INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

INFORMATION TECHNOLOGY

DEFINITION AND TERMINOLGY

Course Title	**THEORY OF COMPUTATION**				
Course Code	AITC04				
Program	B.Tech				
Semester	IV	IT			
Course Type	Core				
Regulation	IARE - R20				
Course Structure	Theory			Practical	
	Lecture	Tutorials	Credits	Laboratory	Credits
	3	1	4	-	-
Course Coordinator	Dr. U Sivaji, Assistant Professor				

COURSE OBJECTIVES:

The students will try to learn:

I	The fundamental knowledge of automata theory which is used to solve computational problems.
II	TThe reorganization of context free language for processing infinite information using push down automata..
III	The computer based algorithms with the help of an abstract machine to solve recursively Enumerable problems.

COURSE OUTCOMES:

After successful completion of the course, students should be able to:

CO 1	**Make use of** deterministic finite automata and non deterministic finite automata for modeling lexical analysis and text editors.	Apply
CO 2	**Extend** regular expressions and regular grammars for parsing and designing programming languages.	Understand
CO 3	**Illusrate** the pumping lemma on regular and context free languages for perform negative test .	Understand
CO 4	**Demonstarte** context free grammars, normal forms for generating patterns of strings and minimize the ambiguity in parsing the given strings.	Understand

CO 5	**Construct** push down automata for context free languages for developing parsing phase of a compiler.	Apply
CO 6	**Apply** Turing machines and Linear bounded automata for recognizing the languages,complex problems.	Apply

DEFINITION AND TERMINOLOGY:

S.No	DEFINITION	CO's
	MODULE I	
	FINITE AUTOMATA	
1	**Define Automata?** It is an abstract model of digital computer. It has a mechanism for reading input. Automata enable the scientists to understand how machines compute the functions and solve problems.	CO 1
2	**Define Symbol?** Symbol is the smallest building block, which can be any alphabet, letter or any picture. Ex: a, b, 0,1..	CO 1
3	**Define Alphabet with example** Alphabet is set of symbols, which are always finite. It is denoted by $\sum$. Ex: {0, 1}, {a, b}, {a, b, c}.	CO 1
4	**What is String?Mention an example** String is a finite sequence of symbols from 1 error3 warnings some alphabet. String is generally denoted as w. Ex: String over $\sum$ = {a,b} is ababa	CO 1
5	**What is length of string?** The number of symbols in the given string is called length of the string. length of a string is denoted as $\mid w \mid$. Example: $\mid abb \mid= 3$ over $\sum$ = {a,b}.	CO 1
6	**What is substring of a string?** Any sequence of 0 or more consecutive symbols of the given string over an alphabets is called substring.Example: For string abb the substrings are ϵ,a, ab, abb.	CO 1
7	**Define Null String?** Empty string is the string with zero occurrences of symbol, represented as $\in$.	CO 1
8	**What is Prefix of a String?** A group of leading symbols of string is called as prefix. Ex: Prefixes of sting abc are ϵ,a, ab, abc .	CO 1
9	**What is Suffix of a String?** A group of trailing symbols of string is called as suffix. Ex: Suffixes of sting abc are ϵ, a, bc, abc	CO 1

10	**What is proper Prefix of a String?**	CO 1
	A group of leading symbols other than string is called as proper prefix. Ex: Proper Prefixes of sting abc are ϵ, a, ab	
11	**What is proper Suffix of a String?**	CO 1
	A group of trailing symbols other than the string is called as proper suffix. Ex: Proper Suffixes of sting abc are ϵ, a, bc.	
12	**What is power of alphabet?**	CO 1
	Consider $\sum$ = {0,1} the following are the powers. 1.$\sum^0 = \{\epsilon\}$ 2.$\sum^1 = \{0, 1\}$ 3.$\sum^2 = \{00, 01, 10, 11\}$ 4.$\sum^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$	
13	**What is kleenclosure?**	CO 1
	The set of all strings over on alphabet. It is denoted as $\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \sum^3 \cup \ldots$	
14	**What is positive closure?**	CO 1
	The set of all strings except ϵ over on alphabet. It is denoted as $\sum^+ = \sum^1 \cup \sum^2 \cup \sum^3 \cup \ldots$	
15	**Define Language?**	CO 1
	A language is a set of strings, chosen form some $\sum^*$ or A language is a subset of $\sum^*$. A language which can be formed over $\sum$ can be Finite or Infinite. Language that contains strings over $\sum = \{a, b\} are \{\epsilon, a, b, aa, ab, \ldots.\}$	
16	**What is grammar?**	CO 1
	It contains set of rules to generate the strings of a language.	
17	**Define formal languages?**	CO 1
	Formal language is set of all strings where each string restricted over particular forms of strings. String with given input.	
18	**What are the types of formal languages?**	CO 1
	There are four types 1.Regular languages 2.Context Free languages 3.Context Sensitive languages 4.Recursively Enumerable languages	
19	**What are the types of automation?**	CO 1
	There are four types: 1.Finite Automation 2.Push Down Automation 3.Linear Bound Automation 4.Turing Machine	
20	**What is finite automata?**	CO 1
	An automation the accepts regular language is called finite automata.	

