Subject Name: Compiler Design
Subject Code:3170701

Faculty: Mr. Ankit Patel & Mrs. Shivangi Desai

Sr. No	# CHAPTER NO- 1 :Introduction :	Marks
	Language processor, Applications of language processors, Definition-Structure-Working of compiler, the science of building compilers, Basic understanding of interpreter and assembler. Difference between interpreter and compiler. Compilation of source code into target language, Cousins of compiler, Types of compiler	
	SHORT QUESTIONS	
1	A compiler that runs on platform (A) and is capable of generating executable code for platform (B) is called ? **[LJIET]** A. cross-compiler B. complex-compiler C. object-compiler D. post-compiler E. one pass compiler F. Two pass compiler ANS: A	**1**
2	Which tool is used for grouping of characters in tokens in the compiler? **[LJIET]** A. Parser B. Code optimizer C. Code generator D. Scanner E. Symbol table F. None of the given ANS: D	**1**
3	Grammar of the programming is checked at _______ phase of compiler. **[LJIET]** A. Semantic analysis B. Syntax analysis C. Code optimization D. Code generation E. Code Optimization F. Lexical analysis ANS: B	**1**
4	Type checking is normally done during **[LJIET]** A. Lexical analysis B. Syntax analysis C. Semantic analysis D. Code optimization E. Code generation F. None of the given ANS: C	**1**
5	Which data structure in a compiler is used for managing information about variables and their attributes? **[LJIET]** A. Abstract syntax tree B. Symbol tree	**1**

	C. Semantic stack D. Symbol table E. Parse Tree F. Syntax tree ANS: D	
6	In a compiler, keywords of a language are recognized during **[LJIET]** A. parsing of the program B. the code generation C. during sematic analysis D. the lexical analysis of the program E. dataflow analysis F. none of the given ANS: D	1
7	Which compiler runs on one machine and generates code for multiple machines? **[LJIET]** A. complex-compiler B. object-compiler C. post-compiler D. one pass compiler E. cross-compiler F. Two pass compiler ANS: E	1
8	Which part of the compiler highly used the grammar concept? **[LJIET]** A. Parser B. Code optimizer C. Code generator D. Scanner E. Symbol table F. None of the given ANS: A	1
9	Symbol table can be used for: **[LJIET]** A) Checking type compatibility B) Suppressing duplication of error message C) Storage allocation D) All of these A. A and B both B. A and C both C. Only C D. Only A E. Only D F. B and C Both ANS: E	1
10	In a compiler the module that checks every character of the source text is called? **[LJIET]** A. Lexical analysis B. Syntax analysis C. Semantic analysis D. Code optimization E. Code generation F. None of the given ANS: A	1
11	How many parts of compiler are there? **[LJIET]** A. 1	1

	B. 2 C. 3 D. 4 E. 5 F. None of the given ANS: B	
12	_________ is a process of finding a parse tree for a string of tokens **[LJIET]** A. Parsing B. Analysing C. Recognizing D. Tokenizing E. Lexeme F. Lex ANS: A	1
13	What is the name of the process that determining whether a string of tokens can be generated by a grammar? **[LJIET]** A. Analysing B. Recognizing C. Translating D. Parsing E. Scanning F. None of the given ANS: D	1
14	In a compiler, keywords of a language are recognized during **[LJIET]** A. parsing of the program B. the code generation C. during sematic analysis D. the lexical analysis of the program E. dataflow analysis F. none of the given ANS: D	1
	DESCRIPTIVE QUESTIONS	
1	Explain the analysis synthesis model of compilation. List the factors that affect the design of compiler. Also List major functions done by compiler.**(May-2012)(Nov-2019)(Nov-2019_New)[LJIET]**	**06,07**
2	What does the linker do? What does the loader do? What does the preprocess do? Explain their role(s) in compilation process**.(May-2012)(Nov-2019) [LJIET]** List and Explain Cousins of Compiler. **(Dec-2018)[LJIET]** List the cousins of compiler and explain the role of any one of them. **(Nov-2018_New) [LJIET]**	**04,07, 3.5, 03**
3	Explain the roles of linker, loader and preprocessor. **(Nov-2013)[LJIET]**	**08**
4	What is a pass in a compiler? What is the effect of reducing the number of passes? **(Nov-2011)(Nov-2017_New) [LJIET]**	**04, 03**
5	Explain lexical analysis phase of a compiler and, for a statement given below, write output of all phases (except of an optimization phase) of a complier. Assume a, b and c of type float a = a + b * c * 2; **.(Dec-2012)[LJIET]**	**07**
6	Draw structure of Compiler. Also explain Analysis Phase in brief**.(Nov-2013)[LJIET]**	**07**
7	What is the pass of a compiler? Explain how the single and multi-pass compilers work**.(May-2014)[LJIET]**	**07**
8	Explain the phases of compiler with an example**.(May-2015)[LJIET]** Explain different phases of compiler.**(May-2016, May-2019,OCT-2020_New)[LJIET]** Describe all phases of a compiler.**(Nov-2016)[LJIET]**	**07**

	Explain various phases of compiler with example.**(May-2017) [LJIET]** Explain phases of compiler with example. **(April-2018_New)[LJIET]** Discuss the phases of a compiler with sketch. **(Dec-2018)[LJIET]** Explain front end and back end of compiler in detail. **(Nov-2018_New)[LJIET]** Explain phases of compilers with suitable example. **(Jan-2021_NEW)[LJIET]**	
9	Explain Semantic Analysis and Syntax Analysis phases of compiler with suitable example. Also explain the reporting errors by these two phases.**(Dec-2015)[LJIET]**	**07**
10	Explain linker & loader.**(Nov-2016,May-2019)[LJIET]**	**03**
11	For a statement given below, write output of all phases (except that of optimization phase) of a complier.**(Nov-2016) [LJIET]** a = a + b * c ;	**07**
12	Explain Semantic analysis and Syntax analysis phases of compiler with suitable example. Also explain the errors generated by these two phases.**(April-2017_New) [LJIET]**	**07**
13	What is the difference between compiler and interpreter? **(Nov-2017_New, May-2019_NEW) [LJIET]**	**03, 04**
14	Explain analysis phase of the source program with example. **(Nov-2017_New)[LJIET]**	**04**
15	Define cross compiler, token and handle. **(April-2018_New)[LJIET]**	**03**
16	Define following terms: **(May-2018)[LJIET]** i. Compiler ii. Interpreter iii. Assembler	**03**
17	Explain analysis of source program for compilers. **(May-2019_NEW)[LJIET]**	**07**
18	Compare and contrast compilers and interpreters. **(Jan-2021_NEW)[LJIET]**	**03**
19	Explain with suitable example what is bootstrapping? **(OCT-2020_New)[LJIET]**	**04**
20	What is Compiler? Explain working of all the phases of compiler with a sample string as input. **(AUG-2021_Old)[LJIET]**	**07**
21	Briefly explain the role of linker and loader in language processing activity. **(AUG-2021_NEW)[LJIET]**	**04**
Sr. No	**CHAPTER NO- 2 :Lexical Analyzer:**	**Marks**
	The Role of the Lexical Analyzer, Specification of Tokens, Recognition of Tokens, Input Buffering, elementary scanner design and its implementation (Lex), Applying concepts of Finite Automata for recognition of tokens.	
	SHORT QUESTIONS	
1	If the lexical analyzer finds a token invalid then? **[LJIET]** A. It generates an exception B. It generates an warning C. It generates an error D. Reads the whole program E. Read next token F. None of the given ANS: C	**1**
2	How many tokens will be generated by the scanner for the following statement ? x = x ∗ (a + b) – 5; **[LJIET]** A. 12 B. 11 C. 7 D. 8	**1**

	E. 10 F. 5 ANS: A	
3	The number of tokens in the following C statement is printf("i = %d, &i = %x", i, &i); **[LJIET]** A. 3 B. 26 C. 10 D. 21 E. 9 F. 11 ANS: C	1
4	Consider the following statements: **[LJIET]** (I) The output of a lexical analyzer is groups of characters. (II) Total number of tokens in printf("i=%d, &i=%x", i, &i); are 14. (III) Symbol table can be implementation by using array and hashtable but not tree. Which of the following statement(s) is/are correct? A. Only (I) B. Only (II) C. Only (III) D. Only (II) and (III) E. All (I), (II), and (III) F. None of these ANS: F	1

5	For the R.E ($a^* + b^*$)*# construct the syntax tree and what will be the FIRSTPOS of root. If you give leaf node number starting from 1 **[LJIET]**
	A. {1,2}
	B. {1,2,3}
	C. { Φ }
	D. {2}
	E. {1}
	F. None of the given
	ANS: B
		1

6	For the R.E a* b* (c + d) f # construct the syntax tree and what will be the FIRSTPOS of root. If you give leaf node number starting from 1 **[LJIET]**
	A. {1,2}
	B. {1,2,3}
	C. {1,2,3,4 }
	D. {2}
	E. {1}
	F. None of the given
	ANS: C
		1

7	Which one is the correct rule for finding FIRSTPOS of node in construction of DFA using syntax tree method? **[LJIET]** A. if (nullable(c1) == true) then { firstpos(c1) U firstpos(c2) } else firstpos(c1) B. if (nullable(c1) == true) then { firstpos(c1) U firstpos(c2) } else firstpos(c2) C. if (nullable(c1) == true) then { firstpos(c1) U firstpos(c2) } else lastpos(c2) D. if (nullable(c1) == true) then { firstpos(c1) U firstpos(c2) } else E. lastpos(c1) F. if (nullable(c2) == true) then { firstpos(c1) U firstpos(c2) } else firstpos(c1) G. None of the given ANS: A	1

