {ε, 1, 0, 11, 00, 101, 010, 111, 000, 10001, 00010,...}

Let CFG be G = (V, T, P, S)

Where,

V={S}

T = {0, 1}

P = {S→0|1|0S0|1S1|ε}

And S=S

12. ***What is halting problem? How can you argue that halting problem is undecidable?*** *[1+4]*

Answer: In computability theory, the halting problem is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running (i.e., halt) or continue to run forever. Alan Turing proved in 1936 that a general algorithm to solve the halting problem for all possible program-input pairs cannot exist. A key part of the proof was a mathematical definition of a computer and program, which became known as a Turing machine; the halting problem is undecidable over Turing machines. Turing's proof is one of the first cases of decision problems to be concluded. The theoretical conclusion of not solvable is significant to practical computing efforts, defining a class of applications which no programming invention can possibly perform perfectly.

The halting problem is a decision problem about properties of computer programs on a fixed Turing-complete model of computation, i.e., all programs that can be written in some given programming language that is general enough to be equivalent to a Turing machine. The problem is to determine, given a program and an input to the program, whether the program will eventually halt when run with that input. In this abstract framework, there are no resource limitations on the amount of memory or time required for the program's execution; it can take arbitrarily long and use an arbitrary amount of storage space before halting. The question is simply whether the given program will ever halt on a particular input.

For example, in pseudocode, the program

while (true) continue

Does not halt; rather, it goes on forever in an infinite loop. On the other hand, the program.

print "Hello, world!"

Does halt.

While deciding whether these programs halt is simple, more complex programs prove problematic. One approach to the problem might be to run the program for some number of steps and check if it halts. But if the program does not halt, it is unknown whether the program will eventually halt or run forever.

MODEL QUESTIONS-ANSWERS SET II

Course Title: Theory of Computation
Course No: CSC257
Level: B. Sc CSIT Second Year/ Fourth Semester

Full Marks: 60
Pass Marks: 24
Time: 3 Hrs

Section A

Long Answer Questions

Attempt any two questions.

1. ***Define DFA. How it is differ from NFA? Draw a DFA over {a, b} that accepts the strings ending with abb. Also draw NFA over {a, b} that accepts strings having aa as substring.*** *[2+4+4]*

Answer: Definition of Deterministic finite state automata (DFA)

A deterministic finite automaton (DFA) also known as deterministic finite acceptor (DFA), deterministic finite state machine (DFSM), or deterministic finite state automaton (DFSA)—is a finite-state machine that accepts or rejects strings of symbols and only produces a unique computation (or run) of the automaton for each input string. A deterministic finite automaton is defined by a quintuple (5-tuple) as $(Q, \Sigma, \delta, q0, F)$

Where,

Q = Finite set of states,

Σ = Finite set of input symbols,

δ = A transition function that maps $Q \times S \rightarrow Q$

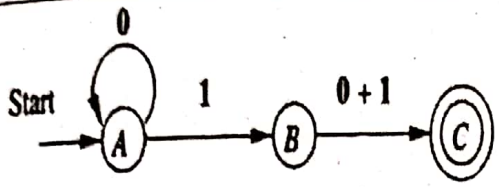
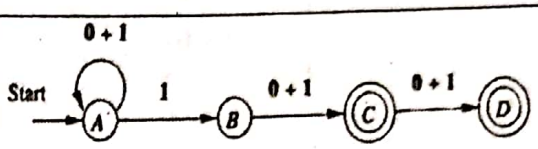
$q0$ = A start state; $q0 \in Q$

F = Set of final states; $F \subseteq Q$.

A transaction function δ that takes as arguments a state and an input symbol and returns a state. In our diagram, δ is represented by arcs between states and the labels on the arcs.

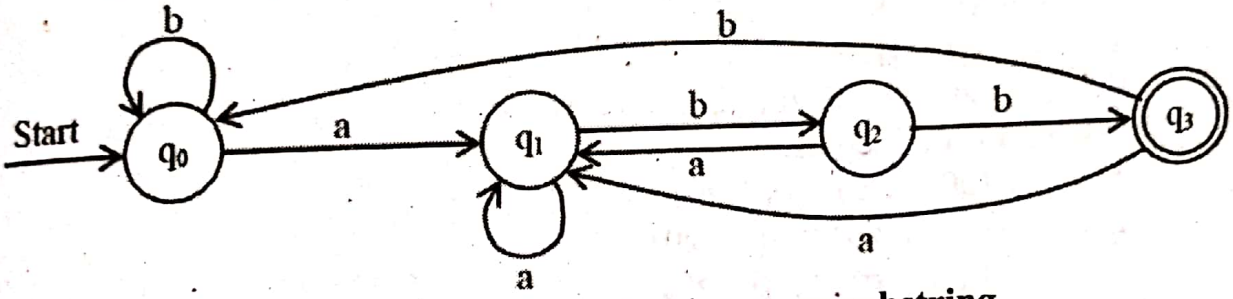
Difference between Deterministic Finite Automata and the Non deterministic Finite Automata ((DFA Vs NFA):

S.N.	DFA	NFA
1.	For Every symbol of the alphabet, there is only one state transition in DFA.	We do not need to specify how does the NFA react according to some symbol.
2.	DFA cannot use Empty String transition.	NFA can use Empty String transition.
3.	DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
4.	DFA will reject the string if it ends at other than accepting state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.

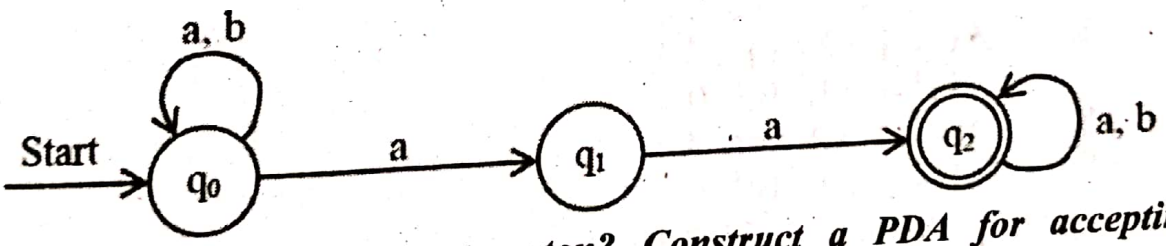
5.	Backtracking is allowed in DFA.	Backtracking is not always allowed in NFA.
6.	DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
7.	DFA will reject the string if it end at other than accepting or final state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.
8.	DFA is more difficult to construct.	NFA is easier to construct.
9.		

[2*10=20]

DFA over {a, b} that accepts the strings ending with abb



NFA over {a, b} that accepts strings having aa as substring.



