

	Unit			
Sr.	No.	Question	BL	CO
1.	1	Discuss types of translators.	U	CO1
2.	1	Compare compiler v/s interpreter v/s assembler		
3.	1	Discuss analysis synthesis phase of compiler.		CO1
4.	1	Explain Phases of compiler.		
5.	1	Describe output of all the phases of compiler for following expression: 1. Position=initial+rate*60 2. A=A+B*C*2 3. F=(9/5)*C+32 4. I=(p*n*r)/100	U	CO1
6.	1	Compare front end and back end	R	CO1
7.	1	Explain cousins of compiler.	U	CO1
8.	1	Discuss pass structure of compiler.	U	CO1
9.	1	Write the effect of reducing the number of passes.	R	CO1
10.	1	Explain types of Compilers.	U	CO1
11.	2	Discuss interaction of scanner and parser.	U	CO2
12.	2	Describe token, pattern, and lexemes.	U	CO2
13.	2	Discuss input buffering methods.	U	CO2
14.	2	Design regular expression for the given language.	A	CO2
15.	2	Discuss types of finite automata.	U	CO2
16.	2	Design NFA from regular expression for the given language. 1. abba 2. bb(a)* 3. (a b)* 4. a* b* 5. a(a)*ab 6. aa*+ bb* 7. (a+b)*abb 8. 10(0+1)*1 9. (a+b)*a(a+b) 10. (0+1)*010(0+1)* 11. (010+00)*(10)* 12. 100(1)*00(0+1)*	A	CO2
17.	2	Design NFA for given regular expression using Thompson's notation and then convert it into DFA. (using Subset construction) OR Explain subset construction method for constructing DFA from an NFA with an example. 1. (a+b)*a(a+b) 2. (a+b)*ab*a 3. (a b)* ab 4. (a b)*abb 5. (0 1)* 6. (0* 1*)* 7. (0 1)*011 8. a(a+b)*ab 9. b(a+b)*abb	A	CO2
18.	2	Perform optimization on given DFA.	A	CO2
19.	2	Design DFA without constructing NFA from given regular expression. (Tree based	A	CO2



method 1. (a b)*abb# 2. (a b)*abb# 2. (a b)*abb# 3. a*(b*)*c* c*)(a c)*# 4. (a*b)*ab*a# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 7. 0*2 7. 0*3 7. 0*2 7. 0*3							T
2. (a b)*a# 3. a*(b* c*)(a c)*# 4. (a+b)*ab*a# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 5. 0*1*(0/1)# 7. 0 7							
3. a*(b* c*)(a c)*# 4. (a+b)*ab*a# 5. 0*t*(9/1)# 20. 2 Discuss LEX Tool. U CO2 21. 3 Discuss role of parser. U CO3 22. 3 Explain context free grammar with example. R CO3 23. 3 Explain leftmost and rightmost derivation with example. Co7 Perform leftmost or rightmost derivation for given string and grammar. 1. S>absb b8sa e (string: abab) 23. 3 2. S>A1B							
4. (a+b)*ab*a# 5. o*1*(0/1)# 20. 2 Discuss LEX Tool. U CO2 21. 3 Discuss role of parser. U CO3 22. 3 Explain context free grammar with example. R CO3 23. 3 Explain leftmost and rightmost derivation with example. Or Perform leftmost or rightmost derivation for given string and grammar. 1. S→abs bas ε (string: abab) 2. S→A1B							
20. 2 Discuss LEX Tool. U CO2							
20. 2 Discuss LEX Tool. U CO2 21. 3 Discuss role of parser. U CO3 22. 3 Explain context free grammar with example. R CO3 23. Explain leftmost and rightmost derivation with example. Or R CO3 23. 3 Explain leftmost or rightmost derivation for given string and grammar. 1. S→aSbS bSaS ε (string: abab) U CO3 23. 3 Explain leftmost or rightmost derivation for given string and grammar. U CO3 24. 3 Explain leftmost or rightmost derivation for given string and grammar. U CO3 25. 3 Dis I E E Id E Id E Id E Id E Id Id							