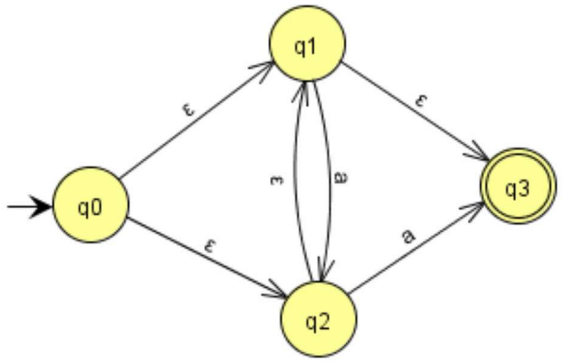
8	Which one is the correct rule for finding LASTPOS of node in construction of DFA using syntax tree method? **[LJIET]** A. if (nullable(c1) == true) then { lastpos(c1) U lastpos(c2) } else lastpos(c2) B. if (nullable(c2) == true) then { lastpos(c1) U lastpos(c2) } else lastpos(c2) C. if (nullable(c2) == true) then { lastpos(c1) U lastpos(c2) } else lastpos(c1) D. if (nullable(c1) == true) then { lastpos(c1) U lastpos(c2) } else Firstpos(c2) E. if (nullable(c2) == false) then { lastpos(c1) U lastpos(c2) } else lastpos(c2) F. None of the given ANS: B	**1**

9	For the R.E $a^* b^* (c + d)^* f$ # construct the syntax tree and what will be the FIRSTPOS of root. If you give leaf node number starting from 1 **[LJIET]**
	A. {1,2,3,4,5}
	B. {1,2,3}
	C. {1,2,3,4,5,6}
	D. {1,2}
	E. {1, 2,3,4}
	F. None of the given
	ANS: A
		1

10	For the R.E $(a + b)^*abb^*$ # construct the syntax tree and what will be the FIRSTPOS of root. If you give leaf node number starting from 1 **[LJIET]**
	A. {1,2}
	B. {1,2,3}
	C. { Φ }
	D. {2}
	E. {1}
	F. None of the given
	ANS: B
		1

11	Consider the following statements: **[LJIET]** (I) The output of a lexical analyzer is groups of characters. (II) Total number of tokens in printf("i=%d, &i=%x", i, &i); are 10. (III) Symbol table can be implementation by using array and hashtable but not tree. Which of the following statement(s) is/are correct? A. Only (I) B. Only (II) C. Only (III) D. Only (II) and (III) E. All (I), (II), and (III) F. None of these ANS: B	**1**
12	The number of tokens in the following C statement is printf("i = %d, &i = %x,%d", i, &I, i); **[LJIET]** A. 3 B. 26 C. 12 D. 21 E. 31 F. 13 ANS: C	**1**
13	How many tokens will be generated by the scanner for the following statement ? d = d * (a + b + c) – 5; **[LJIET]** A. 14 B. 13	**1**

	C. 6 D. 8 E. 10 F. 5 ANS: A	
14	When the lexical analyzer read the source-code, it scans the code? **[LJIET]** A. line by line B. word by word C. letter by letter D. reads the whole program E. read row by row F. read column by column ANS: C	**1**
	DESCRIPTIVE QUESTIONS	
1	Write a regular definition for the language of all strings of 0's and 1's with an even number of 0's and odd number of 1's. Write an algorithm for eliminating left recursion**.(Nov-2011)[LJIET]**	**07**
2	Write the two methods used in lexical analyzer for buffering the input. Which technique is used for speeding up the lexical analyzer? **(Nov-2011)[LJIET]** Write a brief note on input buffering techniques to Lexical Analyzer.**(Nov-2013)(Nov-2019) [LJIET]** Write a brief note on input buffering techniques**.(May-2014) (Nov-2018_New(May-2019)** (**Nov-2019_New) [LJIET]** Explain Buffer pairs and Sentinels. **(Nov-2014)[LJIET]** Write a short note on input buffering methods.**(Nov-2016_New)[LJIET]** Explain input buffering methods.**(May-2017)[LJIET]** Write a short note on input buffering method. **(April-2018_New)[LJIET]** Explain Input Buffering in detail with example. Also explain importance of sentinel character. **(AUG-2021_Old)[LJIET]**	**07, 04**
3	Write a brief note on input buffering techniques (**Nov-2018_New)(OCT-2020_New)[LJIET]**	**04,03**
4	Explain Buffer pairs and Sentinels.**(Nov-2017_New)[LJIET]**	**04**
5	Discuss Input Buffer Pairs for Lexical Analyzer. **(Dec-2018)[LJIET]**	**3.5**
6	Find the Regular Expression corresponding to given statement, subset of {0,1}* **(May-2012)** i. The Language of all strings containing at least one 0 and at least one ii. The Language of all strings containing 0's and 1's both are even. iii. The Language of all strings containing at most one pair of consecutive 1's. iv. The Language of all strings that do not end with 01**.**	**07**
7	How do the parser and scanner communicate? Explain with the block diagram communication between them. Also explain: What is input buffering**?(May-2012)(May-2018) [LJIET]**	**07**
8	Convert the following NFA-ε into equivalent NFA. Here ε is a Λ-transition**.(May-2012)[LJIET]**	**07**

		
9	Construct a DFA for a given regular expression (010+00)*(10)*.**(May-2012)(Nov-2019)(OCT-2020_New)[LJIET]**	07
10	Construct a DFA without constructing NFA for following regular expression. Find minimized DFA. a*b*a(a | b)*b*a#.**(Dec-2012)[LJIET]**	07
11	Construct a NFA for following regular expression using Thompson's notation and then convert it into DFA. aa*(b | c) a*c#.**(Dec-2012)[LJIET]**	07
12	Draw Deterministic Finite Automata for the binary strings ending with 10.**(Nov-2013)[LJIET]**	04
13	Write down the regular expression for the binary strings with even length.**(Nov-2013)**	03
14	Draw Deterministic Finite Automata for **:(May-2014)[LJIET]** 1. (0+1)*101(0+1)* 2. 10(0+1)*1	07
15	List out phases of a compiles. Write a brief not on Lexical Analyzer.**(May-2014)[LJIET]**	06
16	Draw Transition diagram of following:**(Nov-2014)[LJIET]** i. relational operators. ii. unsigned operator.	07
17	Construct minimum state DFA's for following regular expressions.**(Nov-2014)[LJIET]** i. (a|b)*a(a|b) ii. (a|b)*a(a|b) (a|b)	07
18	What are regular expressions? Find the regular expression described by DFA {{A,B},{0,1},δ,A,{B}}, where δ is detailed in following table .**(May-2015)[LJIET]**	07

	0	1
A	A	B
B	ϕ	A

Please note B is accepting state. Describe the language defined by the regular expression.

19	Construct the NFA using thompson's notation for following regular expression and then convert it to DFA. a+ (c |d) b*f # .**(May-2015)[LJIET]**	07
20	Construct DFA for following Regular expression. Use firstpos, lastpos and followpos functions to construct DFA.**(Dec-2015)[LJIET]** (a * | b *) *	07
21	Construct NFA for following Regular Expression using Thomson's Construction. Apply subset construction method to convert into DFA.**(Dec-2015)[LJIET]** (a | b)*abb Draw the DFA for the regular expression (a|b)*abb using set construction method only.**(May-2016)[LJIET]** Write an algorithm for Thompson's construction method. Apply the algorithm to construct NFA for following regular expression. (a | b)*abb. **(Nov-2017_New)[LJIET]**	07
22	What is regular expression, give all the algebraic properties of regular expression.**(May-2016)[LJIET]**	07

23	Draw the state transition diagram for the unsigned numbers.**(May-2016)[LJIET]**	07
24	Unsigned numbers are strings such as 5280, 39.37, 6.336E4 or 1.894E-4, give the regular definitions for the above mentioned strings.**(May-2016)[LJIET]**	07
25	Convert the (a|b|c)*d*(a*|b)ac+# regular expression to DFA directly and draw its DFA.**(May-2016)[LJIET]**	07
26	Convert the following regular expression into deterministic finite automata. (a+b)*abb(a+b)* **.(Nov-2011)[LJIET]**	03
27	Construct DFA by syntax tree construction method .**(May-2015)[LJIET]** a+ b* (c |d) f # Optimize the resultant DFA.	07
28	Construct DFA without constructing NFA for following regular expression: **(Nov-2016)(Nov-2019_New) [LJIET]** a*b*a(a | b)b*a# Minimize the same**.**	07
29	Construct NFA for following regular expression using Thompson’s notation and then convert it into DFA.**(Nov-2016)[LJIET]** a(b | c)*a*c#	07
30	Define the following terms: **(Nov-2016_New)[LJIET]** 1. Token 2. Pattern 3. Lexeme 4. Ambiguous grammar 5. Handle pruning 6. Compiler 7. DAG	07
31	Explain subset construction method for constructing DFA from an NFA with an example.**(Nov-2016_New)(Dec-2018)[LJIET]** Explain subset construction method with an example. **(May-2017)(April-2018_New) [LJIET]**	07
32	Construct DFA for the following regular expression using syntax tree with firspos, laspos and followpos function.**(Nov-2016_New)[LJIET]** (a | b) *a Draw DFA for the following regular expression using firstpos( ), lastpos ( ) and followpos ( ) functions. **(April-2018_New)[LJIET]** (a | b) * a Construct a DFA without constructing NFA for the following regular expression. (a | b) * a. **(May-2017,May-2019)[LJIET]**	07
33	Construct the NFA using thompson’s notation for following regular expression and then convert it to DFA.**(April-2017_New)[LJIET]** (a / b)* ab#	07
34	Explain role of lexical analyzer. **(Nov-2017_New)[LJIET]**	04
35	Draw transition diagram for relational operators. **(April-2018_New)[LJIET]**	04
36	Construct DFA for following regular expression without constructing NFA and optimize the same. **(May-2018)[LJIET]** (a | ε)* a b (a | b)* #	07
37	Explain automatic generation of lexical analyzer and parser. **(May-2018)[LJIET]**	07
38	Define following terms: **(May-2018)[LJIET]** i. Regular Expression ii. Token iii. Lexeme iv. Pattern	04