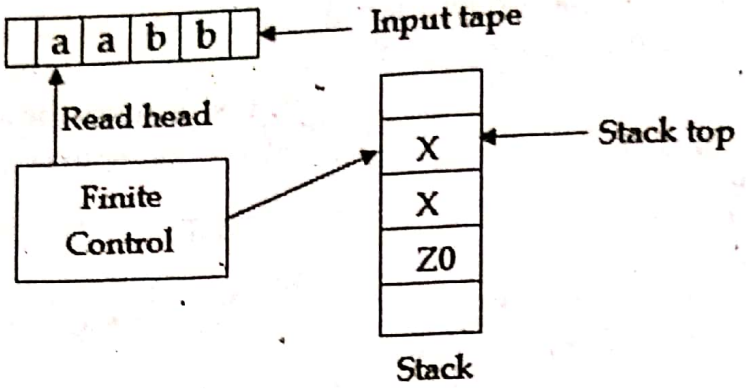
2. ***Define pushdown automaton? Construct a PDA for accepting strings of balanced ordered parenthesis. Show instantaneous description of trace for the string [{ }]*** *[2+4+4]*

Answer: Pushdown Automata is a finite automata with extra memory called stack which helps Pushdown automata to recognize Context Free Languages. A Pushdown Automata (PDA) can be formally defined as follows:

$P = (Q, \sum, \Gamma, \delta, q_0, Z_0, F)$ where,

- Q is the set of states
- $\sum$ is the set of input symbols
- Γ is the set of pushdown symbols (which can be pushed and popped from stack)
- q_0 is the initial state
- Z_0 is the initial pushdown symbol (which is initially present in stack)
- F is the set of final states

- δ is a transition function which maps Q x {Σ ∪ ∈} x Γ into Q x Γ*. In a given state, PDA will read input symbol and stack symbol (top of the stack) and move to a new state and change the symbol of stack.



A DPDA accepting strings of balanced ordered parenthesis P;

Q = (q0, q1, q2),

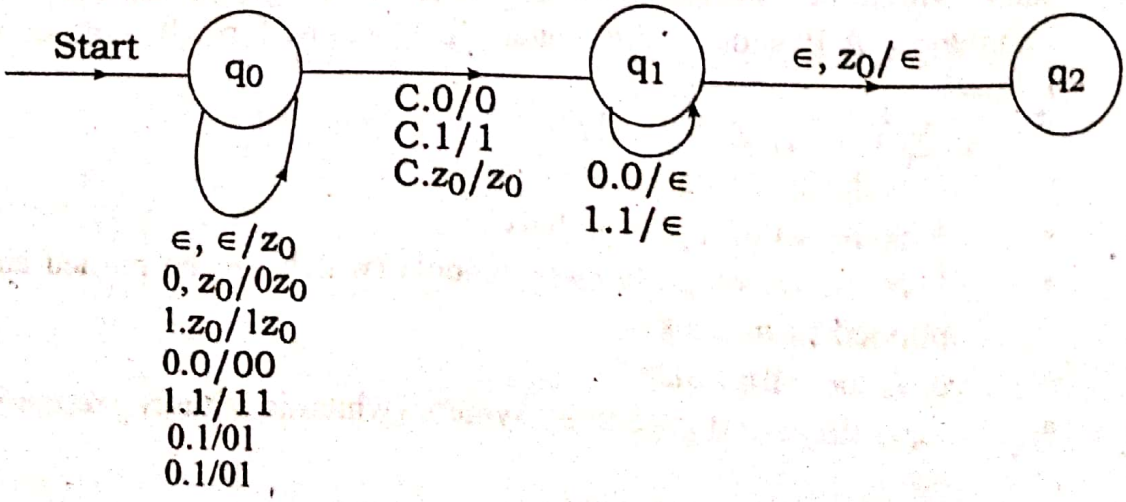
S = {{,}, (,), [,]},

Γ = S U {z0}, d, q0, z0, {q2}

Where δ is defined as:

1. δ (q0, Є, Є) = (q0, z0)
2. δ (q0, {, z0) = (q1, {z0)
3. δ (q0, [, z0) = (q1, [z0)
4. δ (q0, (, z0) = (q1, (z0)
5. δ (q1, {, {) = (q1, {{)
6. δ (q1, [, [ ) = (q1, [[ )
7. δ (q1, (, ( ) = (q1, (( )
8. δ (q1, }, {) = (q1, Є)
9. δ (q1,], [ ) = (q1, Є)
10. δ (q1,), ( ) = (q1, Є)
11. δ (q1,), {) = (q1, ({)
12. δ (q1, {, [ ) = (q1, {[ )
13. δ (q1, Є, z0) = (q2, Є)

The graphical notation for the above PDA is:



Evaluate or trace for the string [{ }]

$(q_0, [\{\}], z_0) = (q_1, \{\}], [z_0)$

$= (q_1, \}], \{[z0)$

$= (q_1,], [z0)$

$= (q_1, \epsilon, z0)$

$= (q_2, \epsilon, \epsilon)$ Accepted

3. ***Difference between TM and Other Automata (FSA and PDA). Construct a Turing Machine accepting the language, $L = \{0^n1^n \mid n >= 1\}$. Also show the transition diagram of the machine. Illustrate whether a string 0011 is accepted by the Turing Machine or not.*** *[2+6+2]*

Answer: The most significant difference between the TM and the simpler machine (FSA or PDA) is that; in a Turing Machine, processing a string is no longer restricted to a single left to right pass through input. The tape head can move in both directions and erase or modify any symbol it encounters. The machine can examine part of the input, modify it, take time execute some computation in a different area of the tape, return to re-examine the input, repeat any of these actions and perhaps stop the processing before it has locked at all input.

Turing Machine accepting the language, $L= \{0^n1^n \mid n >= 1\}$

- Given finite sequence of 0's and 1's on its tape preceded and followed by blanks.
- The TM will change 0 to an X and then a 1 to Y until all 0's and 1's are matches.
- Starting at left end of the input, it repeatedly changes a 0 to an X and moves to the right over whatever 0's and Y's it sees until comes to a 1.
- It changes 1 to a Y, and moves left, over Y's and 0's until it finds X. At that point, it looks for a 0 immediate to the right. If finds one 0 then changes it to X and repeats the process changing a matching 1 and Y.

Now, TM will have;

$M = (\{q0, q1, q2, q3, q4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q0, B, \{q4\})$

The transition rule for the move of M is described by following transition table:

	0	1	X	Y	B
q0	(q1,X,R)			(q3, Y, R)	
q1	(q1,0,R)	(q2,Y,L)		(q1, Y, R)	
q2	(q2,0,L)		(q0, X, R)	(q2, Y, L)	
q3				(q3, Y, R)	
q4					(q4, B, R)

Now, the acceptance of 0011 by the TM, M1 is described by following sequence of moves;

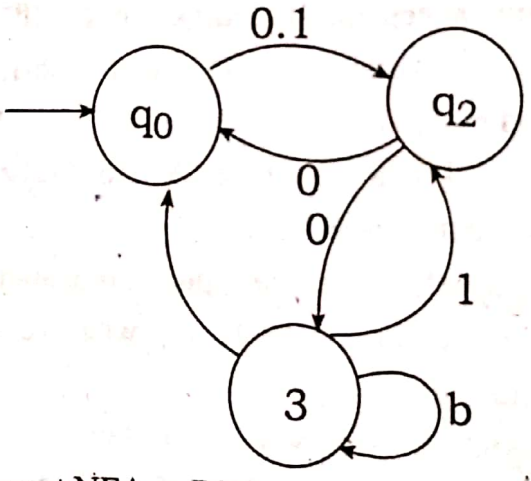
$q_0\,0011 \vdash X\,q_1011$
$\vdash X0q_111$
$\vdash Xq_20Y1$
$\vdash Xq_00Y1$
$\vdash XXq_1Y1$
$\vdash XXYq_11$
$\vdash XXq_2YY$
$\vdash Xq_2XYY$
$\vdash XXq_0YY$
$\vdash XXYq_3Y$
$\vdash XXYYq_3B$
$\vdash XXYYBq_4B$ Halt and accept.