21	**What are the types of finite automata?**	CO 1
	There are the two types: 1.Deterministic Finite Automata (DFA) 2.Non- Deterministic Finite Automata (NDFA) Both are equivalent in power.	
22	**What is Push Down automata?**	CO 1
	An automata the accepts context free language is called Push Down automata.	
23	**What are the types of Push Down automata?**	CO 5
	There are the two types: 1.Deterministic Push Down Automata (DPDA) 2.Non- Deterministic Push Down Automata (NPDA)	
24	**What is Linear Bound automata?**	CO 5
	An automata the accepts context sensitive language is called Linear Bound automata.	
25	**What are the types of Linear Bound automata?**	CO 5
	There are the two types: 1.Deterministic Linear Bound Automata (DLBA) 2.Non- Deterministic Linear Bound Automata (NLBA)	
26	**What is Moore Machine?**	CO 5
	A Moore Machine's output depends only on the current state. It does not depend on the current input.	
. 27	**Define Mealy Machine?**	CO 6
	A Mealy Machine changes its output on the basis of its present state and current input.	
28	**Define Finite Automata?**	CO 6
	A finite automata consists of finite set of states and transitions from state to state occur on input symbols chosen from an alphabet.	
29	**What is Initial state?**	CO 1
	It is the starting state from which reading of input symbols from the string starts.	
30	**What is Final state?**	CO 1
	It is the last state reached by a accepted state means accepted string should reach final state.	
MODULE II		
REGULAR LANGUAGES		
1	**What is a regular expression?**	CO 2
	A regular expression is a string that describes the whole set of strings according to certain syntax rules.	

2	**Define acceptor?** An automaton that computes a Boolean function is called an acceptor. All the states of an acceptor is either accepting or rejecting the inputs given to it.	CO 2
3	**What is a regular language?** A language accepted by the regular expression is called as Regular Language.	CO 2
4	**What is a regular set?** A regular language which is accepted by finite automata is called regular set.	CO 2
5	**What is Kleen closure?** Kleen closure is defined as * that is 0 or more occurrence.	CO 2
6	**What is positive closure ?** positive closure is defined as + that is 1 or more occurrence.	CO 2
7	**Identify the equivalent identity rule for RR*.** The equivalent identity rule for RR* is R*R	CO 2
8	**Identify the equivalent identity rule for (R*)*** The equivalent identity rule for (R*)* is R*	CO 5
9	**Identify the equivalent identity rule for R*R*** The equivalent identity rule for R*R*is R*.	CO 5
10	**Identify the equivalent identity rule for R+R** The equivalent identity rule for R+R is R(Idempotent law)	CO 5
11	**Identify the equivalent identity rule for Rϵ or ϵR** The equivalent identity rule for Rϵ or ϵR is R	CO 5
12	**Identify the equivalent identity rule forϕ L or L ϕ** The equivalent identity rule for ϕ L or L ϕ is ϕ	CO 4
13	**Identify the equivalent identity rule for ϵ +RR*** The equivalent identity rule for ϵ+RR* is R*	CO 5
14	**Identify the equivalent identity rule for (PQ)*P** The equivalent identity rule for (PQ)*P is P(QP)*	CO 4
15	**What are the applications of pumping lemma?** Pumping lemma is used to check if a language is regular or not. (i) Assume that the language(L) is regular. (ii) Select a constant 'n'. (iii) Select a string(z) in L, such that $\mid z \mid > n$. (iv) $Splitthewordzintou, vandwsuchthat \mid uv \mid \leq n$ and $\mid v \mid \geq 1$. (v) $Youachieveacontradictiontopumpinglemmathatthereexistsan`i'$	CO 5
16	**What is Arden's Theorem?** Arden's theorem helps in checking the equivalence of two regular expressions. The regular expression R is given as : R=Q+RP Which has a unique solution as R=QP*.	CO 6