39	Construct a DFA without constructing NFA for the following regular expression. **(Dec-2018) [LJIET]** a(a | b)*ab	07
40	Construct NFA for following regular expression using Thompson's notation and then convert it into DFA. **(Nov-2018_New,May-2019)[LJIET]** (a/b)*abb#	07
41	Define lexemes, patterns and tokens. **(May-2019_NEW)[LJIET]**	03
42	Give regular definition for signed and unsigned numbers. **(May-2019_NEW)[LJIET]**	03
43	Draw DFA from regular expression without constructing NFA. (a | b | c)* a (b | c)* # **(May-2019_NEW)[LJIET]**	07
44	Draw NFA from regular expression using Thomson's construction and convert it into DFA. (a | b)* a b* a **(May-2019_NEW)[LJIET]**	07
45	Explain tokens, lexemes, Pattern with example. **(Nov-2019_NEW)[LJIET]**	03
46	Write a regular definition for 1. The Language of all strings that do not end with 01. 2. All strings of digit that contain no leading 0's. **(Nov-2019_NEW)[LJIET]**	03
47	Construct a DFA for a given regular expression (a|b)*abb. **(Nov-2019_NEW)[LJIET]**	07
48	Write a short note on LEX Tool. **(Jan-2021_NEW)[LJIET]**	04
49	Define terms: pattern, lexeme, token **(Jan-2021_NEW)[LJIET]**	03
50	Write any one method used in lexical analyzer for buffering the input. **(Jan-2021_NEW)[LJIET]**	04
51	Construct nondeterministic finite automata by using Thomson's Construction for following regular expression. Show the sequence of moves made in processing the input string ababbab. (a|b)* abb **(Jan-2021_NEW)[LJIET]**	07
52	Construct DFA accepting the strings of binary digits which are even numbers. **(OCT-2020_New)[LJIET]**	07
53	Draw DFA for given expression- aa*ab*c # **(AUG-2021_Old)[LJIET]**	07
54	Prove give regular expressions are equal by generating their optimized DFAs. 1) (a | b)* 2) (a* | b*)* **(AUG-2021_Old)[LJIET]**	07
55	What is the role of lexical analyzer? Justify your answer with example. **(AUG-2021_NEW)[LJIET]**	03
56	Convert the following regular expression to DFA using subset construction method. ((0|1)(0|1))*# **(AUG-2021_NEW)[LJIET]**	07
57	Write CFG for the following languages: i. {ambm: m>=n} ii. generates strings having equal number of a's and b's over symbols {a,b} **(AUG-2021_NEW)[LJIET]**	03
Sr. No	**CHAPTER NO- 3 :Parsing Theory:**	**Marks**
	TOPIC:1 Top Down Parsing Algorithms	
	SHORT QUESTIONS	
1	The grammar A → Aa | Aab | ^ is not suitable for predictive-parsing because the grammar is? "Note that ^ is NULL symbol" **[LJIET]** A. ambiguous B. left-recursive C. left-factoring D. right-recursive and ambiguous	1

	E. left-factoring and left-recursive F. None of these ANS: E	
2	Consider the grammar given below: **[LJIET]** S → Aa A → BD B → b | ^ D → d | ^ Compute the FOLLOW set of the non-terminal B Note:"^ is NULL Symbol" A. {a, b, d, $} B. {d} C. {a, d, $} D. {a, d} E. { d, $} F. None of these ANS: D	**1**
3	Consider the following grammar: **[LJIET]** S → FR R → S | ε F → id In the predictive parser table, M, of the grammar the entries M[S, id] and M[R, $] respectively Note: "ε" is NULL symbol. **[LJIET]** A. {S → FR} and {R → ε } B. {S → FR} and {error} C. {S → FR} and {R → *S} D. {F → id} and {R → ε} E. {error} and {error} F. None of the given ANS: A	**1**
4	S → AB A → Ca | ε B → BaAC | c C → b | ε What is the FOLLOW set of A Note: "ε" is NULL symbol. **[LJIET]** A. {a, b, c} B. {a, b, c, $} C. {b, c} D. {a, $} E. {a, b} F. {a, b, c, ε } ANS: B	**1**
5	Which of the following derivations does a top-down parser use while parsing an input string? The input is assumed to be scanned in left to right order **[LJIET]** A. Rightmost derivation. B. Leftmost derivation traced out in reverse. C. Rightmost derivation traced out in reverse. D. Leftmost derivation. E. Left Recursion F. None of the given	**1**

	ANS: D	
6	S → AcB | cbB | Ba A → da | BC B → g | ε C → h | ε What is the FIRST set of S **[LJIET]** Note: “ε” is NULL symbol. A. {c, d, g, ε } B. {a, d, ε } C. { $ } D. {a, c, d, g, h, ε } E. {a, c, d, g, h} F. None of the given ANS: E	**1**
7	S → AaAb | BbBa A → ε B → ε What is the FIRST set of A **[LJIET]** Note: “ε” is NULL symbol. A. {b, ε} B. {a, ε} C. { ε } D. {a, b, ε} E. {a, b } F. None of the given ANS: C	**1**
8	E → TE’ E’ → +TE’ | ε T → FT’ T’ → *FT’ | ε F → (E) | id What is the FOLLOW set of F **[LJIET]** Note: “ε” is NULL symbol. A. { +, *,), $} B. {), $} C. { +, *, (, $} D. { +,), $} E. { +, *,) } F. None of the given ANS: A	**1**

9	Consider the grammar shown below
	S → iEtSS’ | a
	S’ → eS | ε
	E→b
	In the predictive parse table, M, of this grammar, the entries M[S’,e] and M[S’,ϕ] respectively are: **[LJIET]**
	A. {S’ → eS } and {S’→ε }
	B. {S’ → eS } and { }
	C. {S’ → ε } and {S’ → ε }
	D. { } and { }
	E. {S’ → eS , S’ → ε } and {S’ → ε }
		1

	F. None of the given ANS: E	
10	Which one is the correct answer after removing left recursion of given grammar **[LJIET]** A → ABd | Aa | a A. A → aA' A' → BdA' | a | ε B. A → aA' , A' → BdA' | aA' | ε C. A → aA' | ε , A' → BdA' | a D. A → aA' | A' , A' → BdA' | aA' E. A → aA' , A' → BdA | aA | ε F. None of the given ANS: B	**1**
11	Which of the following be sufficient to convert an arbitrary CFG to an LL(1) grammar **[LJIET]** A. Removing left recursion alone B. Factoring the grammar alone C. Removing left recursion and factoring the grammar D. Removing ambiguity only E. Removing all the terminals F. None of these ANS: F	**1**
12	Which one is the correct answer after removing left recursion of given grammar **[LJIET]** E → E + T | E * T T → id A. E → +TE' | *TE' | ε, E' → TE', T → id B. E → TE', E' → +TE' | *TE' | ε, T→ id C. E → +TE' | *TE', E' → TE' | ε, T → id D. AE→ +TE' | *T | ε, E' → TE', T → id E. E → TE', E' → +T | *TE' | ε, T→ id F. None of the mention ANS: B	**1**
13	Which one is the correct answer after removing left factoring of given grammar **[LJIET]** A → aAb | aA | a A. A → aA' , A' → Ab | A | ε B. A → aA' | ε, A' → Ab | A C. A → aA', A' → AA'' | ε, A'' → b | ε D. A → aA', A' → AA'' , A'' → b | ε E. A → aA', A' → AA'' | ε, A'' → b F. None of the mentioned ANS: C	**1**
14	A grammar that produces more than one parse tree for some sentence is called **[LJIET]** A. Ambiguous B. Unambiguous C. Regular D. Left recursive E. Left factoring F. None of the mentioned	**1**

	ANS: A	
15	What is the name of the process that determining whether a string of tokens can be generated by a grammar? **[LJIET]** A. Analysing B. Recognizing C. Translating D. Parsing E. Scanning F. None of the given ANS: D	1
16	__________ is a process of finding a parse tree for a string of tokens. **[LJIET]** A. Parsing B. Analysing C. Recognizing D. Tokenizing E. Lexeme F. Lex ANS: A	1
17	Which one is the correct answer after removing left recursion of given grammar **[LJIET]** S → Aa | b A → Ac | Sd | f A. S → Aa | b, A → SdA' | fA' , A' → cA' | ε B. S → Aa | b, A → cA' | ε, A' → SdA' | fA' C. S → Aa | b, A → bdA' | fA' , A' → cA' | adA' | ε D. S → faS' | bS' | AcaS', S' → daS' | ε, A → cA' | ε, A' → SdA' | fA' | ε E. There is no left recursion in given grammar F. None of the given ANS: C	1
18	The grammar A → AA | (A) | ^ is not suitable for predictive-parsing because the grammar is? "Note that ^ is NULL symbol" **[LJIET]** A. ambiguous B. left-recursive C. left-factoring D. right-recursive ambiguous E. ambiguous and left-recursive F. None of these ANS: E	1
19	Consider the grammar given below: S → AB A → BD B → b | ^ D → d | ^ Compute the FOLLOW set of the non-terminal D **[LJIET]** Note:"^ is NULL Symbol" A. {b, d, $} B. {d} C. { d, $} D. {b, d} E. { $} F. None of these	1

	ANS: F	
20	Which of the following derivations does a top-down parser use while parsing an input string? The input is assumed to be scanned in left to right order. **[LJIET]** A. Leftmost derivation. B. Leftmost derivation traced out in reverse. C. Rightmost derivation. D. Rightmost derivation traced out in reverse. E. Left Recursion F. None of the given ANS: A	1
21	Consider the following grammar: S → FR R → S | ε F → id In the predictive parser table, M, of the grammar the entries M[R, id] and M[R, $] respectively **[LJIET]** Note: “ε” is NULL symbol. A. {S → FR} and {R → ε } B. {R → S} and { R → ε } C. {S → FR} and {R → ε} D. {F → id} and {error} E. {error} and {error} F. None of the given ANS: B	1
22	S → AB A → Ca | ε B → BaAC | c C → b | ε What is the FIRST set of S **[LJIET]** Note: “ε” is NULL symbol. A. {a, c} B. { b, ε } C. {a, b, c, ε} D. { $} E. {a, b, c} F. {a, b, ε } ANS: E	1
23	S → AcB | cbB | Ba A → da | BC B → g | ε C → h | ε What is the FOLLOW set of B **[LJIET]** Note: “ε” is NULL symbol. A. {a, $ } B. {a, h, $ } C. {a, h, $, ε } D. {a, h, d, c, $ } E. {a, c, h, $ } F. None of the given ANS: E	1
24	S → AaAb | BbBa	1

A → ε
B → ε
What is the FIRST set of S **[LJIET]**
Note : "ε" is NULL symbol.