21. 3 Discuss role of parser. U CO3 22. 3 Explain context free grammar with example. R CO3 22. Sexplain leftmost and rightmost derivation with example. Perform leftmost or rightmost derivation for given string and grammar. U CO3 23. 3 2. S→A1B. A→OA ε B→OB IB ε (String: 1001) U CO3 24. 3. E→E+E E ^{TE} Id (E) -E (String: id+id*id) A + S→ AS Sa ε (string: aa+a*) U CO3 25. S→SS+ SS* a (string: aa+a*) Discuss Ambiguity with example. S→ABA A→AA ε A→AA Λ ε A→AA ε A			, ,				
22. 3 Explain context free grammar with example. R CO3 23. 2 Explain leftmost and rightmost derivation with example. U CO3 23. 3 2. S⇒ASIB SaSI ε (string: abab) U CO3 23. 3 2. S⇒AIB S⇒OSI SI ε (String: 1001) A⇒OA ε B⇒OB I (E) - E (String: abaa') U CO3 24. 3 Discuss Ambiguity with example. S⇒ASS SSI ε (string: aava'') S⇒ABA A⇒AA ε A⇒AA Λ A⇒AA Λ A⇒AA ε E E E E E E E E E	-						
Explain leftmost and rightmost derivation with example. Or Perform leftmost or rightmost derivation for given string and grammar. 1. $S o aShS bSaS \epsilon (string: abab)$ 2. $S o A1B$ $A o 0A \epsilon$ $B o 0B 1B \epsilon (String: 1001)$ 3. $E o E o E E o E id (E) - E (String: id + id + id)$ 4. $S o aS Sa \epsilon (string: aa + a + a)$ Discuss Ambiguity with example. S o a Sa bSS SSb SbS S o AABA A o AAA \epsilon A \otimes AABA \otimes \	-						
Or Perform leftmost or rightmost derivation for given string and grammar. 1. S→aSbS bSaS ε (string: abab) 2. S→A1B A→0A ε B→0B 1B ε (String: 1001) 3. E→E+E E*E id (E) -E (String: id+id*id) 4. S→aS Sa ε (string: aaaa*)	22.	3	1 0			R	CO3
Discuss Ambiguity with example. $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	23.	3	Or Perform leftmost or rightmost derivation for given string and grammar. 1. $S \rightarrow aSbS \mid bSaS \mid \varepsilon \text{ (string: abab)}$ 2. $S \rightarrow A1B$ $A \rightarrow 0A \mid \varepsilon$ $B \rightarrow 0B \mid 1B \mid \varepsilon \text{ (String: 1001)}$ 3. $E \rightarrow E + E \mid E \neq E \mid id \mid (E) \mid -E \text{ (String: id+id*id)}$ 4. $S \rightarrow aS \mid Sa \mid \varepsilon \text{ (string: aaaa)}$			U	CO3
24. 3 $ \begin{array}{ c c c c c }\hline S \rightarrow a Sa bSS SSb SbS & S \rightarrow ABA, & S \rightarrow ABA \\ A \rightarrow aA \mid \epsilon & A \rightarrow aA \mid \Lambda \\ B \rightarrow bb \mid \epsilon & B \rightarrow bB \mid \Lambda \\ \hline S \rightarrow A \mid B & S \rightarrow S + S \mid S * S \mid a \mid b \\ A \rightarrow aAb \mid aabb & Write the unambiguous CFG based on precedence rules for the above grammar. Derive the parse tree for expression (a + a)*b \\ \hline 25. & 3 & Explain types of parsers. & U & CO3 \\ \hline 26. & 3 & Discuss backtracking with suitable example. & U & CO3 \\ \hline 26. & 3 & Discuss backtracking with suitable example. & U & CO3 \\ \hline 27. & A \rightarrow Abd \mid Aa \mid a & 1. S \rightarrow iEtS \mid iEtSeS \mid a \\ B \rightarrow Be \mid b & 2. A \rightarrow ad \mid a \mid ab \mid abc \mid x \\ 2. & A \rightarrow AB \mid AC \mid a \mid b \\ 3. & S \rightarrow A \mid B & A \rightarrow Ad \mid Ae \mid aB \mid aC \\ A \rightarrow ABC \mid Acd \mid a \mid aa & B \rightarrow bBC \mid f \\ C \rightarrow g & A \rightarrow ABC \mid Acd \mid a \mid ab \mid abc \mid x \\ A \rightarrow ABC \mid Acd \mid a \mid ab \mid abc \mid $				-			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$					S → ABA		