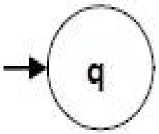
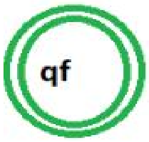
17	**What is the closure property of regular sets?**	CO 5
	The regular sets are closed under union, concatenation and Kleene closure. r1Ur2= r1 +r2 r1.r2= r1r2 (r)*=r* The class of regular sets are closed under complementation, substitution, homomorphism and inverse homomorphism.	
18	**Identify the RE for string length exactly 2?**	CO 4
	The regular expression for strings of length exactly 2 is (a+b)(a+b)	
19	**Identify the RE for string length exactly 3**	CO 4
	The regular expression for strings of length exactly 3 is (a+b)(a+b)(a+b)	
20	**Identify the RE for string length atleast 2**	CO 5
	The regular expression for strings of length exactly 2 is (a+b)(a+b)(a+b)	
21	**Identify the RE for string length atmost 2**	CO 2
	The regular expression for strings of length exactly 2 is $(a+b+\epsilon)(a+b+\epsilon)$	
22	**Identify the regular set for given regular expression (0 + 10*)**	CO 2
	Regular set is L ={ 0, 1, 10, 100, 1000, 10000,..}	
23	**Identify the regular set for given regular expression (0*10*)**	CO 2
	Regular set is L = {1, 01, 10, 010, 0010, . . . }	
24	**Identify the regular set for given regular expression $(0+\epsilon)(1+\epsilon)$**	CO 2
	Regular set is $(0 + \epsilon)(1 + \epsilon)$	
25	**Identify the regular set for given regular expression (11)***	CO 2
	Set consisting of even number of 1's including empty string, So L= { ϵ, 11, 1111, 111111, ...}	
26	**Identify the regular set for given regular expression (aa)*(bb)*b**	CO 2
	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, ...}	
27	**Identify the regular set for given regular expression (aa + ab + ba + bb)***	CO 2
	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, ...}	

28	**Identify the regular set for given regular expression (a+b)*abb**	CO 4
	Set of strings of a's and b's ending with the string abb. So L = {abb, aabb, babb, aaabb, ababb, ...}	
29	**Identify the RE for strings which begin or end with either 00 or 11.**	CO 5
	The regular expression for begins and ends with 00 or 11 is=[(00+11)(0+1)*] + [(0+1)* (00+11)]	
30	**Identify the RE for strings with atleast two c's over the set $\sum$={c,b}**	CO 4
	The regular expression for strings with atleast two c's: (b+c)* c (b+c)* c (b+c)*	
	MODULE III	
	CONTEXT FREE GRAMMARS GENERATION	
1	**What are the applications of Context free languages?**	CO 6
	Context free languages are used in: a)Defining programming languages. b)Formalizing the notion of parsing. c)Translation of programming languages. d)String processing applications.	
2	**Define a context free grammar**	CO 4
	A context free grammar (CFG) is denoted as G=(V,T,P,S) where V = Finite non-empty set of variables / non-terminal symbols T = Finite set of terminal symbols P = Finite non-empty set of production rules of the form A → α where A∈ V and $\alpha \in (V \cup T)^\star$ S = Start symbol	
3	**Define left-most Derivation?**	CO 6
	In the left most derivation, the input is scanned and replaced with the production rule from left to right. So in left most derivatives read the input string from left to right.	
4	**What is right-most Derivation?**	CO 6
	In the right most derivation, the input is scanned and replaced with the production rule from right to left. So in right most derivatives read the input string from right to left.	
5	**What is CFL?**	CO 6
	L is a context free language (CFL) if it is L(G) for some CFG G.	
6	**What is Derivation?**	CO 6
	Synthesized attributes and inherited attributes.	
7	**Define parse tree?**	CO 6
	Parse tree is the graphical representation of symbol. The symbol can be terminal or non-terminal.In parsing, the string is derived using the start symbol. The root of the parse tree is that start symbol.	