A. {b, ε}
B. {a, ε}
C. { ε }
D. {a, b, ε}
E. {a, b }
F. None of the given

ANS: E

25 E → TE'
E' → +TE' | ε
T → FT'
T' → *FT' | ε
F → (E) | id
What is the FOLLOW set of T **[LJIET]**
Note: "ε" is NULL symbol.

A. { +, *,), $}
B. {), $}
C. { +, *, (, $}
D. { +,), $}
E. { +, *,) }
F. None of the given

ANS: D **1**

26 Consider the grammar shown below
S → iEtSS' | a
S' → eS | ε
E→b
In the predictive parse table, M, of this grammar, the entries M[S',e] and M[S',ϕ] respectively are: **[LJIET]**

A. {S' → eS } and {S' → ε }
B. {S' → eS } and { }
C. {S' → ε } and {S' → ε }
D. { } and { }
E. {S' → eS , S' → ε } and {S' → ε }
F. None of the given

ANS: E **1**

27 A grammar that produces more than one parse tree for some sentence is called **[LJIET]**

A. Ambiguous
B. Unambiguous
C. Regular
D. Left recursive
E. Left factoring
F. None of the mentioned

ANS: A **1**

28 Which of the following be sufficient to convert an arbitrary CFG to an LL(1) grammar **[LJIET]** **1**

A. Removing left recursion alone
B. Factoring the grammar alone
C. Removing left recursion and factoring the grammar
D. Removing ambiguity only

	E. Removing all the terminals F. None of these ANS: F	
29	Which one is the correct answer after removing left recursion of given grammar **[LJIET]** A → A + T | A * T T → id A. A → +TA' | *TA' | ε , A' → TA' , T → id B. A → TA' , A' → +TA' | *TA' | ε , T→ id C. A → +TA' | *TA' , A' → TA' | ε , T → id D. A → +TA' | *T | ε , A' → TA' , T → id E. A → TA' , A' → +T | *TA' | ε , T→ id F. None of the mention ANS: B	**1**
30	Which one is the correct answer after removing left factoring of given grammar **[LJIET]** A → ad | a | ab | abc | b A. A → aA' , A' → d | b | bc | ε B. A → aB' | b , B' → bB'' | d | ε , B'' → c | ε C. A → aB' | b , B' → bB'' | d , B'' → c D. A → aB' | b , B' → bB'' | d | ε , B'' → c E. A → aA' | ε , A' → d | b | bc , F. None of the mentioned ANS: B	**1**
31	Which one is the correct answer after removing left recursion of given grammar **[LJIET]** S → Aa | b A → Ac | Sd | f S → Aa | b A. A → SdA' | fA' , A' → cA' | ε B. S → Aa | b, A → cA' | ε, A' → SdA' | fA' C. S → Aa | b, A → bdA' | fA' , A' → cA' | adA' | ε D. S → faS' | bS' | AcaS', S' → daS' | ε, A → cA' | ε, A' → SdA' | fA' | ε E. There is no left recursion in given grammar F. None of the given ANS: C	**1**
32	Which one is the correct answer after removing left recursion of given grammar **[LJIET]** A → ABd | Aa | a | b A. A → aA' | bA' , A' → BdA' | a | ε B. A → aA' | bA' , A' → BdA' | aA' C. A → aA' | bA' , A' → BdA' | aA' | ε D. A → aA' | bA' , A' → BdA' | aA' | ε E. A → aA' , A' → BdA | aA | ε F. None of the given ANS: C	**1**
	DESCRIPTIVE QUESTIONS	
1	Is the following grammar suitable for LL(1) parsing? If not make it suitable for LL(1) parsing. Compute FIRST and FOLLOW sets. Generate the parsing table.**(Nov-2011)[LJIET]**	**07**

	S→AB A→Ca | ϵ B→BaAC | c C→b | ϵ	
2	Draw the transition diagrams for predictive parsers for the following grammar.**(Nov-2011) [LJIET]** E → TE' E'→ +TE' | ϵ T → FT' T' → *FT' | ϵ F → (E) | id	**07**
3	Explain non-recursive predictive parsers. Draw the block diagram of it.**(May-2012)(Nov-2019_New)[LJIET]** Explain working of non-Recursive Predictive Parser with diagram. **(Dec-2018)[LJIET]**	**04, 07**
4	Construct predictive parsing table for following.**(May-2012,May-2019)[LJIET]** S -> A A -> aB | Ad B -> bBC | f C -> g	**07**
5	Eliminate left recursion from the following grammar and rewrite the Grammar.**(May-2012)(Nov-2019) [LJIET]** S -> Aa | b A -> Ac | Sd | ε	**03,07**
6	Perform the Left factoring of following Grammar: **(May-2012)[LJIET]** A →ad | a | ab | abc | b	**04**
7	Find out FIRST & FOLLOW set for all the Nonterminals.**(May-2012)[LJIET]** S→ AcB | cbB | Ba A→da | BC B→g | ε C→h | ε	**06**
8	Test whether the following grammar is LL (1) or not. Construct predictive parsing table for it.**(May-2012)(Nov-2019) [LJIET]** S→1AB | ε A→1AC | 0C B→0S C→1	**07**
9	Write unambiguous grammar for producing arithmetic expression consisting of symbols id, +, - , /, $. Find first & follow of non terminal symbols of the grammar for non recursive predictive parser. Construct parse table and parse following string.**(Dec-2012)[LJIET]** id – id + id – id $.	**07**
10	Write down C program for Recursive Descend Parser for **:(Nov-2013)[LJIET]** S → ABC B→ 1B | Λ A→ 0A1 | Λ C→1C0 | Λ	**07**
11	Write down the algorithm for left factoring.**(Nov-2013)[LJIET]**	**03**
12	Draw parsing table for Table Driven Parser for the given grammar. Is the grammar LL(1)? A→ AaB | x B→ BCb | Cy C→ Cc | Λ **(Nov-2013)[LJIET]**	**07**
13	What do you understand by a handle? Explain the stack implementation of shift reduce parser with the help of example.**(Nov-2014)[LJIET]**	**07**
14	Explain recursive-descent and predictive parsing..**(Nov-2014)[LJIET]**	**07**
15	Implement the following grammar using Table Driven parser and check whether it is LL(1) 14or not. S -> aBDh, B -> cC, C -> bC / ^, D -> EF, E -> g / ^, F-> f / ^ .**(May-2014)[LJIET]**	**07**

16	Implement the following grammar using Recursive Descent Parser.**(May-2014)(Nov-2018_New)[LJIET]** S -> Aa | bAc | bBa, A -> d, B -> d	**07**
17	For the following grammar D → T L ; L → L , id | id T → int | float **(May-2015)[LJIET]** 1)Remove left recursion (if required) 2)Find first and follow for each non terminal for Resultant grammar 3)Construct LL(1) parsing table 4)Parse the following string (show stack actions clearly) and draw parse tree for the input: int id, id;	**07**
18	Develop a predictive parser for the following grammar.**(May-2015)[LJIET]** S'->S S->aA|b|cB|d A->aA|b B->cB|d	**07**
19	How top down and bottom up parser will parse the string 'bbd' using grammar A → bA | d. Show all steps clearly.**(May-2015)[LJIET]**	**07**
20	Construct LL(1) parsing table for the following Grammar:**(Dec-2015)[LJIET]** S → (L) | a L → L , S | S	**07**
21	Define: Left Recursive. State the rule to remove left recursive from the grammar. Eliminate left recursive from following grammar.**(Dec-2015)[LJIET]** S → Aa | b A → Ac | Sd | f	**07**
22	What is left recursion? Eliminate the left recursion from the following grammar.**(May-2016,May-2019)[LJIET]** E → E + T | T T → T * F | F F → (E) | id	**07**
23	Design the FIRST SET and FOLLOW SET for the following grammar.**(May-2016)[LJIET]** E → E + T | T T → T * F | F F → (E) | id	**07**
24	(i) Compare top-down and bottom-up parser. (ii) Explain right-most-derivation-in-reverse with the help of an example. **(Nov-2016,May-2019)[LJIET]**	**07**
25	Differentiate Top Down Parsing and Bottom up parsing **(Nov-2018_New,May-2019_NEW) [LJIET]**	**04,03**
26	Draw transition diagrams corresponding to production rules for arithmetic expressions consisting of operators + and ^ for predictive parser. Explain how parsing takes place for the same using transition diagrams.**(Nov-2016)[LJIET]**	**07**
27	Explain left factoring with the help of an example. **(Nov-2016)[LJIET]** Write a rule of Left factoring a grammar and give example. **(Nov-2017_New)[LJIET]** What is left factoring? Discuss it with the help of an example. **(Dec-2018)[LJIET]**	**04, 03, 07**
28	Construct LL(1) Parsing table for the following grammar. Also show moves made by input string : abba.**(Nov-2016_New)[LJIET]** S → aBa B →bB | ^	**07**
29	Check following grammar is LL (1) or not?**(April-2017_New)[LJIET]** S -> aB | € B -> bC | € C -> cS | €	**07**
30	What is left factoring and left recursion? Explain it with suitable example.**(April-2017_New)[LJIET]**	**07**
31	How do you check whether the grammar is LL (1) or not? Justify your answer with appropriate example. **(May-2017)[LJIET]**	**07**

