Reg. No.															

Question Paper Code	13450

B.E. / B.Tech. - DEGREE EXAMINATIONS, APRIL / MAY 2025

Sixth Semester

Computer Science and Engineering

(Common to Computer Science and Engineering (IoT))

20CSPC602 - COMPILER DESIGN

Regulations - 2020

Duration: 3 Hours Max. Marks: 100

PART - A (MCQ) (10 × 1 = 10 Marks)

Answer ALL Questions

		Marks	*K – Level*	*CO*
1.	Which computer program accepts the high-level language and converts it into assembly language? (a) Interpreter (b) Linker (c) Assembler (d) Compiler	*1*	*K1*	*CO1*
2.	How many phases are there in the Compilation process? (a) 7 (b) 6 (c) 8 (d) 4	*1*	*K1*	*CO1*
3.	In the context of lexical analysis, which of the following is an example of a token? (a) Keyword if (b) Integer value 42 (c) Identifier x (d) All of the above	*1*	*K1*	*CO2*
4.	Which of the following is true about lexical analyzers in terms of error handling? (a) They stop execution immediately when an error is found (b) They report errors but continue analyzing the rest of the code (c) They automatically fix the errors encountered (d) They do not need to report errors	*1*	*K1*	*CO2*
5.	Show the main difference between LL and LR parsers (a) LL parsers are top-down, while LR parsers are bottom-up. (b) LL parsers use more memory than LR parsers. (c) LR parsers cannot handle left-recursive grammar. (d) LL parsers cannot handle right-recursive grammar	*1*	*K2*	*CO3*
6.	Which of the following is NOT a type of Syntax Analyzer? (a) Top-down parser (b) Bottom-up parser (c) Recursive descent parser (d) Tokenizer	*1*	*K1*	*CO3*
7.	Which of the following is a common form of intermediate representation used in many modern compilers? (a) Three-address code (TAC) (b) Abstract Syntax Tree (AST) (c) Syntax Tree (d) Assembly code	*1*	*K1*	*CO4*
8.	Which of the following is typically done during intermediate code generation? (a) Lexical analysis (b) Generating assembly code (c) Performing machine-independent optimizations (d) Error handling	*1*	*K1*	*CO4*
9.	What is register allocation in the context of code generation? (a) Assigning variables to registers to minimize memory access (b) Generating machine code from the intermediate code (c) Selecting instructions for the target architecture (d) Mapping syntax tree nodes to instructions	*1*	*K1*	*CO5*
10.	What does the "constant folding" optimization do? (a) Replaces a function call with a constant value (b) Moves constant expressions out of loops (c) Evaluates constant expressions at compile-time and replaces them with their values (d) Combines multiple loops into a single loop	*1*	*K1*	*CO6*

PART - B (12 × 2 = 24 Marks)

Answer ALL Questions

11.	List the various Compiler Construction tools.	*2*	*K1*	*CO1*

12.	Illustrate diagrammatically how a language is processed.		*2*	*K2*	*CO1*
13.	Interpret the transition diagram for an identifier.		*2*	*K2*	*CO2*
14.	List the possible error recovery actions in lexical analyzer.		*2*	*K1*	*CO2*
15.	Convert Non Deterministic Grammar to Deterministic Grammar S->iEtS | iEtSeS | a E->b		*2*	*K3*	*CO3*
16.	What is an Operator Precedence Parser?		*2*	*K1*	*CO3*
17.	What is static checking?		*2*	*K1*	*CO4*
18.	Define Backpatching.		*2*	*K1*	*CO4*
19.	What is an Activation tree?		*2*	*K1*	*CO5*
20.	What is register descriptor and address descriptor?		*2*	*K1*	*CO5*
21.	What is code motion?		*2*	*K1*	*CO6*
22.	What is the induction variable?		*2*	*K1*	*CO6*

PART - C (6 × 11 = 66 Marks)

Answer ALL Questions

23.	a) (i)	Write the Regular Expression for Delimiters and keywords and construct transition diagrams for the same.	*6*	*K2*	*CO1*
	(ii)	Discuss the cousins of the compiler.	*5*	*K2*	*CO1*
		OR			
	b)	Explain the various phases of compiler and trace it with the program segment c=a+b*5.	*11*	*K2*	*CO1*
24.	a)	Translate DFA using Direct method for the following Regular Expression (a|b)*abb.	*11*	*K2*	*CO2*
		OR			
	b) (i)	Describe in detail the tool for generating lexical analyzer with an example program.	*6*	*K2*	*CO2*
	(ii)	Describe in detail about the issues in lexical analysis.	*5*	*K2*	*CO2*
25.	a)	Construct LL(1) parsing table for the following grammar S->iEtS | iEtSeS | a E->b Also check whether the given grammar is LL(1) or not?	*11*	*K3*	*CO3*
		OR			
	b)	Construct a SLR parsing table by examining the following grammar. S->CC C->cC | d and use the string "cdd" to parse with the SLR Parsing Table.	*11*	*K3*	*CO3*
26.	a)	Generate an intermediate code for the following code segment if (a > b) x = a + b else x = a - b.	*11*	*K3*	*CO4*
		OR			
	b)	Construct the following using the arithmetic expression a+- (b+c) . (i)Syntax tree (ii)Quadruples (iii)Triples (iv)Indirect Triples	*11*	*K3*	*CO4*
27.	a) (i)	Explain about simple code generation algorithm in detail.	*6*	*K2*	*CO5*
	(ii)	Explain any four issues when designing a code generator.	*5*	*K2*	*CO5*
		OR			
	b)	Explain about runtime storage management.	*11*	*K2*	*CO5*

28. a) Explain in detail about optimization of Basic Blocks and Construct the DAG for the following Basic Block. *11 K3 CO6*

1. t1:=4*i
2. t2:=a[t1]
3. t3:=4*i
4. t4:=b[t3]
5. t5:=t2*t4
6. t6:=prod+t5
7. prod:=t6
8. t7:=i+1
9. i:=t7
10. if i<=20 goto(1)

OR

b) Apply global data flow analysis for the following. *11 K3 CO6*