8	**Define subtree?**	CO 4
	A subtree of a derivation tree is a particular vertex of the tree together with all its descendants ,the edges connecting them and their labels.The label of the root may not be the start symbol of the grammar.	
9	**If $S \to aSb \mid aAb$, $A \to bAa$, $A \to ba$ .Find out the CFL**	CO 6
	$S \to aAb \implies abab$ $S \to aSb \implies$ a aAb b $\implies$ a a ba b b(sub $S \to aAb$) $S \to aSb \implies$ a aSb b $\implies$ a a aAb b b $\implies$ a a a ba b bb Thus L={anbmambn, where n,m $\geq$ 1}	
10	**Define ambiguous grammar?**	CO 4
	A grammar is said to be ambiguous if it has more than one derivation trees for a sentence or in other words if it has more than one leftmost derivation or more than one rightmost derivation	
11	**Write the three ways to simplify a context free grammar?**	CO 4
	a)By removing the useless symbols from the set of productions. b)By eliminating the empty productions. c)By eliminating the unit productions.	
12	**List the Intermediate forms of source programsWrite about normal form?**	CO 4
	By reducing the grammar, although the grammar gets minimized but does not get standardized. This is because the RHS of productions have no specific format. In order to standardize the given grammar, we normalize it.	
13	**Differentiate sentences and sentential forms**	CO 4
	A sentence is a string of terminal symbols. A sentential form is a string containing a mix of variables and terminal symbols or all variables.This is an intermediate form in doing a derivation.	
14	**Define Chomsky normal form?**	CO 4
	A grammar is said to be Chomsky Normal Form (CNF), if all its productions must derive either two non-terminals or a single terminal	
15	**State the pumping lemma for CFLs.**	CO 4
	Let L be any CFL. Then there is a constant n, depending only on L, such that if z is in L and $\mid z \mid \geq n$, then z=uvwxy such that : (i) $\mid vx \mid \geq 1$ (ii) $\mid vwx \mid \leq n$ and (iii) for all i $\geq$ 0 uviwxiy is in L.	
16	**List the types of normal forms?**	CO 4
	There are two types of normal form a)Chomsky Normal Form (CNF) b)Greibach Normal Form (GNF).	

17	**What are the properties of Context free languages?**	CO 4
	a)Context free languages are closed under union b) Context free languages are closed under concatenation c) Context free languages are closed under kleen closure d) Context free languages are closed under intersection e) Context free languages are closed under complement	
18	**Define syntax tree?**	CO 4
	Concrete syntax tree is an ordered, rooted tree that represents the syntactic structure of a string according to some context-free grammar.	
19	**List the applications of CFG?**	CO 4
	a)For defining programming languages b)For parsing the program by constructing syntax tree c)For translation of programming languages d)For describing arithmetic expression e)For construction of compilers	
20	**What are decisions to be considered during derivation?**	CO 4
	During parsing two decisions are taken. These are as follows: a) Decide the non-terminal which is to be replaced. b) Decide the production rule by which the non-terminal will be replaced.	
21	**List the properties of Derivation tree?**	CO 4
	a)The root node is always a node indicating start symbol b)The derivation is read from left to right c)The leaf nodes always terminals nodes d)The interior nodes are always non terminal nodes	
22	**What is nonterminal?**	CO 4
	Nonterminal symbols are those symbols which can be replaced. They may also be called simply syntactic variables. A formal grammar includes a start symbol, a designated member of the set of non-terminals from which all the strings in the language may be derived by successive applications of the production rules.	
23	**Define Terminal?**	CO 4
	Terminal symbols are literal symbols which may appear in the outputs of the production rules of a formal grammar and which cannot be changed using the rules of the grammar	
24	**List different types of derivations?**	CO 4
	There are two types as follows: a)Left-most Derivation b)Right-most Derivation	
25	**What is root vertex?**	CO 4
	Root vertex is always called as Non-terminal.It must be labeled by start symbol	