32	Check the following grammar is left recursive or not. Justify your answer. If Left recursive then make grammar as non-left recursive. **(April-2018_New)[LJIET]** S → (L) | a L → L , S | S	04
33	Consider the following grammar and construct the corresponding left most and right most derivations for the sentence abab. **(April-2018_New)[LJIET]** S -> aSbS | bSaS | €	03
34	Find out FIRST and FOLLOW for the following grammar. **(April-2018_New)[LJIET]** S -> 1AB | € A -> 1AC | 0C B -> 0S C -> 1	04
35	Check whether the given grammar is LL (1) or not? **(May-2018)[LJIET]** S -> aAC | bB B -> f | g A -> Abc | Abd | e C -> h |i	07
36	How do you check whether the grammar is LL (1) or not? Generate LL(1) parsing table for given Grammar. **(Dec-2018)[LJIET]** S → iEtS | iEtSeS | a E→ b Is Grammar LL(1) or not?	07
37	Construct LL(1) parsing table for the following Grammar: **(Nov-2018_New)[LJIET]** E -> E+T | T T -> T*F | F F -> (E) | a	07
38	Construct recursive descent parser for following grammar. **(May-2019_NEW)[LJIET]** E→ T A A→ + T A A→€ T→ F B B→* F B B→€ F→ (E) F→ id	07
39	Construct LL(1) parsing table for following grammar. Check whether the grammar is LL(1) or not. **(May-2019_NEW)[LJIET]** A → A a B A → x B → B C b B → C y C → C c C → €	07
40	Explain backtracking with example. **(Nov-2019_New)[LJIET]**	04
41	Perform the Left factoring of following Grammar. S → iEtS / iEtSeS / a E → b **(Nov-2019_New)[LJIET]**	03
42	Explain Recursive Descent Parser with example. **(Nov-2019_New)[LJIET]**	07
43	Consider the following grammar to construct leftmost and right most derivation for the sentence abab. S→aSbS|bSaS|€ **(Jan-2021_NEW)[LJIET]**	03

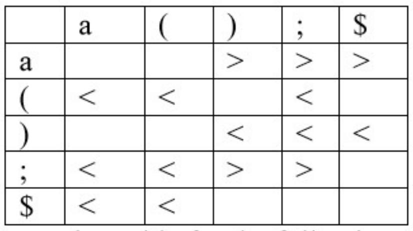
44	Write rule(s) to check grammar is left recursive or not. Remove left recursive from the following grammar. S→aBDh B→Bb|c D→EF E→g| € F→f| € **(Jan-2021_NEW)[LJIET]**	**04**
45	Construct an LL(1) parsing table for the following grammar: S→aBDh B→cC C → bC | € D→EF E→g| € F→f| € **(OCT-2020_New)[LJIET]**	**07**
46	Eliminate left recursion from following grammar. S → Aa | b A → Ac | Sd | f **(OCT-2020_New)[LJIET]**	**04**
47	Draw LL(1) parsing table for the following grammar. S → iEtSS' | a S' → eS | € E → b **(AUG-2021_Old)[LJIET]**	**07**
48	Explain rules to eliminate left recursion with example. Also explain left factoring in detail with example. **(AUG-2021_Old)[LJIET]**	**07**
49	Design predictive parsing table for following grammar. E → E+T E → T T → T*F T → F F → id **(AUG-2021_Old)[LJIET]**	**07**
50	What is recursive descent parsing? Design recursive descent parsing for the following grammar: S → aSa | aa **(AUG-2021_NEW)[LJIET]**	**04**
51	Explain the algorithm to remove left recursion in context free grammar with example. **(AUG-2021_NEW)[LJIET]**	**04**
	TOPIC:2 Bottom Up Parsing Algorithms	
	DESCRIPTIVE QUESTIONS	
1	Compute the operator precedence matrix and precedence function for the following grammar if it exists. +,*,-,/,id,num,(and) are terminal symbols.**(Nov-2011)[LJIET]** G→E E→E+T|E-T|T T→T*F|T/F/F F→num|id|(E)	**07**
2	Consider the following grammar.**(Nov-2011)[LJIET]** E → E+T | T T → TF | F F → F* |a |b 1) Construct the SLR parsing table for this grammar. 2) Construct the LALR parsing table.	**07**
3	Construct the canonical parsing table for the following Grammar.**(May-2012)(Nov-2019) [LJIET]** S'→S S→CC C→cC|d	**07**
4	Generate the SLR parsing table for the following Grammar.**(May-2012)[LJIET]**	**07**

	S→Aa|bAc|bBa A→d B→d	
5	Write unambiguous production rules for producing arithmetic expression consisting of symbols id, *, -, ( ) and ^, where ^ represents exponent. Parse following string using shift – reduce parser: id – id * id ^ id * (id ^ id) ^ id. Explain various conflicts of a shift – reduce parser.**(Dec-2012)[LJIET]**	**07**
6	Explain SLR parser in detail with the help of an example.**(Dec-2012)(May-2016)[LJIET]** Explain SLR parser. How is its parse table constructed?**(Nov-2016)[LJIET]** Explain SLR parsing method with example.**(Nov-2016_New)(May-2017)(April-2018_New)[LJIET]**	**07**
7	Explain Shift-Reduce parsing with suitable example.**(Nov-2013)(Nov-2019) [LJIET]**	**07**
8	Write down steps to set precedence relationship for Operator Precedence Grammar. Design precedence table for:**(Nov-2013)[LJIET]** E → E+ T | T T → T * F | F F → a	**07**
9	Write SLR parsing table for : S → T T → CC C → cC C → d .**(Nov-2013)[LJIET]**	**08**
10	What is bottom-up parsing? Discuss Shift Reduce parsing technique in brief. What is a handle? **.(May-2014)[LJIET]**	**08**
11	Define an Operator Precedence Grammar. Also write down the rules to find relationship between each pair of terminal symbols.**(May-2014)[LJIET]**	**08**
12	Construct SLR parsing table for the following grammar : **(May-2014)[LJIET]** E->E+T E->T T->T*F T->F F->(E) F->a	**10**
13	Show that the following grammer S-> AaAb | BbBa A -> ϵ B -> ϵ is LL(1) but not SLR(1)**. (Nov-2014)(Dec-2018)[LJIET]**	**07**
14	Show that the following grammer S->Aa | bAc | dc | bda A->d is LALR(1) but not SLR(1)**. (Nov-2014)[LJIET]**	**07**
15	Show that the following grammer:**(Nov-2014)[LJIET]** S->Aa | bAc | Bc | bBa A->d B->d is LR(1) but not LALR(1).	**07**
16	Construct the collection of sets of LR(0) items for the following grammar. S-> Aa | bAc | dc | bda A->d .**(May-2015)[LJIET]**	**07**
17	Construct an SLR Parsing table for the following grammar.**(May-2015) [LJIET]** E->E-T|T T->F↑T|F F->(E) | id	**07**
18	Check whether the following grammar is CLR or not.**(Dec-2015)[LJIET]** S → Aa | bAc | Bc | bBa A → d B → d	**07**
19	Construct SLR Parsing Table for the following grammar.**(Dec-2015)[LJIET]** S→ 0S0 | 1S1 | 10	**07**
20	Explain Operator Precedence Parsing method with example.**(Dec-2015)[LJIET]** Explain operator precedence parser by giving example for constructing a precedence graph and table.**(Dec-2012)[LJIET]** Write a short note on operator precedence parsing with an example.**(Nov-2016_New)[LJIET]** Explain operator precedence parsing method.**(May-2017)[LJIET]** Explain Operator precedence Parsing technique in detail. **(May-2018,May-2019)[LJIET]**	**07**
21	Differentiate SLR, Canonical LR and LALR. Also justify the statement "A class of grammar that can be parsed using LR methods is a proper subset of the class of grammars that can be parsed with predictive parser".**(May-2016)[LJIET]**	**07**
22	Explain LALR parser in detail. Support your answer with example.**(Dec-2015)[LJIET]**	**07**
23	Where do we use operator precedence parsing technique? Give the general precedence table for operating precedence parsing, considering all the generalized rules.**(May-2016)[LJIET]**	**07**

24	Apply shift reduce parser for parsing following string using unambiguous grammar.**(Nov-2016)[LJIET]** id - id * id – id	**07**
25	Construct a precedence graph, precedence table for operator precedence parser to be used for parsing a string consisting of id, - , * , $. Parse following string.**(Nov-2016)[LJIET]** $ id - id * id * id $	**07**
26	Check that following grammar is LALR or not.**(Nov-2016_New)(AUG-2021_Old) [LJIET]** S →L=R S →R L→*R L→ id R → L	**07**
27	Construct CLR parsing table for following grammar.**(April-2017_New)[LJIET]** S -> aSA | € A -> bS | c	**07**
28	Show that following grammar is not a SLR (1) grammar.**(April-2017_New)[LJIET]** S -> AaBa | BbBa A -> € B -> € Check the following grammar is LR(1) or not.**(May-2017)[LJIET]** S → AaAb S →BbBa A → ^ B →^	**07**
29	Define operator precedence grammar. Construct precedence matrix and precedence graph for arithmetic grammar as shown below:**(April-2017_New)[LJIET]** E -> E + T | T T -> T * F | F F -> (E) | id	**07**
30	Check given grammar is LL(1) but not SLR(1). **(Nov-2017_New)[LJIET]** S -> AaAb | BbBa A -> € B -> €	**07**
31	What is operator grammar? Generate precedence function table for following grammar. **(Nov-2017_New)[LJIET]** E -> EAE | id A -> + | *	**07**
32	Define handle and handle pruning. Explain the stack implementation of shift reduce parser with the help of example. **(Nov-2017_New)[LJIET]**	**07**
33	What is operator grammar? Check the following grammar is operator or not. Justify your answer. **(April-2018_New)[LJIET]** E -> EOE E -> id O -> * | + | -	**03**
34	Construct CLR parsing table for the following grammar. **(April-2018_New)[LJIET]** S -> CC C -> cC | d	**07**
35	Construct a SLR parsing table for following grammar. **(May-2018)[LJIET]** S -> aAb | bB A -> Aa | ϵ B -> Bb |ϵ	**07**