Section B

Short Answer Questions

Attempt any eight questions. **[8×5=40]**

4. How to convert NFA to DFA? Write down the conversion steps. Also convert following NFA to their equivalent DFA; *[1+4]*



Answer: Steps to convert NFA→ DFA:

1. Construct the transaction table of given NFA machine.
2. Scan the next states column in the transaction table from initial state to final state.
3. If any of the next state consists more than one state on the single input alphabet. Then merge them and make it new state. Place this new constructed state in DFA transaction table as present state.
4. The next state of this new constructed state on input alphabet will be the summation of each next state which parts in the NFA transaction table.

5. Repeat step 2 to step 4 until all the states in NFA transaction table will be scanned completely.
6. The finial transaction table must have single next state at single input alphabet.

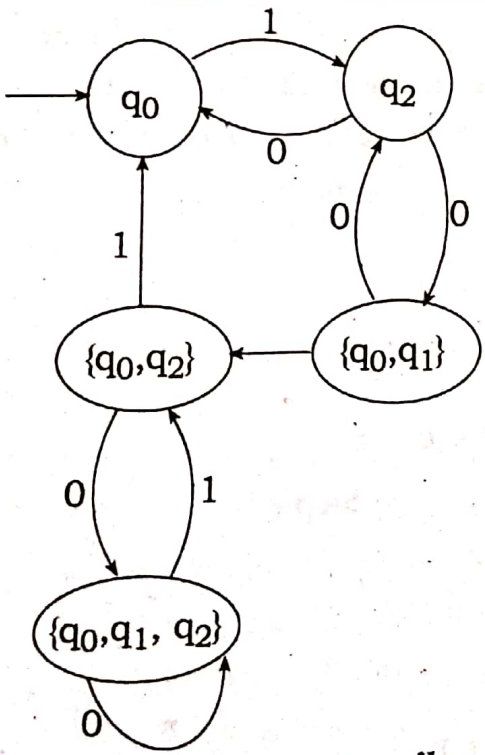
Construct the equivalent DFA of the NFA given DFA.

Step 1: Transaction Table of NFA from given DFA is;

Present State	Next State	
	0	1
→q0	{q2}	Φ
q1	Φ	{q0, q2}
q2*	{q0, q1}	{q0}

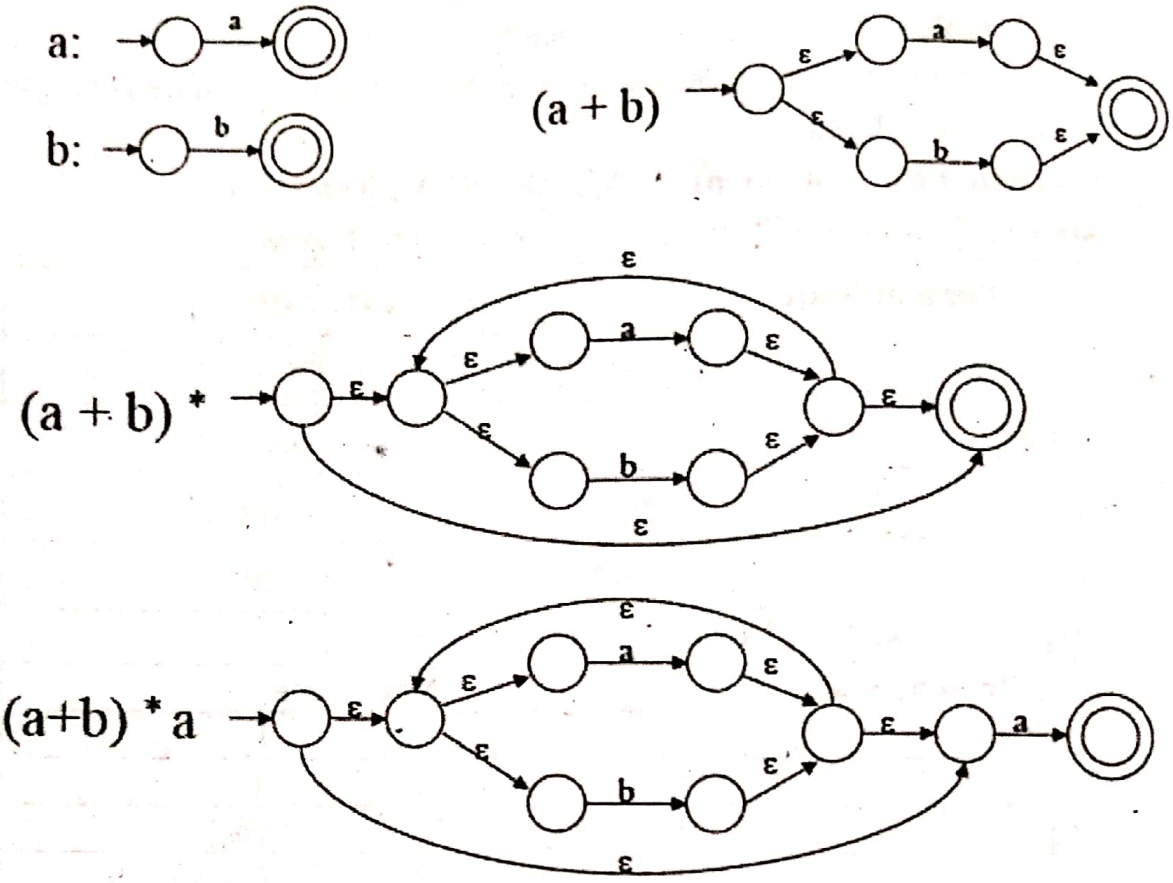
Step 2: Transaction Table of DFA:

Present State	Next State	
	0	1
→{q_0}	{q_2}	Φ
{q_2}	{q_0, q_1}	{q_0}
{q_0, q_1}	{q_2}	{q_0, q_2}
{q_0, q_2}	{q_0, q_1, q_2}	{q_0}
{q_0, q_1, q_2}	{q_0, q_1, q_2}	{q_0}



5. ***Define ε-NFA. Configure equivalent epsilon NFA for the regular expression (a + b) * a*** *[1+4]*

Answer: In NFA if a transition made without any input symbol is called ε-NFA. Here we need ε-NFA because the regular expressions are easily convertible to ε-NFA.



6. What is Parse Trees? How Ambiguity of a grammar is detected? *[3+2]*

Answer: A parse tree is a graphical depiction of a derivation. It is convenient to see how strings are derived from the start symbol. The start symbol of the derivation becomes the root of the parse tree.