26	**Define vertex and leaves in Derivation tree?**	CO 4
	Vertex are labeled by a non terminal symbol Leaves are labeled by a terminal symbol or epsilon(empty symbol)	
27	**List the approaches for representing Derivation tree?**	CO 4
	There are two approaches a)Top-down approach b)Bottom-up approach	
28	**Define Top-down approach?**	CO 2
	This approach starts with the starting symbol S. and moves down from root node to leaf nodes using productions	
29	**Define Bottom-up approach?**	CO 2
	This approach start from leaf nodes .and it proceeds upwards to the root which is the starting symbol S	
30	**What is partial Derivation tree?**	CO 4
	Apartial derivation tree is a sub tree of a derivation tree such that all of its children are in the subtree or none of them are in the subtree	
MODULE IV		
PUSHDOWN AUTOMATA		
1	**What is the PDA? why it is developed ?**	CO 5
	PDA is a tool to implement CFLs .The FA has no memory to remember the count of input symbols or to match the relationship between the different types of symbols occurring in the string, that's why PDA is developed	
2	**Define Push Down Automata?**	CO 5
	A pushdown automaton is a way to implement a context-free grammar in a similar way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information. Basically a pushdown automaton is "Finite state machine" + "a stack" A pushdown automaton has three components • an input tape, • a control unit,and • a stack with infinite size.	
3	**How to represent the initial store of PDA ?**	CO 5
	A Special pushdown symbol called the initial symbol on push down store represented by Z0	
4	**How to represent the PDA**	CO 5
	PDA M = $(Q,\Gamma,\sum,\delta,q_0,Z_0,F)$	
5	**How to represent transition function in PDA**	CO 5
	$\delta : Q \times \sum \times \Gamma => Q \times \Gamma$	

6	**Name the 7 tuples of PDA** Q= Set Of All States Γ = stack symbols $\sum$ = input alphabet δ = transition function q_0 =start state Z_0 = Initial stack top symbol F = Final/accepting states	CO 5
7	**How many operations are performed on the stack ?** A stack does two operations Push —a new symbol is added at the top. Pop —the top symbol is read and removed.	CO 5
8	**How to represent the empty stack initially?** Initially we put a special symbol$into the empty stack.	CO 5
9	**Define Deterministic Context-free Language?** Deterministic CFL are subset of CFL which can be recognized by Deterministic PDA. Deterministic PDA has only one move from a given state and input symbol.	CO 5
10	**Show the notation of the empty string in PDA** If $\lvert$ S $\rvert$ = 0, it is called an empty string.(Denoted by λ or ϵ)	CO 5
11	**Show the notation of the input state in PDA ?** 	CO 5
12	**What is meant by empty stack acceptability** A PDA accepts a string when, after reading the entire string, the PDA has emptied its stack. For a PDA (Q,$\sum$, S, δ, q$_0, I, F$)the language accepted by the empty stack is $L(PDA)$=$\{$w$\mid$ (q_0, w, I) $\vdash^\star$ (q,ϵ, ϵ), q $\in$ Q $\}$	CO 5
13	**Show the notation of the output state in PDA ?** 	CO 5
14	**Define acceptance of PDA by Final State** In final state acceptability, a PDA accepts a string when, after reading the entire string, the PDA is in a final state. From the starting state, we can make moves that end up in a final state with any stack values. The stack values are irrelevant as long as we end up in a final state. For a PDA (Q,$\sum$, S, δ, q$_0, I, F$)the language accepted by the set of final states F is $L(PDA)$=$\{$w$\mid$ (q_0, w, I) $\vdash^\star$ (q,ϵ, x), q $\in$ F $\}$for any input stack string x.	CO 5

15	**Define acceptance of PDA by empty Stack?** In final state acceptability, a PDA accepts a string when, after reading the entire string, the PDA is in a final state. For a PDA $(Q,\sum$, S, δ, $q_0, I, F)$the language accepted by the empty stack is $L(PDA)$=$\{$w$|$ $(q_0, w, I) \vdash^\star (q,\epsilon, \epsilon)$, q $\in$ Q $\}$	CO 5
16	**Show the notation of the transition symbol between two states in PDA ?** $\longrightarrow$	CO 5
17	**Why input tape is used in PDA ?** Input tape is a buffer ,which is divided into cells and each cell stores a one input symbol from the give string input symbols are read by control head from left to right. $ is placed at the end of the string.	CO 5
18	**Why stack is used in the PDA ?** finite control reads the input symbols and places them in stack which is last in first out data structure	CO 5
19	**Why pop operation is used in PDA ?** while reading the input symbols from input tape based on the condition top of the stack symbol is popped from stack to recognize a particular string	CO 5
20	**why push operation is used in PDA ?** While reading the symbols from the input tape finite control will push them into the top of the stack.	CO 5
21	**What is Non Deterministic PDA ?** Heap allocation is used to dynamically allocate memory to the variables and claim it back when the variables are no more required. Memory allocation and deallocation can be done at any time and at any place depending on the requirement of the user.Conceptually NPDA is similar to NFA. in NPDA there could be multiple transitions for a unique input. Construction of NPDA is much more simplest than PDA	CO 5
22	**What is The Condition for converting CFG to PDA ?** in a production rule first symbol on the RHS is terminal symbol when we want to convert CGF to PDA	CO 5
23	**Why two stacks are used in PDA?** There are certain languages which cannot be accepted by PDA with single stack. This is a special case and when we want to perform two checks on the input strings we will use two stacks with PDA	CO 5
24	**Why finite automata is less powerful than PDA** since no stack concept is used in FA,That's why PDA is more powerful than FA	CO 5