36	Construct the LALR parsing table for the following grammar. **(May-2018)[LJIET]** S -> CC C -> aC C -> d	**07**
37	Write a short note on operator precedence parsing for +, *, $, id. **(Dec-2018)[LJIET]**	**07**
38	Define the following terms and give suitable example for it. 1) Handle 2) Handle pruning 3) Left Factoring **(Nov-2018_New)[LJIET]**	**03**
39	Define the following terms and give suitable example for it. 1) Augmented Grammar 2) LR(0) Item 3) LR(1) Item **(Nov-2018_New)[LJIET]**	**03**
40	Construct SLR parsing table for the following grammar : S ->(L)|a L->L,S|S **(Nov-2018_New)[LJIET]**	**07**
41	Give the difference between SLR and CLR Parser. **(Nov-2018_New)[LJIET]**	**03**
42	List the different conflicts that occur in Bottom up parsing and give examples for that. **(Nov-2018_New)(OCT-2020_New) [LJIET]**	**04**
43	Construct CLR parsing table for the following grammar : **(Nov-2018_New)[LJIET]** S ->AA A->aA|b	**07**
44	Define handle and handle pruning. **(May-2019_NEW)[LJIET]**	**03**
45	Construct operator precedence relations table for following grammar. **(May-2019_NEW)[LJIET]** E→ E+E E→ E-E E→ E*E E→ (E) E→ id Assume suitable operator associativity and precedence.	**07**
46	Construct LR(0) item sets for following grammar. **(May-2019_NEW)[LJIET]** S → AaAb S → BbBa A → € B → €	**04**
47	Differentiate LR(1) and LALR(1) parsers. **(May-2019_NEW)[LJIET]**	**03**
48	Explain the following: 1. Handle 2. Forward Reference 3. Conflicts in LR Parsing **(Nov-2019_NEW)[LJIET]**	**03**
49	Generate the SLR parsing table for the following Grammar. S→Aa | bAc | bBa A→d B→d **(Nov-2019_NEW)[LJIET]**	**07**
50	Write down steps to set precedence relationship for Operator Precedence Grammar. Design precedence table for: E→E+E | E*E | E^E | id. **(Nov-2019_NEW)[LJIET]**	**07**
51	Construct the SLR parsing table for the following Grammar. E→ E+T | T T→ T*F | F F→ (E) | id **(Jan-2021_NEW)[LJIET]**	**07**

52	Write rule(s) to check grammar is operator grammar or not. **(Jan-2021_NEW)[LJIET]**	**03**
53	Construct LALR parsing table for the following grammar. S -> CC C -> cC C -> d **(Jan-2021_NEW)[LJIET]**	**07**
54	Define the following terms: 1) Handle 2) Handle pruning 3) Left Factoring **(OCT-2020_New)[LJIET]**	**03**
55	Explain shift reduce parsing technique in brief.**(OCT-2020_New)[LJIET]**	**04**
56	Construct an SLR Parsing table for the following grammar. E → E + T|T T → T * F|F F → (E) F → id **(OCT-2020_New)[LJIET]**	**07**
57	Check whether given grammar is Valid LR(0) grammar or not. E → E+T E → T T → T*F T → F F → id **(AUG-2021_Old)[LJIET]**	**07**
58	Construct the predictive parser for sentence (a, a) using the grammar: S → (L)|a L → L,S|S Check whether the given grammar is LL(1) or not? **(AUG-2021_NEW)[LJIET]**	**07**
59	Given the algorithm for computing precedence function. Consider the following OPG table matrix, compute precedence function. **(AUG-2021_NEW)[LJIET]**	**04**



	a	(	)	;	$
a			>	>	>
(	<	<		<	
)			<	<	<
;	<	<	>	>	
$	<	<			

60	Construct the SLR parsing table for the following grammar. **(AUG-2021_NEW)[LJIET]** S → AaAb S→ BbBa A→ ^ B→ ^	**07**
61	Construct the LALR parsing table for the following grammar: **(AUG-2021_NEW)[LJIET]** S → AA A → aA|b	**07**
62	List the different conflicts that occur in Bottom up parsing and give examples for that. **(AUG-2021_NEW) [LJIET]**	**03**
	TOPIC:3 Syntax-Directed Definitions(Bottom-Up Evaluation of S-Attributed Definitions, L-Attributed Definitions)	
1	Write a syntax directed definition for desk calculator. Justify whether this is an S-attributed definition or L-attributed definition. Using this definition draw annotated parse tree for 3*5+4n. **(Nov-2011)(May-2018)[LJIET]**	**07**
2	What is inherited attribute? Write syntax directed definition with inherited attributes for type declaration for list of identifiers. Show annotated parse tree for the sentence real id1,id2,id3.**(Nov-**	**07**

	2011)[LJIET] What is inherited attribute? Write syntax directed definition with inherited attributes for type declaration for list of identifiers. **(April-2018_New)[LJIET]** What is Inherited attribute? Explain with suitable example**.(Nov-2013)[LJIET]** Explain Inherited attributes with the help of an example. **(Dec-2018)[LJIET]**	
3	A robot is to be moved to a unit step in a direction specified as a command given to it. The robot moves in the direction North, South, East, West on receiving N, S, E, W command respectively & in the direction North-East , North-West, South-East, South-West on receiving A, B, C, D commands respectively. The current position of the robot is initialized to (0,0) Cartesian coordinates on receiving command Start. Write production rules for producing sequence of commands and semantic rules for knowing position of a robot after receiving a sequence of commands. Draw annotated parse tree for following sequence**:(Dec-2012)[LJIET]** Start N N A A C C N	**07**
4	Write a syntax directed definition of a simple desk calculator and draw an annotated parse tree for 4*3 + 2*5 n**.(May-2014)[LJIET]**	**06**
5	Differentiate Synthesized and Inherited attributes**.(May-2014)[LJIET]** Discuss differences between inherited attributes and synthesized attributes.**(May-2017)[LJIET]** Give the difference between synthesized attributes and inherited attributes **(Nov-2018_New)[LJIET]** Compare inherited attributes vs. synthesized attributes. **(OCT-2020_New)[LJIET]**	**04**
6	Construct a Syntax-Directed Translation scheme that translates arithmetic expressions from infix into postfix notation. Show the application of your scheme to the string “3*4+5*2”.**(Nov-2014)[LJIET]**	**07**
7	Explain with an appropriate example how to perform bottom up evaluation of an inherited attributes.**(Nov-2014)[LJIET]**	**07**
8	Discuss synthesized and inherited attributes using a suitable grammar.**(May-2015)[LJIET]** Discuss synthesized attributes and inherited attributes in details. **(Nov-2016_New)[LJIET]**	**07**
9	Show syntax directed definition for simple desk calculator. Also show annotated parse tree for 3*5+4n, where n indicates newline.**(Dec-2015)(April-2018_New)(Nov-2019_New) [LJIET]** Write a syntax directed definition of a simple desk calculator and draw an annotated parse tree for 4 * 3 + 2 n. **(Dec-2018)[LJIET]**	**07**
10	Give the translation scheme that converts infix to postfix expression for the following grammar and also generate the annotated parse tree for input string 7+3+2.**(Dec-2015)[LJIET]** E → E+T E → T T → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9	**07**
11	Write syntax directed Defination for translating following grammar for postfix notation. Also draw annotated parse tree for 9-5+2**.(May-2012)[LJIET]** expr -> expr + term expr -> expr – term term -> 0 | 1 |…..|9	**07**
12	What is attributed grammar? Which phase of the compilation process does it facilitate? Explain with example**.(May-2012)[LJIET]**	**03**

13	Write production and semantic rules for producing and analyzing statements like : **(Nov-2016)[LJIET]**
	int * ip , i , j , *ip1;
	float * fp , f;
		07

14	Explain synthesized attributes with the help of an example.**(Nov-2016)[LJIET]**	**07**
15	Write S-attributed syntax directed definition for simple desk calculator. Draw annotated parse tree for any valid input.**(Nov-2016_New)[LJIET]**	**07**
16	Develop a syntax directed definition for following grammar.**(April-2017_New)[LJIET]**	**07**

	E -> TE' E' -> +TE' | € T -> (E) T -> id	
17	Write a grammar to declare variables with data type int or float or char. Also develop a syntax directed definition for that. Draw the dependency graph for same.**(April-2017_New)[LJIET]**	07
18	Explain syntax directed translation scheme with example.**(May-2017)[LJIET]**	07
19	Give the translation scheme that converts infix to postfix notation. Generate the annotated parse tree for input string 3-5+4. **(Nov-2017_New)[LJIET]** Explain syntax directed translation scheme for Infix to Postfix conversation with example. **(Dec-2018)[LJIET]**	03, 07
20	Define syntax tree. What is s-attributed definition? Explain construction of syntax tree for the expression a-4+c using SDD. **(Nov-2017_New)[LJIET]**	07
21	Write Syntax Directed Definition to produce three address code for the expression containing the operators := , + , - (unary minus), ( ) and id. **(Nov-2017_New)[LJIET]**	04
22	Give the translation scheme that converts infix to postfix expression for the following grammar and also generate the annotated parse tree for input string "id+id*id" **(Nov-2018_New)[LJIET]** E -> E+T | T T -> T*F | F F -> id	04
23	Construct syntax directed translation scheme for infix to postfix conversion. **(May-2019_New)[LJIET]**	04
24	Explain the types of attributed grammar? Which phase of the compilation process does it facilitate? Explain with example. **(Nov-2019) (Nov-2019_New) [LJIET]**	07, 03
25	Define : 1) synthesized attribute 2) inherited attribute **(Jan-2021_NEW)[LJIET]**	03
26	Write SDD for arithmetic expression and Construct annotated parse tree for the input expression (4*7+1)*2 . **(Jan-2021_NEW)[LJIET]**	07
27	Construct syntax directed translation scheme and definition that translate arithmetic operation from infix to postfix in which an operator appears before its operand as for example xy- is a positive notation for x-y . Give annotated parse tree for the input 9-5+2 and 9-5*2. **(Jan-2021_NEW)[LJIET]**	07
28	Write SDD for desk calculator grammar. **(AUG-2021_Old)[LJIET]**	07
29	Write Translation scheme for desk calculator grammar **(AUG-2021_Old)[LJIET]**	07
30	Consider the following grammar. Write syntax directed definition. Consider "char a, b, c" as the input sentence and draw augmented parse tree. Also determine evaluation order. **(AUG-2021_New) [LJIET]** S → T List T → integer/float/double/char List → List1, id List → id	07
	TOPIC:4 Using Ambiguous Grammars , Parser Generators, Automatic Generation of Parsers	
	DESCRIPTIVE QUESTIONS	
1	Consider the grammar S -> SS+ | SS* | a Show that the string aa+a* can be generated by the grammar. Construct the parse tree for it. Is the grammar ambiguous? Justify.**(Dec-2012)(Nov-2016)[LJIET]**	04, 07