24. 3				•			
24. 3 $S \rightarrow A \mid B$ $A \rightarrow aAb \mid aabb$ Write the unambiguous CFG based on precedence rules for the above grammar. Derive the parse tree for expression $(a + a)*b$ U CO3 25. 3 Explain types of parsers. U CO3 26. 3 Discuss backtracking with suitable example. U CO3 Perform Left recursion and left factoring on given grammar. Left recursion Left factoring 1. $A \rightarrow Abd \mid Aa \mid a$ 1. $S \rightarrow \text{IETS} \mid \text{IETSES} \mid a$ $B \rightarrow \text{Be} \mid b$ 2. $A \rightarrow \text{Ad} \mid a \mid ab \mid abc \mid x$ 2. $A \rightarrow AB \mid AC \mid a \mid b$ 3. $S \rightarrow A$ 3. $S \rightarrow A \mid B$ 4. $A \rightarrow ABC \mid Ac \mid a \mid a$ B $A \rightarrow BC \mid Ac \mid a \mid a$ B $A \rightarrow BC \mid Ac \mid a \mid a$ B $A \rightarrow BC \mid Ac \mid a \mid a$ B $A \rightarrow BC \mid Ac \mid a \mid a$ B $A \rightarrow BC \mid Ac \mid a \mid a$ B $A \rightarrow BC \mid Ac \mid a \mid a$ B $A \rightarrow BC \mid Ac \mid a \mid a$ B $A \rightarrow BC \mid a$ CO3 28. 3 Explain Recursive Decent Parsing method. A CO3 Design LL(1) Parsing table for the given grammar. 1. $S \rightarrow aBa$ B $\rightarrow bB \mid \epsilon$ A CO3				•	-		
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	24.	3		•	•	Α	CO3
$ \begin{array}{ c c c c c } & & & & & & & & & & & & & & & & & & &$			•	•	• •		
Derive the parse tree for expression (a + a)*b			- - - - - - - - -				
26. 3 Discuss backtracking with suitable example. U C03 Perform Left recursion and left factoring on given grammar. Left recursion			•				
Perform Left recursion and left factoring on given grammar. Left recursion 1. A \rightarrow Abd Aa a B \rightarrow Be b 2. A \rightarrow ad a ab abc x 2. A \rightarrow AB AC a b 3. S \rightarrow A 3. S \rightarrow A B A \rightarrow ABC Acd a aa B \rightarrow Bee b C \rightarrow g 4. Exp \rightarrow Exp+term Exp-term term 5. S \rightarrow (L) a L \rightarrow L, S S 28. 3 Explain Recursive Decent Parsing method. Design LL(1) Parsing table for the given grammar. 1. S \rightarrow aBa B \rightarrow bB ϵ A C03	25.	3	Explain types of parsers.			U	CO3
Left recursion 1. $A \rightarrow Abd \mid Aa \mid a$	26.	3	Discuss backtracking with suitable example.		·.	U	CO3
28. 3 Explain Recursive Decent Parsing method. A CO3 Design LL(1) Parsing table for the given grammar. 1. $S \rightarrow aBa$ $B \rightarrow bB \mid \epsilon$ A CO3	27.	3	Left recursion 1. A→Abd Aa a	ū	Left factoring 1. S→iEtS iEtSeS a 2. A→ ad a ab abc x 3. S→A A→Ad Ae aB aC B→bBC f	A	СО3
Design LL(1) Parsing table for the given grammar. 1. $S \rightarrow aBa$ $B \rightarrow bB \mid \epsilon$ A CO3	28.	3				A	CO3
29. 3 1. $S \rightarrow aBa$ $B \rightarrow bB \mid \epsilon$ A CO3	20.		-		mmar.	1 • • •	300
$\begin{array}{ c c c c c c c c c c c c c c c c c c c$		•		U			con
	29.	3				A	CO3
			•	mmar is LL((1) or not.		



