Sub Code: R2331463 **R23** **Set No. 1**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY-GURUJADA VIZINAGARAM
III B. Tech I Semester Regular Examinations November -2025
AUTOMATA THEORY & COMPILER DESIGN
(COMPUTER SCIENCE & ENGINEERING (AI&DS)**)**

Time: 3 hours **Max. Marks: 70**

The Question paper consists of Part A & Part B.
Part A is compulsory, Answer all questions.
Part B Answers any one question from each unit.
*****

1		PART-A	(20Marks)
	a)	Write the regular expression for all strings over {0,1} begin with 00 and ending with 01.	[2]
	b)	Consider the following grammar E -> Ea | Eb | c, re-write the grammar by eliminating left-recursion	[2]
	c)	Define syntax-directed translation (SDT).	[2]
	d)	Define shift-reduce conflict and reduce-reduce conflict.	[2]
	e)	Define a context-sensitive language with one example grammar.	[2]
	f)	What is type checking in compiler design?	[2]
	g)	Define access link and its purpose in nested subprograms.	[2]

	h)	void update(int a, int *b) {
		a = a + 5;
		*b = *b + a;
		}
		If variables x = 15 and y = 20, compute the final values of x and y after calling update(x, &y)
			[2]

	i)	What is object code? List any two common forms of object code output.	[2]
	j)	What are address descriptors and how are they used in register allocation?	[2]
		PART-B	(50Marks)
		Question from **Unit - I**	
2	a)	Construct a DFA that accepts all binary strings divisible by 3.	[5]
	b)	Construct a Non-Deterministic FA which accepts set of all strings over ∑={ 0, 1} such that fourth symbol from right is 0 and second symbol from right is 1	[5]
		(OR)	
3		Consider the following grammar with productions E-> E * id | E – id | id i) Eliminate left recursion from the grammar ii) Consider the result of (i), Find FIRST and FOLLOW of all the non-terminals iii) Construct LL(1) parsing table iv) Show the parsing of string *id * id – id -id* by using LL(1) parser	[2+3+3+2]
		Question from **Unit - II**	
4	a)	Explain the differences among LR(0), SLR(1), canonical LR(1), and LALR(1) parsers, emphasizing trade-offs in table size and power.	[5]
	b)	E → E + T { E.val = E1.val + T.val } E → T { E.val = T.val } T → num { T.val = num.lexval } Given grammar and semantic rules, Construct the annotated parse tree for input 2 + 3 + 5	[5]

		(OR)	
5		S → CC C → cC | d Construct an SLR parsing table for the grammar:	[10]
		Question from **Unit - III**	
6	a)	Explain the Chomsky hierarchy of grammars, including examples and computational power of each type.	[5]
	b)	Explain with examples how a C compiler performs automatic type conversion during arithmetic expression evaluation.	[5]
		(OR)	
7	a)	Explain the role of type checking in compiler design and how it ensures program correctness.	[5]
	b)	Explain how function overloading is resolved during compilation, with an example showing type matching.	[5]
		Question from **Unit - IV**	
8	a)	Compare static, stack-based, and heap-based storage allocation strategies with examples.	[5]
	b)	Explain the peephole optimization techniques such as redundant instruction elimination and strength reduction.	[5]
		(OR)	
9		0. input(a,e) 1. L1: a = a + 1 2. d = e -a 3. if d > 0 goto L1 4. f = a + d 5. g = f * 2 (a) Identify leaders and explain your choice. (b) Divide the code into basic blocks. (c) Construct the flow graph for the program	[4+4+2]
		Question from **Unit - V**	
10	a)	t1 = a + b t2 = t1 * c t3 = a + b t4 = t3 * d construct a DAG representation, identify common subexpressions, and show the optimized code after eliminating redundancy.	[5]
	b)	Explain the difference between register allocation and register assignment.	[5]
		(OR)	
11	a)	Explain the purpose of the code generation phase in a compiler.	[4]
	b)	t1 = a * b t2 = t1 + c t3 = t2 – d Generate equivalent assembly instructions assuming a simple three-address machine with registers R1–R3.	[6]

*****