2	Write production rules for producing following language. Strings of 0's and 1's with equal numbers of 0's and 1's.**(Dec-2012)[LJIET]**	03
3	Draw transition diagrams corresponding to production rules for operators +, - , *, / and id for a predictive parser. Explain how parsing takes place for it.**(Dec-2012)[LJIET]**	07
4	Write ambiguous and unambiguous production rules for if then else construct. Illustrate parsing using both types of rules by giving an example. Also explain left factoring and its use.**(Dec-2012)[LJIET]**	07
5	What is the difference between parse tree and syntax tree? Write appropriate grammar and draw parse as well as syntax tree for a*(a-a^a).**(Nov-2013,May-2019)(Nov-2019) [LJIET]**	07
6	Explain the following:**(May-2015)[LJIET]** 1) The Handle 2) Left Factoring 3) Directed Acyclic Graph 4) Conflicts in LR Parsing 5) Parser Generator 6) Dependency Graph 7) Locality of reference	07
7	Write a context free grammar for arithmetic expressions. Develop a syntax directed definition for the grammar. Draw an annotated parse tree for the input expression: (3*2+2)*4 **(May-2015)[LJIET]**	07
8	Write short note on context free grammar (CFG) explain it using suitable example.**(May-2016)[LJIET]**	07
9	What is the difference between parse tree and syntax tree? Draw the parse tree for following expression: a= a + a * b + a * b * c – a / b + a * b and write three address code for it.**(May-2012)[LJIET]**	04
10	Write unambiguous production rules for if then else construct.**(Nov-2016)[LJIET]**	03
11	What do you mean by ambiguous grammar? Show that following is an ambiguous grammar. $E \rightarrow E+E \mid E*E \mid E-E \mid E/E \mid (E) \mid id$ **(Dec-2018)[LJIET]**	07
12	What is Ambiguous Grammar? Describe with example. **(Nov-2018_New)**	03
13	Check whether the following grammar is ambiguous or not. **(May-2019)(OCT-2020_New) [LJIET]** S→ (S) S S→ €	04
14	Distinguish between ambiguous and unambiguous grammar? **(Nov-2019_New)[LJIET]**	04
15	Explain difference between ambiguous and unambiguous grammar with example in detail. Write unambiguous grammar for desk calculator. **(AUG-2021_Old)[LJIET]**	07
16	Define the following terms: (i) Context free grammar (ii) Handle pruning (iii) Symbol table **(AUG-2021_NEW)[LJIET]**	03
Sr. No	**CHAPTER NO- 4 :Error Recovery:**	**Marks**
	Error Detection & Recovery, Ad-Hoc and Systematic Methods	
	DESCRIPTIVE QUESTIONS	
1	How can panic mode and phrase level recovery be implemented in LR parsers? Consider the expression grammar :$E \rightarrow E + E \mid E * E \mid (E) \mid id$ Prepare the SLR parsing table with error detection and recovery routines.**(Nov-2011)[LJIET]**	07
2	Explain: Error Recovery Strategies in Compiler in brief.**(May-2012)(May-2014)(Nov-2014,May-2019)(Nov-2019) [LJIET]** Explain all error recovery strategies using suitable examples.**(May-2016)[LJIET]** Discuss various error recovery strategies of compiler.**(Nov-2016_New,May-2019_NEW)[LJIET]** Write down short note on Error – Recovery Strategies. **(May-2018)[LJIET]** Explain Error Recovery Strategies in Compiler in brief. **(Nov-2019_New) [LJIET]**	07, 04

	Explain any two error-recovery strategies. **(Jan-2021_NEW)[LJIET]** Explain error recovery techniques in detail with examples. **(AUG-2021_Old)[LJIET]**	
3	Explain error recovery strategies used by parser. **(Nov-2017_New)(Nov-2018_New) [LJIET]**	**04**

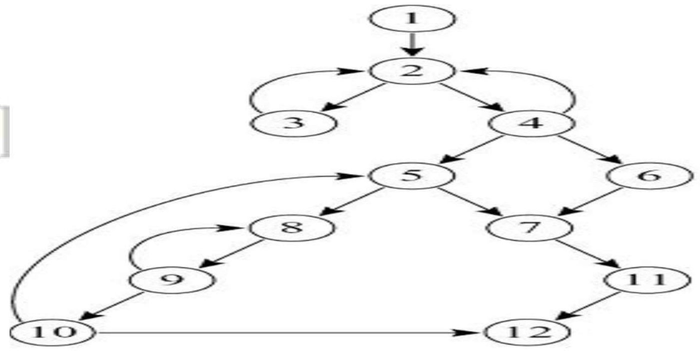
4	Find errors and identify the phase of compiler detecting them for following C program segment. Justify your answers.**(Dec-2012)[LJIET]**
	int fi(int);
	char a[10], * cptr;
	int k = 1 ; int j = 2;
	float f;
	cptr = a;
	if (k);
	fi(k);
	fi(j)
	++k;
	*(cptr + 1) = 0 ;
	++ a;
	n + *k ;
		07

5	Explain how panic mode recovery can be implemented.**(Nov-2013)[LJIET]** Explain panic mode recovery strategy. **(April-2018_New)(OCT-2020_New) [LJIET]**	**07, 03**
6	List the errors generated by the syntax analysis phase. Discuss error handling methods in the syntax analysis phase.**(May-2015)[LJIET]**	**07**
7	Explain the strategies used to recover from syntactic errors. **(Aug-2021_New) [LJIET]**	**03**
Sr. No	**CHAPTER NO- 5 :Intermediate Code Generation :**	
	Variants of Syntax Trees, Three-Address Code, Types and Declarations, Translation of Expressions, Type Checking, Syntax Directed Translation Mechanisms, Attributed Mechanisms And Attributed Definition.	
	DESCRIPTIVE QUESTIONS	
1	Translate the arithmetic expression a*-(b+c) into **(Nov-2011)(Nov-2017_New) (OCT-2020_New) [LJIET]** 1. Syntax tree 2. Postfix notation 3. Three address code.	**07,03**
2	Translate the expression –(a+b)*(c+d)+(a+b+c) into **(Nov-2011,May-2019) (Jan-2021_NEW) (OCT-2020_New) (Aug-2021_NEW) [LJIET]** 1. Quadruples 2. Triples 3. Indirect triples	**07, 04**
3	What is Intermediate form of the code? What are the advantages of it? What are generally used intermediate forms? Write N-Tuple notation for: (a+b)*(c+d)-(a+b+c).**(May-2012)(Nov-2019) [LJIET]**	**07**
4	Explain how type checking & error reporting is performed in a compiler.**(Dec-2012)(May-2016)[LJIET]**	**03**
5	Explain quadruple, triple and indirect triple with suitable example.**(Nov-2013)(Nov-2014)(April-17_New)(April-2018_New)(Nov-2018_New,May-2019_NEW)[LJIET]**	**07, 06**
6	Convert the following into quadruple, triple and indirect triple forms : -(a+b)*(c-d).**(May-2014)[LJIET]**	**06**
7	Convert the following statement into triple, indirect triple and quadruple forms. A= (B+C) $ E + (B+C) *F **(May-2015)[LJIET]**	**07**
8	What is intermediate code? What is its importance? Discuss various representations of three address code.**(May-2015)(Nov-2016_New)[LJIET]**	**07**
9	Translate following arithmetic expression:**(Dec-2015)[LJIET]**	**07**

	- (a * b) + (c + d) – (a + b + c + d) into 1] Quadruples 2] Triple 3] Indirect Triple	
10	Translate the expression – (a*b)+(c*d)+(a*b*c) into **(May-2016)[LJIET]** 1. Quadruples 2. Triples 3. Indirect triples.	**07**
11	Draw syntax tree and DAG for the statement:**(May-2016)[LJIET]** a = (a * b + c) ^ (b + c) * b + c. Write three address codes from both.	**04**
12	Draw syntax tree and DAG for following statement. Write three address codes from both. a = (a + b * c) ^ (b * c) + b * c ^ a ; **(Nov-2016)[LJIET]**	**07**
13	Construct DAG for a + a * (b- c) + (b – c) * d. also generate three address code for same. **(April-2017_New)(April-2018_New)[LJIET]**	**07, 03**
14	Write a short note on various representations of three address code.**(May-2017)[LJIET]**	**07**
15	What is importance of intermediate code? Discuss various representations of three address code using the given expression. a = b * -c + b * -c. **(Nov-2017_New)[LJIET]**	**07**
16	What is intermediate code? Explain different types of intermediate code representations. Also discuss importance of intermediate code. **(May-2018)[LJIET]**	**07**
17	What is Intermediate Code? Discuss various representations of three address code for a = (a + b * c) * (b * c) + (b + c) ^ a **(Dec-2018)[LJIET]**	**07**
18	Draw syntax tree and DAG for the statement x=(a+b)*(a+b+c)*(a+b+c+d) **(Nov-2018_New)[LJIET]**	**03**
19	Draw syntax tree and DAG for following statement. **[LJIET]** a = (a + b * c) ^ (b * c) + b * c Write three address codes from both**.**	**04**
20	Define Intermediate code and its importance. **(May-2019_NEW)[LJIET]**	**03**
21	Construct syntax tree and DAG for following expression. **(May-2019_NEW)[LJIET]** a = (b+c+d) * (b+c-d) + a	**04**
22	Construct a DAG for (a+b)* (a+b+c). **(Nov-2019_New)[LJIET]**	**03**
23	Translate following arithmetic expression (a * b) + (c + d) - (a + b) into 1] Quadruples 2] Triple 3] Indirect Triple **(Nov-2019_New)[LJIET]**	**07**
24	What is Intermediate form of the code? What are the advantages of it? **(Jan-2021_NEW)[LJIET]**	**04**
25	Write three address code for a := a + a * b + a * b * c – a/b + a * b **(OCT-2020_New)[LJIET]**	**04**
26	Draw a DAG for expression: a + a * (b – c) + (b – c) * d. **(OCT-2020_New)[LJIET]**	**03**
27	Explain following in detail. 1) Quadruple 2) Triple 3) Indirect Triple **(AUG-2021_Old)[LJIET]**	**07**
28	Write a three address code for the following expression: a < b or c < d **(AUG-2021_NEW)[LJIET]**	**04**
29	Give the translation scheme for converting the assignment statements into three address code. Assume suitable example for the same. **(AUG-2021_NEW)[LJIET]**	**07**
Sr. No	**CHAPTER NO- 6 :Run Time Memory Management :**	
	Source Language Issues, Storage Organization. Stack Allocation of Space, Access to Nonlocal Data on the Stack, Heap Management	
	DESCRIPTIVE QUESTIONS	
1	Explain activation tree and control stack.**(Nov-2011)[LJIET]**	**04**
2	What are the limitations of static storage allocation? Explain the problem of dangling references.**(Nov-2011)[LJIET]**	**03**