25	**What is instantaneous description?**	CO 5
	Instantaneous description of a PDA is a triple of three parameters triple(q,w,y) -q is the state - w is the remaining input string -y is the stack contents	
26	**show the PDA machine configuration**	CO 5
	A machine configuration is an element of K$\times\sum^\star$ $\times$ $\Gamma^\star$. (p,w,γ) = (current state, unprocessed input, stack content).	
27	**show the condition to accept the string in PDA**	CO 5
	A string w is accepted by a PDA if and only if (s,w, e) $\vdash^\star$ (f, e, e)	
28	**What are the different types of language acceptances by a PDA and define them.**	CO 5
	For a PDA M=(Q,$\sum$, Γ,δ, q$_0$, Z_0, F)we define the language accepted by the final state L(M) as: $^\star\{w|$ $(q_0, w, Z_0) \vdash (p$,ϵ, $\gamma)$for some p in F and $\gamma in \Gamma^\star\}$.Language accepted by empty / null stackN(M) is: $\star$	
29	**What is the significance of PDA?**	CO 5
	Finite Automata is used to model regular expression and cannot be used to represent non regular languages. Thus to model a context free language, a Pushdown Automata is used.	
30	**When is a string accepted by a PDA?**	CO 5
	The input string is accepted by the PDA if: The finalstate is reached. The stack is empty.	
	MODULE V	
	TURING MACHINE	
1	**what is Turing machine**	CO 6
	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 ora 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.	
2	**Define recursively enumerable languages**	CO 6
	A language is recursively enumerable if there exists a Turing machine that accepts every string of the language, and does not accept strings that are not in the language.	
3	**Define recursive languages**	CO 6
	A language is recursive if there exists a Turing machine that accepts every string of tShe language and rejects every string that is not in the language.	

4	**What is Church's hypothesis**	CO 6
	The Church-Turing thesis (formerly commonly known simply as Church's thesis) says that any real-world computation can be translated into an equivalent computation involving a Turing machine.	
5	**Describe counter machine**	CO 6
	Counter machine has the same structure as the multi-stack machine but in place of each stack is a counter. Counters hold any non-negative integer,but we can only distinguish between zero and nonzero counters. That's the move of the counter machine depends on its state,input symbol and which if any of the counters are zero.	
6	**Define linear bounded automata**	CO 6
	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.	
7	**which generate context sensitive language**	CO 6
	Type-1 grammars generate context-sensitive languages.	
8	**List Chomsky hierarchy of languages**	CO 6
	One way showing the relationship among the languages if Chomsky hierarchy. Type 0 known as unrestricted grammar. Type 1 known as context sensitive grammar. Type 2 known as context free grammar. Type 3 Regular Grammar.	
9	**Define Regular Grammar.**	CO 6
	Type-3 grammars generate regular languages. Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal. The productions must be in the form $X \rightarrow a$ or $X \rightarrow aY$ where $X, Y \in N$(Non terminal) and $a \in T$(Terminal) The rule $S \rightarrow \in$ is allowed if S does not appear on the right side of any rule.	