Example: Let's take a grammar;

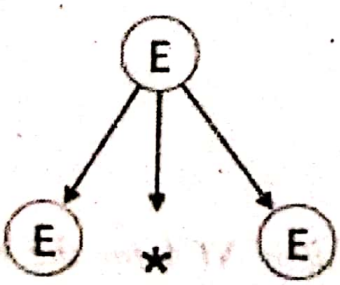
$E \rightarrow E + E$

$E \rightarrow E * E$

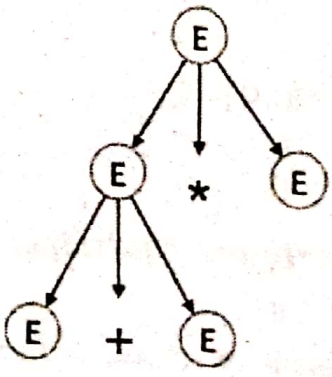
$E \rightarrow id$

Input string: id + id * id

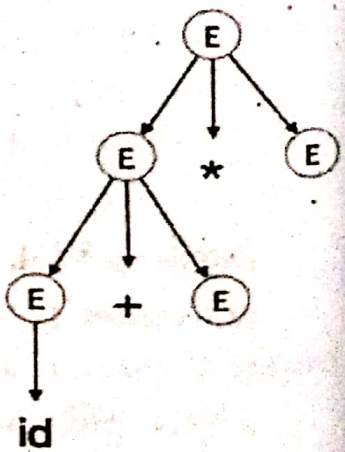
Step 1:



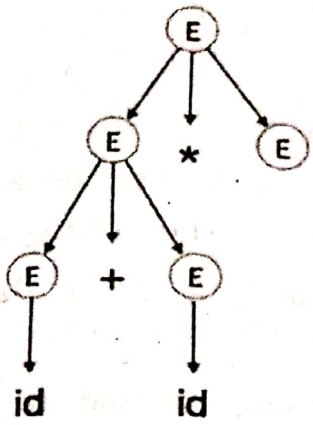
Step 2:



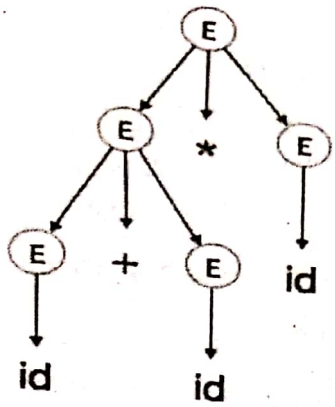
Step 3:



Step 4:



Step 5:



Ambiguity of a grammar

A grammar G is said to be ambiguous if it has more than one parse tree (left or right derivation) for at least one string. Also A grammar G is said to be ambiguous if there is a string w ∈ L(G) for which we can construct more than one parse tree rooted at start symbol of the production.

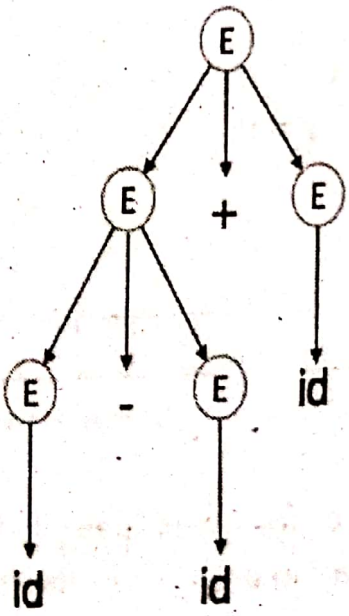
Example

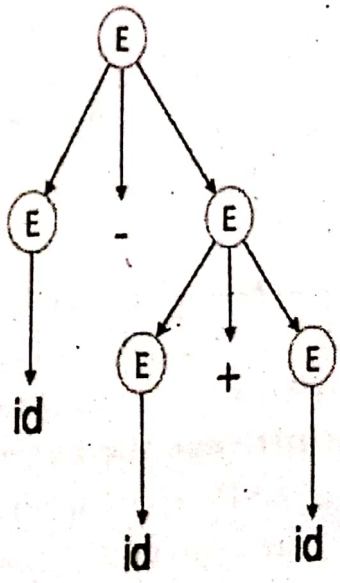
$E \rightarrow E + E$

$E \rightarrow E - E$

$E \rightarrow id$

For the string id + id – id, the above grammar generates two parse trees:





7. ***Write the regular expression over {0, 1} for strings*** *[2.5+2.5]*
   - ***all string that begins with a '1' and ends with a '0'***
   - ***string having at most two 0's***

Answer: RE for String that begins with a '1' and ends with a '0'.

1(1+0)*0

RE for String that having at most two 0's

1*(0+ ε)1*(0+ ε) 1*

8. ***What is Universal Turing Machine? Define Non-Deterministic Turing Machine.*** *[2+3]*

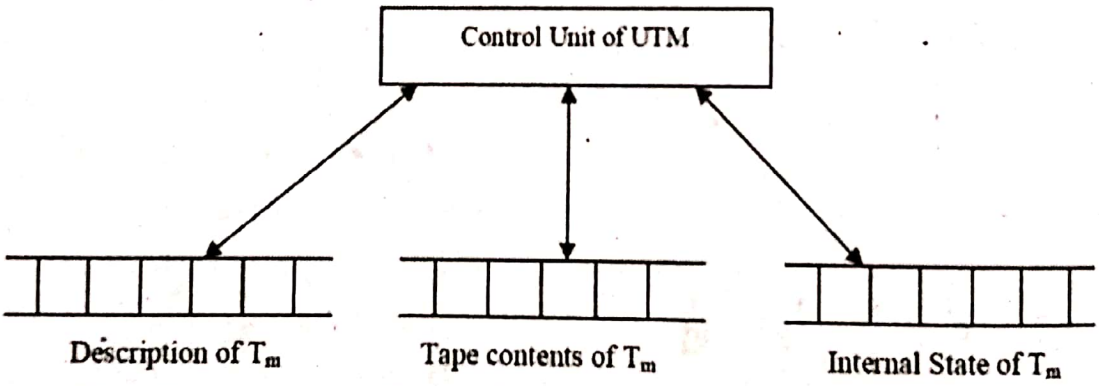
Answer: Universal Turing Machine

In computer science, a universal Turing machine (UTM) is a Turing machine that can simulate an arbitrary Turing machine on arbitrary input. The universal machine essentially achieves this by reading both the description of the machine to be simulated as well as the input thereof from its own tape.

If a TM really is a sound model of computation, it should be possible to demonstrate that it can act as a stored program machine, where the program is regarded as an input, rather than hard-wired. We shall construct a Turing Machine Mu, that takes as input a description of a Turing Machine M and an input word x, and simulates the computation of M on input x. A machine such as Mu that can simulate the behavior of an arbitrary TM is called a Universal Turing Machine. Thus, we can describe a Universal Turing Machine Tu as a TM, that on input <M, w>; where M is a TM and w is string of input alphabets, simulates the computation of M on input w.

Specially,

- Tu accepts <M, w> iff M accepts w.
- Tu rejects <M, w> iff M rejects w.



Non-Deterministic Turing Machine

In a Non-Deterministic Turing 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 Turing Machine is a tree of configurations that can be reached from the start configuration.