3	For what purpose compiler uses symbol table? How characters of a name are stored in symbol table? **(Nov-2011)[LJIET]**	**04**
4	Explain the static scope rule and dynamic scope rule**.(Nov-2011)[LJIET]**	**03**
5	Explain the structure of an activation record with all its components**.(May-2012)[LJIET]**	**04**
6	Explain: Symbol Table Management. How symbol table differs from other data structures? **(May-2012)(Nov-2019) [LJIET]**	**03,07**
7	Compare: Static v/s Dynamic Memory Allocation.**(May-2012)(Nov-2013)(April-2018_New)(Jan-2021_NEW)[LJIET]**	**04, 03**
8	What is a symbol table? Discuss any two data structures suitable for it & compare their merits /demerits. Also compare one pass & two pass compilers**.(Dec-2012)[LJIET]**	**07**
9	Explain activation record. How is task divided between calling & called program for stack updating? **.(Dec-2012)(Nov-2016)(Dec-2018)(Nov-2019) [LJIET]**	**07**
10	Explain various parameter passing methods**.(Dec-2012)(Nov-2016)(Dec-2018)(Nov-2019_New) [LJIET]** Explain parameter passing techniques for procedure. **(Nov-2017_New)(Nov-2018_New) (AUG-2021_New) (OCT-2020_New) [LJIET]** Explain any two methods of parameter passing. **(April-2018_New,May-2019_NEW) (Jan-2021_NEW) [LJIET]**	**03, 3.5, 04**
11	Explain the following parameter passing methods**.(Nov-2016_New)[LJIET]** 1. Call-by-value 2. Call-by-reference 3. Copy-Restore 4. Call-by-Name	**07**
12	Explain heap, dynamic storage allocation techniques and synthesized attributes**.(Dec-2012)[LJIET]**	**07**
13	Explain Stack Allocation and Activation Record Organization in brief**.(Nov-2013)(Nov-2014)(Nov-2019_New) [LJIET]** Write a note on stack allocation strategy. **(Nov-2018_New)[LJIET]**	**06, 07, 03**
14	What is the use of a symbol table? How the identifiers are stored in the symbol table? **.(May-2014)(Nov-2014)(OCT-2020_New) [LJIET]**	**07**
15	Write a note on static and dynamic memory allocation. What do you mean by dangling reference? **(May-2014)[LJIET]**	**07**
16	What is an activation record? Explain how they are used to access various local and global variables **.(May-2014)(May-2016))(Nov-2018_New,May-2019)[LJIET]**	**07**
17	Elaborate the term "Activation Record" in detail**.(May-2015)[LJIET]**	**07**
18	Write a short note on Symbol Table Management.**(Dec-2015)(April-2017_New)(Jan-2021_NEW) [LJIET]**	**07,04**
19	Explain various dynamic storage allocation techniques.**(Dec-2015)[LJIET]** Explain Dynamic storage allocation technique**.(Nov-2016_New)[LJIET]** Explain dynamic memory allocation strategy. **(Nov-2018_New)[LJIET]**	**07, 04**
20	What is a symbol table? Discuss the most suitable data structure for it by stating merits / demerits**.(Nov-2016,May-2019)[LJIET]**	**04**
21	Explain Activation record and Activation tree in brief.**(April-2017_New)[LJIET]**	**07**
22	Explain symbol table with two data structures suitable for it**.(May-2017)[LJIET]**	**07**
23	Explain static storage allocation technique**.(May-2017,May-2019_NEW)[LJIET]**	**07,03**
24	Explain activation record organization in brief**.(May-2017)[LJIET]**	**07**
25	What is activation record? Explain stack allocation of activation records using example. **(Nov-2017_New)[LJIET]**	**07**
26	What is activation tree? **(Nov-2017_New)[LJIET]**	**03**

27	What is symbol table? For what purpose , compiler uses symbol table? **(April-2018_New)[LJIET]**	03
28	Write a short note on activation record. **(April-2018_New)[LJIET]**	04
29	Write difference(s) between stack and heap memory allocation. **(April-2018_New)[LJIET]**	03
30	Discuss various Storage allocation strategies in detail. **(May-2018)[LJIET]**	07
31	Explain various data structures used in symbol table management. **(May-2018)[LJIET]**	07
32	Explain activation tree, control stack, the Scope of Declaration and Bindings of Names. **(May-2018)[LJIET]**	07
33	Explain activation record. **(Aug-2021_New)[(May-2019_NEW)[LJIET]**	03, 04
34	Compare Static and Dynamic memory allocation. **(Nov-2019)[LJIET]**	07
35	Differentiate: static v/s dynamic memory allocations. **(Nov-2019_New)[LJIET]**	03
36	Discuss symbol table management in detail. **(Nov-2019_New)[LJIET]**	04
37	What is activation record? **(Jan-2021_NEW)[LJIET]**	03
38	What are the limitations of static storage allocation? **(Jan-2021_NEW)[LJIET]**	03
39	What is a symbol table? Discuss any two data structures suitable for it. **(OCT-2020_New)[LJIET]**	03
40	Explain Control Stack. **(OCT-2020_New)[LJIET]**	03
41	What do you mean by dangling references? **(OCT-2020_New)[LJIET]**	04
42	Explain various storage allocation techniques in detail. **(AUG-2021_Old)[LJIET]**	07
43	Discuss the various storage allocation strategies for compilers in detail. **(AUG-2021_NEW)[LJIET]**	07
Sr. No	**CHAPTER NO- 7: Code Generation and Optimization :**	**Marks**
	Issues in the Design of a Code Generator, The Target Language, Addresses in the Target Code, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, A Simple Code Generator, Machine dependent optimization, Machine independent optimization Error detection of recovery	
	DESCRIPTIVE QUESTIONS	
1	Write the generic issues in the design of code generators.**(Nov-2011)(Nov-2013)(OCT-2020_New) [LJIET]** Describe code generator design issues.**(Dec-2012)(Nov-2014)[LJIET]** Explain various issues in design of code generator.**(Dec-2015)(April-2018_New,May-2019_NEW)(Nov-2019_New) [LJIET]** Discuss generic issues in the design of code generator. **(Nov-2016_New)[LJIET]** Discuss the issues in the design of code generation.**(April-2017_New,May-2019)[LJIET]** Explain code generator design issues.**(May-2017)[LJIET]** Describe code generator design issues. **(Nov-2017_New)[LJIET]** Discuss Design Issues of Code Generator. **(Dec-2018)[LJIET]** List the different issues in code generation phase and describe any two issues. **(Nov-2018_New)[LJIET] or** Discuss generic issues in the design of code generation. **(Nov-2019) [LJIET]** Explain any two issues in design of code generator. **(Jan-2021_NEW)[LJIET]**	07,03
2	Explain peephole optimization.**(Nov-2011)(Nov-2013)(May-2014)(May-2016)(Nov-2016)(Nov-2017_New)(May-2018,May-2019)[LJIET]** Explain Peephole Optimization method.**(Nov-2016_New,May-2019_NEW)[LJIET]** Write a note on peephole optimization.**(April-2017_New)[LJIET]** Explain peephole code optimization in detail with examples. **(AUG-2021_Old)[LJIET]**	07, 08
3	Explain Peephole Optimization method.**(Nov-2017_New)[LJIET]**	04
4	Discuss the factors affecting the target code generation.**(May-2012)[LJIET]**	03
5	Write down the algorithm for partitioning of basic blocks.**(Nov-2013)[LJIET]**	07
6	Define: DAG. Explain DAG representation of basic block with example.**(May-2017,May-2019)[LJIET]**	07

7	Define a following: Basic block, Constant folding, Natural loop, Handle. **(April-2017_New)[LJIET]**	**07**
8	Define dominators. Construct dominator tree for following graph.**(April-2017_New)[LJIET]** 	**07**
9	Define following : DAG, Basic Blocks, Flow graph **(Nov-2017_New)[LJIET]**	**03**
10	Explain algebraic simplifications and flow of control optimization characteristics of peephole optimization. **(April-2018_New)[LJIET]**	**03**
11	i. Construct the DAG for the following basic block: **(May-2018)[LJIET]** d: = b * c e: = a + b b: = b * c a: = e – d ii. Describe issues in code generation process.	**07**
12	Define basic block with a simple example. **(May-2019_NEW)[LJIET]**	**03**
13	Explain Code Generation algorithm in detail. **(AUG-2021_Old)[LJIET]**	**07**
14	What is DAG? Construct DAG for following expression: a + a * (b-c)+ (b–c)* d. **(AUG-2021_NEW) [LJIET]**	**03**
15	Write Short notes on **(AUG-2021_NEW) [LJIET]** 1. Local and loop optimization 2. induction variable elimination	**07**
16	What is flow graph? Give suitable example. **(AUG-2021_NEW) [LJIET]**	**03**
17	Discuss briefly about the Peephole optimization. **(AUG-2021_NEW) [LJIET]**	**04**
18	Explain the issues regarding code generation in compiler design with example. **(AUG-2021_NEW) [LJIET]**	**07**
Sr. No	**CHAPTER NO- 8 : Instruction-Level Parallelism:**	**Marks**
	Processor Architectures, Code-Scheduling Constraints, Basic-Block Scheduling, Pass structure of assembler	
1	What is Instruction Pipelines and Branch Delays? Explain in detail. **[LJIET]**	**07**
2	What is Data Dependence? What are the types of data dependence? **[LJIET]**	**07**
3	Explain control dependence. **[LJIET]**	**07**
4	Explain Prioritized Topological Orders. **[LJIET]**	**07**
5	Explain Pass structure of assembler. **[LJIET]**	**07**