10	**What are Type-2 grammars** Type-2 grammars generate context-free languages. The productions must be in the form $A \rightarrow \gamma$ where $A \in N$ (Non terminal) and $\gamma \in (T \cup N)^*$ (String of terminals and non-terminals). These languages generated by these grammars are be recognized by a non-deterministic pushdown automaton.	CO 6
11	**Describe Context-free grammar** A context-free grammar (CFG) consisting of a finite set of grammar rules is a quadruple (N, T, P, S) where N is a set of non-terminal symbols. T is a set of terminals where $N \bigcap T$ = NULL. P is a set of rules, P: $N \rightarrow (N \cup T)^*$, i.e., the left-hand side of the production rule.P does have any right context or left context. S is the start symbol.	CO 6
12	**Define Context-sensitive grammar** Context sensitive grammars generates context-sensitive languages. The productions must be in the form: $\alpha \rightarrow \beta$, with the following condition $\mid \beta \mid \geq \mid \alpha \mid$,i.e. length of β is greater than length of α.	CO 6
13	**Recall Unrestricted grammar** Type-0 grammars generate recursively enumerable languages. The productions have no restrictions. They are any phase structure grammar including all formal grammars. They generate the languages that are recognized by a Turing machine.	CO 6
14	**Define Pushdown automata** A pushdown automaton is a way to implement a context-free grammar in a similar way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information. Basically a pushdown automaton is – ”Finite state machine” + ”a stack” A pushdown automaton has three components an input tape, a control unit, and a stack with infinite size. The stack head scans the top symbol of the stack.	CO 6
15	**what is Multi-tape Turing Machine.** 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.	CO 6

16	**Waht is Turing machine with multiple tapes** 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.	CO 6
17	**Name Chomsky hierarchy of grammars** Type-0, Type-1, Type-2 and Type-3 grammars	CO 6
18	**How Turing machine with two dimensional tapes** Turing machine with two dimensional tapes that have one finite control, one read-write head and one two dimensional tape. The tape has the top end and the left end but extends indefinitely to the right and down. It is divided into rows of small squares.	CO 6
19	**What is Turing machine with infinite tape** a tape has infinite length and divided into cells	CO 6
20	**What is Deterministic Turing machine** Turing machines are a model of computation. It is believed that anything that can be computed can be by a Turing Machine	CO 6
21	**Overview of Multi-track Turing machines** 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.	CO 6
22	**Decribe Type 0 Grammar** Type 0 grammars generate recursively enumerable languages. The productions have no restrictions. They are any phase structure grammar including all formal grammars. They generate the languages that are recognized by a Turing machine.	CO 6
23	**Define Type 1 Grammar** Type 1 grammars generate context-sensitive languages. The productions must be in the form: $\alpha \rightarrow \beta$, with the following condition $\mid \beta \mid \geq \mid \alpha \mid$,i.e. length of β is greater than length of α.	CO 6
24	**Recall Type 2 Grammar** Type-2 grammars generate context-free languages. The productions must be in the form A $\rightarrow \gamma$ where A $\in$ N (Non terminal) and $\gamma \in$ (T $\cup$ N)* (String of terminals and non-terminals). These languages generated by these grammars are be recognized by a non-deterministic pushdown automaton.	CO 6

25	**Define Type 3 Grammar** Type-3 grammars generate regular languages. Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal. The productions must be in the form X → a or X → aY	CO 6
26	**What is Multiple Tracks** This is the final phase of compilation. which takes input as a optimized code and convert in to machine/assembly language.It is a tape, which is divided into multiple tracks, on the first track the input is placed is surrounded by # and $,using which we can perform arithmetic operations	CO 6
27	**Brifly about the Author of Turing Machine** Alan Turing OBE FRS was an English mathematician, computer scientist, logician, cryptanalyst, philosopher and theoretical Born: 23 June 1912, Maida Vale Died: 7 June 1954, Wilmslow, United Kingdom Award: Smith's Prize Education: Princeton University (1936–1938)	CO 6
28	**Describe about Left linear Grammar** In a left-linear grammar, all productions have one of the two forms: V→VT* or V→T* That is, the left-hand side must consist of a single variable, and the right-hand side consists of an optional single variable followed by any number of terminals.	CO 6
29	**Describe about Right linear Grammar** A strictly right-linear grammar is a context free grammar G where each rule has one of the following forms: $A \rightarrow xB$ for $x \in \sum$ $A \rightarrow 1$ Any derivation of a word w from a start symbol S in a strictly right-linear grammar has the form $S \Longrightarrow x1A1 \Longrightarrow x1x2A2 \Longrightarrow \cdots \Longrightarrow x1 ...xnAn \Longrightarrow x1 ...xn$ where $w = x1 ...xn$. Note that some of the xi may be .	CO 6
30	**Illustrate about Regular Grammar** Regular grammars generate regular languages. Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal. The productions must be in the form X → a or X → aY	CO 6

Course Coordinator: **HOD IT**
Dr.U.Sivaji