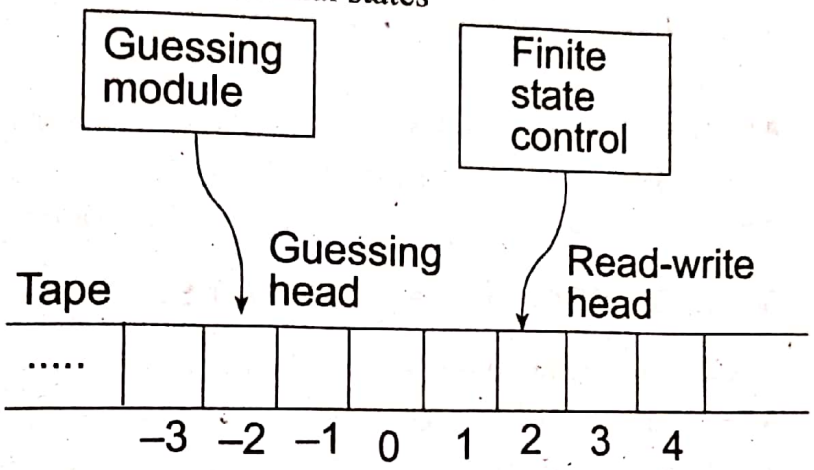
An input is accepted if there is at least one node of the tree which is an accept configuration, otherwise it is not accepted. If all branches of the computational tree halt on all inputs, the non-deterministic Turing Machine is called a Decider and if for some input, all branches are rejected, the input is also rejected.

A non-deterministic Turing machine can be formally defined as a 6-tuple (Q, X, ∑, δ, q_0, B, F) where,

- **Q** is a finite set of states
- **X** is the tape alphabet
- ∑ is the input alphabet
- **δ** is a transition function;

δ: Q × X → P(Q × X × {Left_shift, Right_shift}).

- q_0 is the initial state
- **B** is the blank symbol
- **F** is the set of final states



9. ***What is Chomsky normal form? Write down algorithm to convert CFG to CNF. Also*** **convert the following CFG into CNF.** ***[1+2+2=5]***

S → ASA | aB, A → B | S, B → b | ε

Answer: A CFG is in Chomsky Normal Form if the Productions are in the following forms:

- A → a
- A → BC
- S → ε

Where A, B, and C are non-terminals and **'a'** is terminal.

Algorithm to Convert into Chomsky Normal Form:

- Step 1 – If the start symbol S occurs on some right side, create a new start symbol S' and a new production S'→ S.
- Step 2 – Remove Null productions. (Using the Null production removal algorithm discussed earlier)
- Step 3 – Remove unit productions. (Using the Unit production removal algorithm discussed earlier)
- Step 4 – Replace each production A → B1…Bn where n > 2 with A → B1Cwhere C → B2 …Bn. Repeat this step for all productions having two or more symbols in the right side.

- Step 5 – If the right side of any production is in the form A → aB where a is a terminal and A, B are non-terminal, then the production is replaced by A → XB and X → a. Repeat this step for every production which is in the form A → aB.

Problem: Convert the following CFG into CNF

S → ASA | aB, A → B | S, B → b | ε

Solution:

(1) Since S appears in R.H.S, we add a new state S0 and S0→S is added to the production set and it becomes:

S0→S, S→ ASA | aB, A → B | S, B → b | ∈

(2) Now we will remove the null productions

B → ∈ and A → ∈

After removing B → ε, the production set becomes –

S0→S, S→ ASA | aB | a, A → B | S | ∈, B → b

After removing A → ∈, the production set becomes –

S_0→S, S→ ASA | aB | a | AS | SA | S, A → B | S, B → b

(3) Now we will remove the unit productions.

After removing S → S, the production set becomes –

S_0→S, S→ ASA | aB | a | AS | SA, A → B | S, B → b

After removing S_0→ S, the production set becomes –

S_0→ ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → B | S, B → b

After removing A→ B, the production set becomes –

S0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → S | b

B → b

After removing A→ S, the production set becomes –

S0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → b |ASA | aB | a | AS | SA, B → b

(4) Now we will find out more than two variables in the R.H.S

Here, S0→ ASA, S → ASA, A→ ASA violates two Non-terminals in R.H.S.

Hence we will apply step 4 and step 5 to get the following final production set which is in CNF:

S0→ AX | aB | a | AS | SA

S→ AX | aB | a | AS | SA

A → b |AX | aB | a | AS | SA

B → b

X → SA

(5) We have to change the productions S0→ aB, S→ aB, A→ aB
And the final production set becomes:
S0→ AX | YB | a | AS | SA
S→ AX | YB | a | AS | SA
A → b A → b |AX | YB | a | AS | SA
B → b
X → SA
Y → a

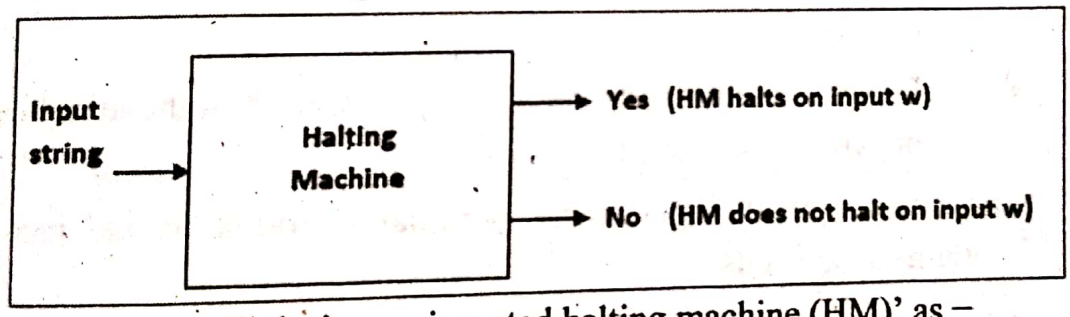
10. Describe Turing Machine Halting Problem with example. **[5]**

Answer:

Input – A Turing machine and an input string w.

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

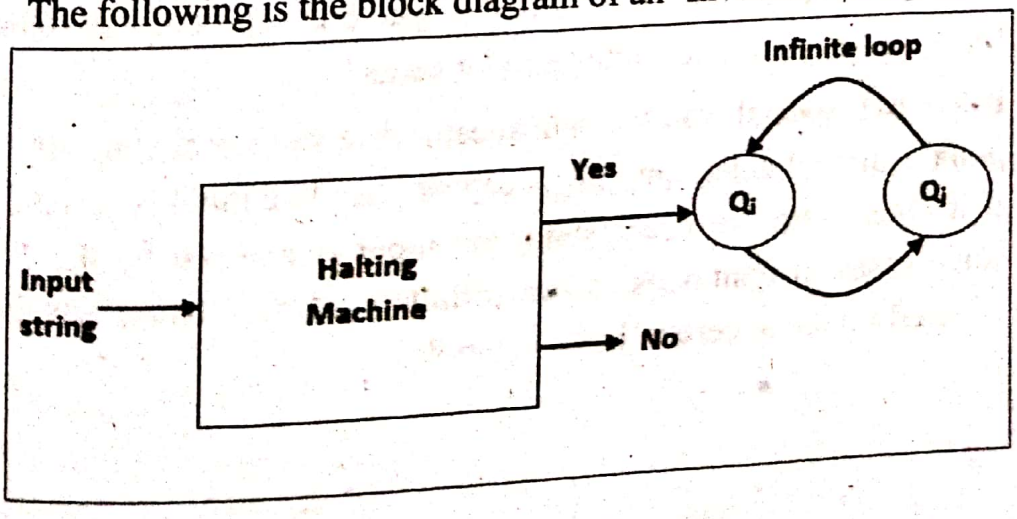
Proof – At first, we will assume that such a Turing machine exists to solve this problem and then we will show it is contradicting itself. We will call this Turing machine as a **Halting machine** that produces a 'yes' or 'no' in a finite amount of time. If the halting machine finishes in a finite amount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

