CBCS SCHEME - Make-Up Exam

USN | | | | | | | | | | |

BCS/BIS613C

Sixth Semester B.E/B.Tech. Degree Examination, June/July 2025
Compiler Design

Time: 3 hrs. Max. Marks:100

Note: *1. Answer any FIVE full questions, choosing ONE full question from each module.*
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
1	a.	Define Compiler. Outline the different phases of a compiler with a neat diagram and demonstrate the output of each phase for the expression. Result = opd1 + opd2 * 25. (Assume all variable are of type float).	10	L3	CO1
	b.	Explain the applications of Compiler Technology.	6	L2	CO1
	c.	Distinguish between type checking and bound checking, with suitable example.	4	L3	CO1
		OR			
2	a.	With neat diagram, explain he model of a compiler front end.	8	L3	CO1
	b.	Consider the context free grammar S→ SS + |SS*| a i) Construct the derivation of the string aa + a* ii) Develop the parse tree for the above string.	6	L3	CO1
	c.	Write the syntax – directed definition for infix to postfix translation of arithmetic expressions and construct annotated parse tree for 9 – 5 +2.	6	L3	CO1
		Module – 2			
3	a.	Examine the reasons for separating analysis phase into lexical and syntax.	6	L4	CO2
	b.	With neat diagram, explain interactions between lexical analyzer and the parser.	6	L2	CO2
	c.	Discuss input buffering mechanism with an algorithm for lookahead code with sentinels.	8	L2	CO2
		OR			
4	a.	Construct the transition diagrams to recognize the tokens given below. i) relop ii) Identifier iii) unsigned numbers	6	L3	CO2
	b.	Consider the context free grammar : S→ + SS |*SS| a and the string w = + * aaa i) Give the left most and right most derivation and parse tree for the string. ii) Is the grammar ambiguous or unambiguous? Justify your answer.	6	L3	CO3
	c.	Demonstrate different error – recovery strategies in syntax analysis with suitable example.	8	L3	CO3

		Module – 3			
5	a.	Define recursive – descent parsing, explain with algorithm. Consider the grammar. S→ cAd A→ ab / a Construct a parse tree top – down for the input string w = cad. Demonstrate how backtracking can be used for tracing.	**10**	**L3**	**CO3**
	b.	With a neat diagram, outline the model of a table driven predictive parser. Explain the predictive parsing algorithm.	**10**	**L2**	**CO3**
		OR			
6	a.	Consider the following grammar S→ TS |[s]s|) s|∈ T→ (x) X→ TX |[x] x| ∈ i) Construct FIRST and FOLLOW sets ii) Construct its LL (1) parsing table iii) Check the grammar LL(1) or not	**10**	**L3**	**CO3**
	b.	Illustrate possible actions a shift – reduce parser can make and demonstrate conflicts daring Shift – Reduce Parsing.	**10**	**L3**	**CO3**
		Module – 4			
7	a.	With neat diagram, explain model of an LR parser and explain LR parsing algorithm.	**10**	**L2**	**CO3**
	b.	Consider the following grammar E→ E + T | T T→ T * F | F F → (E) | id i) Construct LR (o) automation and SLR Parsing Table. ii) Demonstrate the moves of an LR parser on id * id + id	**10**	**L3**	**CO3**
		OR			
8	a.	With algorithm, explain constructing SLR – parsing table.	**10**	**L2**	**CO3**
	b.	Define Syntax Directed Definition (SDD). Write the grammar and SDD for simple desk. Calculator and construct annotated parse tree for (3 + 5) * 4 n.	**10**	**L3**	**CO4**
		Module – 5			
9	a.	Outline how DAG helps in intermediate code generation. Construct the DAG for the following. i) a + b + (a + b) ii) a + b + a + b	**6**	**L3**	**CO5**
	b.	Define three – address code. Construct the quadruple, triple and indirect triple representation of the arithmetic expression. a = b * - c + b * - c	**10**	**L3**	**CO5**
	c.	Define type expressions. Construct the type expression for int [2] [3] of array type.	**4**	**L3**	**CO5**
		OR			
10	a.	Explain syntax – directed definition for flow –of- control statements.	**10**	**L3**	**CO5**
	b.	Discuss the issues in the design of a code generator.	**10**	**L2**	**CO5**

* * * * *