		S→aB € B→bC € C→cS € 3. Develop an LL(1) parser table for the following grammar and Parse the string using the parsing table: (id*id) + (id*id) E→TA A→+TA € T→VB B→*VB € V→id (E) 4. Develop an LL(1) parser table for following: S → iCtSeS iCtS a C→b 5. E→BA A→&BA € B→true false 6. S → AaAb BbBa A→ € B→€ 7. S→ aAB bA € A→ aAb € B→bB € 8. S→iCtSA a A→eS € C→b 9. S→ (L) a L→ L,S S 10. S→1AB € A→1AC 0C B→0S		
30.	3	C→1 Define: Handle pruning and Handle.	R	CO3
30.	J	Apply shift reduce parser for parsing given string using unambiguous grammar:	IV.	LUS
31.	3	 id-id*id-id id+id*id id*id+id+id (id+id)*id id-id-id+id 	A	соз
32.	3	Check the following grammar is operator grammar or not. Justify your answer. Also prepare precedence matrix, and graph. 1. E→E+T T T→T*F F F→(E) id 2. E→EAE id A→+ * 3. E→EOE E→id O→* + -	A	CO3



			Т	ı
		Design SLR parsing table for given grammar.		
33.		1. S→AaBa BbBa		
		B→€		
		A→€		
	3	2. E→ E+T T	Α	CO3
	J	T→ TF F		
		F→ F* a b		
		3. E→ E+T T		
		$T \rightarrow T^*F \mid F$		
		$F \rightarrow (E) \mid id$		
		Design CLR parsing table for given grammar.		
0.4		1. S→CC		con
34.	3	C→cC d	AC	CO3
		2. S→aSA €		
		A→bS c		
		Design LALR parsing table for given grammar. 1. S→L=R		
		S→R L→*R		
		L →id		
35.	3	R→L	A	CO3
		2. S→Aa bAc dc bda		
		A→d		
		3. S→CC		
		C→ cC d		
36.	3	Discuss YACC parser generator.		CO3
37.	3	Explain Syntax directed definitions.		CO3
38.	3	Compare synthesized attribute v/s inherited attribute		CO3
		Design annotated parse tree for a given string.		
		1. 3*5+4n;		
39.	3	2. (3+4)*(5+6)n;	A	CO3
		3. 1*2*(3+4)n;		
		Design Syntax directed definition to translates arithmetic expressions from infix	_	604
40.	4	to prefix notation.	A	CO4
41.	4	Explain L-Attributed definition.		CO4
42.	4	Explain syntax directed translation scheme with example.		CO4
43.	4	Explain quadruple, triple, and indirect triple with example.		CO4
		Discuss various representations of three address code for a given example.		
		1. $(a = b * -c + b * -c)$		
		2. a*-(b+c)		
44.	4	3. a+a*(b-c)*d	U	CO4
		4. ans=a+b*c/2.0		
		5. $x = -a * b + -a * b$		
		6(a*b)+(c+d)-(a+b+c+d)		
45.	4	Express following statement in form of DAG.	U	CO4
1 -1		1. a=b*-c+b*-c		



Darshan Institute of Engineering & Technology BTech-SEM - VII | Question Bank 2101CS701 - Compiler Design | 2025

		2 * (1) . (1) * 1		
		2. $a + a * (b - c) + (b - c) * d$		
46.	4	Discuss source language issues.	U	CO4
47.	4	Explain activation record and activation tree.	U	CO4
48.	4	Explain storage allocation strategies in brief.	U	CO4
49.	5	What is code optimization? Discuss its types.	R	CO5
50.	5	Explain Machine independent code optimization techniques.	U	CO5
51.	5	Explain Machine dependent code optimization techniques.	U	CO5
52.	5	Explain peephole optimization.	U	CO5
53.	5	Discuss loop optimization.	U	CO5
54.	5	Discuss issues in code generation.	U	CO5
55.	5	Calculate instruction cost of given assembly code.	A	CO5
56.	5	What is basic block? Write algorithm to partition a basic block.	U	CO5
57.	5	Discuss transformation on basic block.	U	CO5
58.	5	Explain flowgraph with example.	U	CO5
59.	5	Describe code generation algorithm.	U	CO5
60.	5	Design target code for given assignment statement using code generation algorithm.	U	CO5