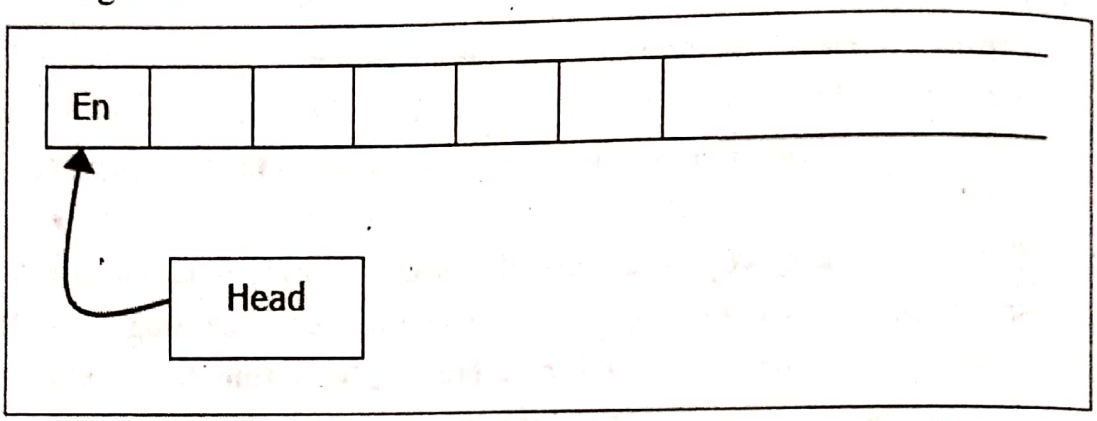
- If $(HM)_2$ halts on input, loop forever.
- Else, halt.

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

11. *D*

efine Semi-Infinite Tape Turing Machine. *[5]*

Answer: 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 q0 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.

12. *Write down the languages (Regular Set) for following regular expressions.* [5]

Regular Expressions

- *(0 + 10*)*
- *(0*10*)*
- *(0 + ε)(1 + ε)*
- *(a+b)**
- *(a+b)*abb*
- *(11)**
- *(aa)*(bb)*b*
- *(aa + ab + ba + bb)**

Answer:

Regular Expressions	Regular Set
(0 + 10*)	L = { 0, 1, 10, 100, 1000, 10000, ... }
(0*10*)	L = {1, 01, 10, 010, 0010, ...}
(0 + ε)(1 + ε)	L = {ε, 0, 1, 01}
(a+b)*	Set of strings of a's and b's of any length including the null string. So L = { ε, a, b, aa , ab , bb , ba, aaa.....}
(a+b)*abb	Set of strings of a's and b's ending with the string abb. So L = {abb, aabb, babb, aaabb, ababb...}
(11)*	Set consisting of even number of 1's including empty string, So L= {ε, 11, 1111, 111111, ...}
(aa)*(bb)*b	Set of strings consisting of even number of a's followed by odd number of b's , so L = {b, aab, aabbb, aabbbbb, aaaab, aaaabbb, ...}
(aa + ab + ba + bb)*	String of a's and b's of even length can be obtained by concatenating any combination of the strings aa, ab, ba and bb including null, so L = {aa, ab, ba, bb, aaab, aaba, ...}

Model Questions Sets For Practice

MODEL SET 1

Course Title: Theory of Computation
Course No: CSC257
Level: B. Sc CSIT Second Year/ Fourth Semester

Full Marks: 60
Pass Marks: 24
Time: 3 Hrs

Section A

Long Answer Questions

Attempt any two questions. **[2*10=20]**

1. Define the extended transition function of DFA. Construct a DFA equivalent to the NFA. M= ({p, q, r},{0, 1}, δ, p,{q, s}) Where δ is defined in the [2+8=10]

	0	1
P	{q, s}	{q}
Q	{r}	{q, r}
R	{s}	{p}
S	-	{p}

2. Explain the construction of NFA with ε-transition from any given regular expression.
   Let A = (Q, ∑, δ, q_0, qf) be a DFA and suppose that for all a in ∑we have δ(q_0, a)= δ(qf ,a). Show that if x is a non empty string in L(A), then for all k > 0,xk is also in L(A).
3. **Explain in detail notes on Turing Machine with example.** Construct a Turing Machine accepting Palindrome. Also show the transition diagram of the machine. Illustrate whether a string bbbaabbb is accepted by the Turing Machine or not. [2+6+2]

Section B

Short Answer Questions

Attempt any eight questions. **[8*5=40]**

4. Simplify the following grammar
   S→aAa | bBb | BB
   A→C
   B→S | A
   C→S | ϵ
5. What are the closure properties of CFL? State the proof for any two properties. [1+4]

6. What is additional feature PDA has when compared with NFA? Is PDA superior over NFA in the sense of language acceptance? Justify your answer.
7. Define instantaneous description of a PDA.
8. What is un-decidable problem? Define Post's Correspondence Problem with an example. [1+4]
9. How pumping lemma can be used to prove that any language is not a regular language? Show that language, $L = \{0^r 1^r | n \geq 0\}$ is not a regular language. [4+1]
10. Discuss how Turing Machine with multiple tracks differs from a Turing Machine with multiple tapes. [5]
11. How context free grammars are defined? Write a context free grammar over {0,1}, where the strings start and end with the same symbol. [2+3]
12. What is halting problem? How can you argue that halting problem is un-decidable? [1+4]

MODEL SET 2

Section A

Long Answer Questions

Attempt any two questions. **[2*10=20]**

1. Is it true that the language accepted by any NFA is different from the regular language? Justify your Answer. Consider the following ε–NFA. Compute the ε–closure of each state and find it's equivalent DFA. (2+1+7)

	E	A	b	C
P	{q}	{p}	Φ	Φ
Q	{r}	Φ	{q}	Φ
R	Φ	Φ	Φ	{r}

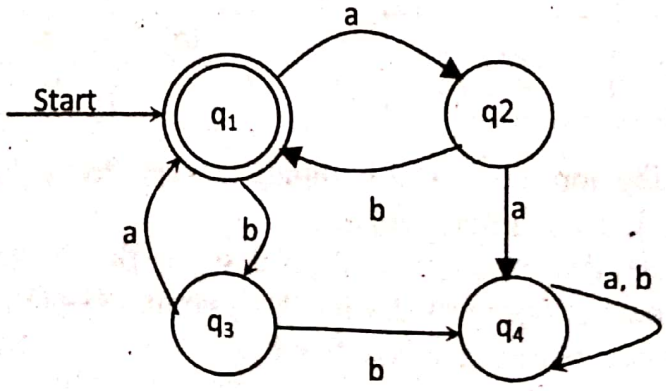
2. Describe the applications of Pumping lemma. Prove that $L = \{a^n b^n$ for n=0, 1, 2, 3.....} is not regular.
3. Define push down automata. Also describe the applications of push down automata. Design PDA for the grammar G= (V, T, P, S) where V={S}, T={a, b, c} and P is defined as,
   S→a S a
   S→ b S b
   S→ c

