ACADEMIC YEAR: 2025-26
REGULATION

DEPARTMENT OF CSE &IT

SUBJECT CODE / NAME: / 23CS6401 -COMPILER DESIGN
YEAR / SEM: III / VI

UNIT I INTRODUCTION TO COMPILER DESIGN
QUESTION BANK

Q.NO

PART- A Questions

COMPILERS-ANALYSIS OF THE SOURCE PROGRAM

1. Define compiler. K1

2. What are the three phases of analysis of the source program by compiler? K1

3. What is linear analysis? K1

PHASES OF A COMPILER-TYPES OF COMPILER

4. List the phases that constitute the front end of a compiler. K1

5. Draw the parse tree showing operator precedence for the expression: Position := initial + rate * 60.” K2

6. Define the term cross compiler K1

7. How does the parser derive the sequence of atoms and represent them in the form of a syntax tree for the given Java source statements?i) a = (b + c) * d; ii) if (a < b) a = a + 1; K2

8. How the given C source inputs can be optimized using global optimization methods? *a)if (x>0) {x = 2; y = 3;}else {y = 4; x = 2;} b)if (x>0) x = 2; else if (x<=0) x = 3; else x = 4;* K2

9. Compare syntax tree and parse tree. K2

10. State the purpose of syntax analysis. K1

11. How can local optimization techniques improve the efficiency of the given object code segments? K2

(a) LD R1,A MULT R1,B ST R1,Temp1 LD R1,Temp1 ADD R1,C ST R1,Temp2

(b) LD R1,A ADD R1,B ST R1,Temp

(c) CMP A,B BH L1
B L2
L1: MOV A,B B L3
L2: MOV B,A L3:

ROLE OF LEXICAL ANALYZER-INPUT BUFFERING

12. What are the issues of the lexical analyser? K1

13. Distinguish tokens, patterns, and Lexeme K2

14. What is sentinel? Mention its usage K1

15. List the lexemes present in the given C++ program and identify the lexemes that are associated with lexical values. K1

float limitedSquare(x) float x {

/ returns x-squared, but never more than 100 */ return (x<=-10.0||x>=10.0)?100:x*x;*
}

16. How does the lexical analyzer identify tokens in the given Java source inputs? K2
a.) while (1.3e-2 if &&
b. if 1.2.3 < 6

17. Define buffer pair K1

18. Why is buffering used in lexical analysis? What are the commonly used buffering. K2

19. How are lexical tokens generated from the given java source inputs? K2

(a) for (i=1; i<5.1e3; i++) func1(x);
(b) if (sum!=133) / sum = 133 */*

SPECIFICATION OF TOKENS-RECOGNITION OF TOKENS

20. What is a regular expression? State the rules, which define regular expression? K1

21. How does the regular expression represent the language $L= \{a^{n}b^{m}/n+m)$ *is even}?* K2

22. How is the transition diagram used to recognize relational operators?. K2

FINITE AUTOMATA- REGULAR EXPRESSIONS TO AUTOMATA-MINIMIZING DFA

23. Differentiate NFA and DFA. K2

24. How does the NFA recognize the regular expression (a/b)*? K2

25. Define Finite Automata K1

26. How is an NFA structured over the alphabet Σ = {0, 1}? K2

27. How does the regular expression represent all strings ending with 00 over Σ={0,1}? K2

LANGUAGE FOR SPECIFYING LEXICAL ANALYZERS - DESIGN OF LEXICAL ANALYZER GENERATOR(LEX) - RECENT TRENDS IN COMPILER DESIGN.

28. What is meant by a language for specifying lexical analyzers? K1

29. How is a lexical analyzer generator (LEX) designed? K2

30. List out the recent trends in compiler design. K1

PART- B

1. Discuss in detail about the role of Lexical analyzer with the possible error recovery actions. K2 (16)

2. Describe about token specification. K2 (8)
   Discuss the various types of compiler with an example. K3 (8)

3. Explain the various phases of a compiler with an illustrative example. K2 (16)

4. Describe the Input buffering techniques in detail. K2 (16)

5. Apply Thompson's construction to convert the regular expression into its equivalent NFA with suitable state transitions.Illustrate regular expression to NFA for the sentence (a|b)*a K3 (16)

6. Explain in detail the recognition of tokens. K2 (16)

7. Apply DFA construction and minimization methods to derive the minimized DFA for the given regular expression *(0+1)*(0+1) 10* K3 (16)

8. Apply the direct DFA construction method to convert the regular expression *(a+b)*abb* into an equivalent DFA. K3 (16)

9. Analyze the relationship between NFA–DFA conversion and DFA minimization by explaining both algorithms with an appropriate example. K4 (16)

10 i)Analyze the given regular (1*01*0)*1* over the alphabet Σ={0,1} and construct an equivalent NFA using Thompson's construction. K4 (8)
   ii) Describe the major issues faced during lexical analysis. K2 (8)

11 Apply Thompson's construction algorithm to construct an NFA for the regular expression *(a/b)*a(a/b)* K3 (16)

12 i) Analyze the construction of a DFA from a regular expression by examining each transformation step with a suitable example. K4 (8)
   ii) Apply the principles of transition diagram construction to model relational operators and unsigned numbers in Pascal. K3 (8)

13 Apply DFA minimization techniques to demonstrate that the two given regular expressions are equivalent. K3 (16)
   *a) (a | b) **
   *b) (a * | b *) **

14 Design an appropriate DFA to accept the specified language *(a|b)* a* K3 (16)

15 Analyze how finite automata are used to represent tokens and perform lexical analysis, with suitable examples. K4 (16)

16 Explain the Language for Specifying Lexical Analyzers and the Design of Lexical Analyzer Generator (LEX). K2 (16)