Section B

Short Answer Questions

Attempt any eight questions. [8*5=40]

4. Consider the following grammar
   S→AB
   A→a
   B→C/b
   C→D
   D→E
   E→a
5. Define DFA? How it is differ from NFA? Design a finite automata which accepts the language L={w/w has both an even number of 0's and an even number of 1's over alphabet ∑={0, 1}.
6. Describe universal Turing machine with suitable example.
7. Design a Turing machine that accepts the language of all strings which contain aba as a substring.
8. Define class P, NP and NP complete problems with suitable practical examples.
9. Define Chomsky normal form with suitable example. How to convert CFG to CNF? Change the following grammar into CNF
   S→1A/0B
   A→1AA/0S/0
   B→0BB/1
10. Let G be CFG
    S→bB/aA
    A→b/bS/aAA
    B→a/aS/bBB
    For the string bbaababa find
    - Left most derivation
    - Right most derivation and
    - Parse tree
11. Define Ardens's theorem. Use Ardens's theorem find the regular expression of following transition diagram,



12. What is left recursion grammar? How left recursion eliminate from given grammar? Explain with suitable example.

MODEL SET 3

Section A

Long Answer Questions

Attempt any two questions. **[2*10=20]**

1. Let input alphabet be $\sum = \{a, b, c\}$ and L be the language of all words in which all the a's come before the b's and there are same number of a's as b's and arbitrarily many c's that can be in front, behind or among the a's and b's. Some words in L are abc, caabcb, ccacaabcccbccbc.
   - Show the language is not regular
   - Design a PDA for above given language.
2. Define formal definition of Turing machine. How Turing machine differ from Push down automata? Design a Turing machine that recognizes the language of all strings of even length over alphabet $\sum = \{a, b\}$.
3. Define CFG. Design CFG for $\sum = \{a, b\}$ that generates the set of
   a. All strings with exactly one a.
   b. All strings with at least one a
   c. All strings with at least 3 a's

Section B

Short Answer Questions

Attempt any eight questions. **[8*5=40]**

4. What is regular expression? How it is differ from CFG? Write the regular expression over alphabet $\sum = \{a, b, c\}$ containing at least one a and at least one b.
5. Define ε-NFA. How it is differ from NFA? Construct the ε-NFA for the regular expression $(0 + 1)^* 1 (0 + 1)$.
6. Define Bacos Naur Form (BNF). Write CFG for the regular expression,
   $r = 0^* 1 (0 + 1)^*$
7. What is the use of PDA? Construct a push down automata to accept the following language,
   $L = \{w\, w^R : w \in \{a, b\}^*\}$
8. Describe properties of Turing machine. Design a Turing machine that accepts the language of all strings which contain aba as a substring.
9. What is the advantage of using multi tapes Turing machine over simple Turing machine? Describe universal Turing machine with suitable example.
10. What is context sensitive grammar? How it is differ from context free grammar? Explain their differences with suitable example.
11. Define class P problem with suitable example. How class P problems differ from class NP problems? Explain.
12. Construct the transaction system for the following regular expressions;
    a. $r = 1 + 00 + 010^*1$
    b. $r = (ab + c^*)^*b$

MODEL SET 4

Section A

Long Answer Questions

Attempt any two questions. **[2*10=20]**

1. Define Pumping lemma for CFG. Prove that language $L = \{ww/w \in \{a, b\}^*\}$ is not context free.
2. Define formal definition of Turing machine. How Turing machine differ from Finite automata? Design a Turing machine that recognizes the language of all strings of odd length over alphabet $\sum = \{a, b\}$.
3. Define CFG. How it is differ from regular expression? Design regular expression for $\sum = \{a, b\}$ that generates the set of
   a. All strings with exactly one a.
   b. All strings with at least one a
   c. All strings with at least 3 a's

Section B

Short Answer Questions

Attempt any eight questions. **[8*5=40]**

4. Define Kleen closure operation. How it is differ from positive closure? Find Kleen closure of following expressions over $\sum = \{a, b\}$
   a. a^*
   b. $(ab)^*$
   c. $(a + b)^*$
   d. $ab + a^*$
   e. $(ba + ab^*)^*$
5. Define NFA. How it is differ from ε-NFA? Construct NFA over {a, b} that accepts either empty strings or ab, or aab, or aba. Use extended transaction function to determine whether the NFA accepts bbbabb and aab.
6. Construct a Turing machine accepting the language, $L = \{a^n b^{2n} | n>=1\}$. Also draw the equivalent transaction diagram.
7. Define CFG. Write CFG for the regular expression, $r = (0+1).(0 + 1)^*$
8. What are the applications of PDA? Construct a push down automata for balanced parentheses, i.e. $\sum = \{, [, (,),], \}$.
9. What is left recursive grammar? How removal of left recursion from grammar is occurring? Remove left recursion from following grammar;

   $S \rightarrow Sa \mid SBb \mid SAab \mid a \mid b \mid c$

   $A \rightarrow Aab \mid Ac \mid a \mid bb \mid c$

   $B \rightarrow abc \mid a \mid b \mid \varepsilon$

10. Define regular expression. Define regular expression for following statements:
    a. An IPV4 Address
    b. A MAC Address
    c. Valid identifier defined in programming like C
11. What does NP completeness means? Show that SAT is NP-complete.
12. Simplify following grammar and convert to the CNF
    S→ abSb | aAb | a
    A→ bS | aAAb

MODEL SET 5

Section A

Long Answer Questions

Attempt any two questions. **[2*10=20]**

1. Is context free grammar exploring the meaning of the term "Context free" with a suitable example? Write the CFG for the language L = {x 0^n y 1^n z | n>=0}, and give the parse tree for the sting x000y11z.
2. Define Turing machine. Design a Turing machine for a language L= {a, b, c}|n > = 0}. Show instantaneous description for aabbcc and abcc.
3. Define transaction diagram and transaction table of DFA. Design a FA over alphabet $\sum$ = {0, 1}, which accepts the set of strings either start with 01 or end with 01.

Section B

Short Answer Questions

Attempt any eight questions. **[8*5=40]**

4. Define 'terms', 'language' and 'alphabet'. Find all pairs of set of A and B for which A.B = { ab, aaa, abab, abbb, abaaa, ababbb}.
5. Let us define the alphabet set as $\sum$ = {0, 1}, then
    a. Build a DFA that accepts all the strings ending in 00.
    b. Build a DFA that accepts the strings that do not have 1110 as their prefix.
6. Show that if M = (Q, $\sum$, δ, q_0, F) and N' = (Q, $\sum$, δ, q_0, Q-F) are two DFA's, then (L(M))'=L(M')
7. How to convert given regular expression to equivalent ε-NFA? Convert following regular expression to their equivalent ε-NFA,
    a. (a + b)*
    b. b

8. Write down the applications of pumping lemma. Prove that $L = \{a^n b a^n \text{ for } n = 0, 1, 2, 3...\}$ is not regular.
9. What is the use of left most derivation and right most derivation? Let G be CFG with production rules,
   $S \rightarrow bB | aA$
   $A \rightarrow b | bS | aAA$
   $B \rightarrow a | aS | bBB$
   For the string bbaababa find
   a. Left most derivation
   b. Right most derivation and
   c. Parse tree
10. Define notations used in Turing machine. Design a Turing machine that recognizes the set of all strings of 0's and 1's containing at least one 1.
11. Configure a push down automata accepting the language, $L = \{w C w | w \in (a + b)^* \}$. Show instantaneous description of string abbCbba and baCba.
12. What do you mean by decision problem? Give Turing's proof of Halting problem as undecided problem.

MODEL SET 6

Section A

Long Answer Questions

Attempt any two questions. **[2*10=20]**

1. Prove that there exists a DFA for every ε-NFA. Prove that the complement of a regular language is also regular.
2. What are the required fields of an instantaneous description of a Turing machine? Design a Turing machine with no more than three states that accepts the language a(a+b)*.
3. Convert the following grammar into CNF
   a. $S \rightarrow cBA, S \rightarrow A, A \rightarrow cB, A \rightarrow AbbS, B \rightarrow aaa$
   b. $S \rightarrow a|AAB, A \rightarrow ab|aB| \varepsilon, B \rightarrow aba| \varepsilon$
   c. $S \rightarrow aAD, A \rightarrow aB|bAB, B \rightarrow b, D \rightarrow d$

Section B

Short Answer Questions

Attempt any eight questions. **[8*5=40]**

4. Define ε-closure (q) with an example. Construct a DFA for the language over {0, 1}* such that it contains "000" as a substring.
5. State the relations among regular expression, deterministic finite automata, non deterministic finite automaton and finite automaton with epsilon transition. Draw a Non-deterministic finite automaton to accept strings containing the substring 0101.

6. Let us define the alphabet set as $\sum = \{0, 1\}$, then
   a. Build a DFA that accepts all the strings ending in 00 and start with 11.
   b. Build a DFA that accepts the strings that do not have 1110 as their prefix.
   c. Build a NFA that contains at least one 0's and one 1's.
7. What are the applications of Turing Machine? Construct a Turing Machine for language $L = \{ww^r \mid w \in \{0, 1\}\}$
8. Prove that if there exists a PDA that accepts by final state then there exists an equivalent PDA that accepts by Null state.
9. Find the context free languages for the following grammars.
   a. S→aSbS|bSaS| ε
   b. S→aSb|ab
   c. S→aSb|aAb, A→bAa, A→ba
10. What is ambiguous grammar? Let G be CFG with production rules,
    S→a|^|(T)
    T→T,S|S
    Find left most derivation, rightmost derivation and parse tree for (((a, a),^(a)),a)
11. What are the conventional notations of Push down Automata? Convert the following CFG to a PDA,
    S→aAA, A→aS|bS|a
12. State the two normal forms and give an example. Convert the following grammar G in greibach normal form.
    S→ABb|a
    A→aaA|B
    B→bAb

MODEL SET 7

Section A

Long Answer Questions **[2*10=20]**

Attempt any two questions.

1. List the primary objectives of Turing Machine. Is the language $L = \{a^n b^n c^n \mid n>=1\}$ is context free? Justify.
2. List the closure properties of Context Free Languages. Let G be the grammar S→aB|bA
   A→a|aS|bAA
   B→b|bS|aBB.
   For the string aaabbabbba, Find
   a. LMD (Left most derivation)
   b. RMD (Right most derivation)

3. Specify the use of context free grammar. Convert the following grammar into an equivalent one with no unit productions and no useless symbols

S→ABA
A→aAA|aBC|bB
B→ A|bB|Cb
C→CC|Cc

Short Answer Questions

Attempt any eight questions. **[8*5=40]**

4. When we say a problem is decidable? Give an example of un-decidable problem.
5. Is it true that the language accepted by a non deterministic Turing Machine is different from recursively enumerable language?
6. Prove that for two recursive languages L1 and L2 their union and intersection is recursive.
7. Discuss the various techniques for Turing Machine Construction. Write about Multi tape Turing Machines.
8. State and prove the pumping lemma for CFL. What is its main application? Give two examples.
9. Is NPDA (Nondeterministic PDA) and DPDA (Deterministic PDA) equivalent? Illustrate with an example.
10. Is the following grammar is ambiguous? Justify your answer.
    a. E→ E+E |E*E | id
    b. E→ E+E|E*E|(E)|a
11. Construct the PDA accepting the language
    a. $L=\{(ab)^n \mid n>=1\}$ by empty stack.
    b. $L=\{ww^R \mid w \text{ is in } (a+b)^*\}$
12. What are the applications of CFG? Write down the context free grammar for the language $L = \{ a^n b^n \mid n>=1\}$

MODEL SET 8

Section A

Long Answer Questions

Attempt any two questions. **[2*10=20]**

1. Define parse tree with an example. Construct a CFG over {a,b} generating a language consisting of equal number of a's and b's. Is the grammar below ambiguous S→ SS | (S) | S (S) S | E?
2. What are the different types of language accepted by a PDA and define them? When is Push down automata (PDA) said to be deterministic? Construct PDA for the language $L = \{a^n b^m \mid n>m>=0\}$. Also show that acceptance of string aaaabbb.

3. Prove any two closure properties of regular languages. Construct a minimized DFA from the regular expression
   a. (b/a)*baa
   b. 0*(01)(0/111)*

Section B

Short Answer Questions

Attempt any eight questions. **[8*5=40]**

4. Which of the following languages is regular? Justify
   a. $L= \{a^n b^m \mid n, m>=1\}$
   b. $L= \{a^n b^n \mid n>=1\}$
   c. $L= \{0^{n2} \mid n \text{ is an integer, } n>=1\}$
5. What is un-ambiguity? Construct the Context free grammar representing the set of palindromes over (0+1)*
6. What do you mean by null production and unit production? Give an example. Consider the following grammar G with productions
   S→ ABC | BaB
   A→aA|BaC|aaa
   B→bBb|a
   C→CA|AC
   Give a CFG with no useless variables that generates the same language.
7. Prove or disprove that the regular languages are closed under concatenation and complement.
8. Compare NFA and PDA. Is NPDA (Nondeterministic PDA) and DPDA (Deterministic PDA) equivalent? Illustrate with an example.
9. What is chomsky normal form? Convert the following grammar into CNF
   S→aAD
   A→aB|bAB
   B→ b
   D→ d
10. Mention the difference between decidable and un-decidable problems. How to prove that the Post Correspondence problem is Un-decidable?
11. Prove that there exists an NFA with ε-transitions that accepts the regular expression γ.
12. Differentiate regular expression and regular language. Give regular expressions for the following
    a. L_1 = set of all strings of 0 and 1 ending in 00 and starting with 1
    b. L_2 =set of all strings of 0 and 1 beginning with 0 and ending with 1 and contain even length of 0 and 1.

❑❑❑