Short Questions

1. What is the structure of a compiler?
2. Define lexical analysis.
3. What is the role of the lexical analyzer?
4. Explain input buffering in lexical analysis.
5. How are tokens recognized in lexical analysis?
6. What is Lex, the lexical-analyzer generator?
7. Describe finite automata in lexical analysis.
8. How are regular expressions related to automata?
9. Explain the design of a lexical-analyzer generator.
10. What are DFA-based pattern matchers?
11. Define programming language basics.
12. What is the science of building a compiler?
13. How does Lex help in generating lexical analyzers?
14. What are the advantages of using finite automata in lexical analysis?
15. How are regular expressions converted to automata?
16. Discuss the importance of optimization in DFA-based pattern matchers.
17. Explain the concept of tokenization in lexical analysis.
18. What is the purpose of the lexeme buffer?
19. How does Lex simplify the process of lexical analysis?
20. Describe the architecture of a compiler.
21. What are the components of a lexical analyzer?
22. How does the lexical analyzer handle white spaces and comments?
23. Discuss the significance of token recognition in lexical analysis.
24. How are lexical errors detected and handled?
25. Explain the concept of lexeme extraction.
26. What role does the lexeme table play in lexical analysis?

27. Describe the process of constructing a lexical analyzer using Lex.
28. What are the challenges in designing a lexical analyzer generator?
29. Discuss the role of deterministic finite automata in lexical analysis.
30. How does Lex prioritize patterns in lexical analysis?
31. Explain the concept of lexical token classification.
32. What are the characteristics of a well-designed lexical analyzer?
33. Describe the process of defining tokens in Lex.
34. How does Lex generate code for lexical analysis?
35. Discuss the benefits of using Lex over manual lexical analysis.
36. What are the limitations of regular expressions in lexical analysis?
37. Explain the difference between NFA and DFA in lexical analysis.
38. How does Lex handle ambiguous patterns?
39. Discuss the impact of lexical analysis on compiler performance.
40. What are the advantages of using Lex for lexical analysis?
41. Describe the role of input buffering in lexical analysis efficiency.
42. How does Lex optimize the generated lexical analyzer?
43. Discuss the trade-offs involved in lexical analysis optimization.
44. Explain the concept of lexical tokenization.
45. What is the purpose of pattern matching in lexical analysis?
46. How does Lex handle token patterns with overlapping definitions?
47. Discuss the significance of pattern prioritization in Lex.
48. What are the characteristics of a good lexical analyzer generator?
49. Describe the steps involved in designing a lexical analyzer using Lex.
50. How does Lex generate efficient lexical analyzers for different languages?
51. What is Syntax Analysis?
52. What is a Context-Free Grammar (CFG)?
53. How do you write a grammar for a language?
54. What is Top-Down Parsing?

55. What is Bottom-Up Parsing?
56. What is LR Parsing?
57. What makes Simple LR Parsing unique?
58. How do LR Parsers differ in power?
59. What are Ambiguous Grammars?
60. What are Parser Generators?
61. What is a Terminal in CFG?
62. What is a Non-Terminal in CFG?
63. What is a Production Rule in CFG?
64. What is a Derivation in CFG?
65. What is a Parse Tree?
66. What is Left Recursion in Grammars?
67. How is Left Recursion Eliminated?
68. What is a Lookahead in Parsing?
69. What is a Shift-Reduce Conflict?
70. What is a Reduce-Reduce Conflict?
71. What is a Parsing Table?
72. What is a Grammar Ambiguity?
73. How can Ambiguity be resolved?
74. What is Predictive Parsing?
75. What is a Recursive Descent Parser?
76. What is Backtracking in Parsing?
77. What is a Canonical LR Parser?
78. What is LALR Parsing?
79. What is a Handle in Parsing?
80. What is a LR(0) Item?
81. What is an SLR Parser?
82. How do you construct an SLR Parsing Table?

83. What is a Follow Set in CFG?
84. What is a First Set in CFG?
85. What is a Sentential Form in CFG?
86. What is a Derivation Tree?
87. What is a Recursive Grammar?
88. What is a LL Parser?
89. How is a Grammar's Ambiguity Detected?
90. What is a Context-Free Language?
91. What is a Symbol Table in Parsing?
92. What is Semantic Analysis?
93. What is a Compiler?
94. What is an Interpreter?
95. What is a Grammar Conflict?
96. How is a Conflict Resolved in LR Parsing?
97. What is the role of a Parser Generator in Compiler Construction?
98. What are the advantages of Bottom-Up Parsing over Top-Down Parsing?
99. What is the difference between Static and Dynamic Parsing?
100. Why is LR Parsing preferred for Compiler Construction?
101. What is Syntax-Directed Translation?
102. What are Syntax-Directed Definitions (SDDs)?
103. What is an Evaluation Order for SDDs?
104. What are the Applications of Syntax-Directed Translation?
105. What is a Syntax-Directed Translation Scheme?
106. How are L-attributed SDDs Implemented?
107. What are Attributes in SDDs?
108. What is the difference between Synthesized and Inherited Attributes?
109. What is a Dependency Graph in SDDs?
110. Why is an Evaluation Order important in SDDs?

111. How do SDDs relate to Parsing Techniques?
112. What is a Semantic Action?
113. How can SDDs be used in Code Generation?
114. 114.What is an L-Attributed Definition?
115. 115.What are the challenges in implementing SDDs?
116. 116.What role do SDDs play in Compiler Design?
117. 117.How do you choose between Top-Down and Bottom-Up Parsing for SDDs?
118. 118.What is a Translation Grammar?
119. 119.How are Inherited Attributes evaluated in Top-Down Parsing?
120. 120.Can SDDs handle semantic errors?
121. 121.What is a Directed Acyclic Graph (DAG) in the context of SDDs?
122. 122.How does SDT differ from Traditional Parsing?
123. 123.What is the significance of Evaluation Strategies in SDDs?
124. 124.What is Attribute Flow in SDDs?
125. How is Semantic Analysis integrated with Syntax-Directed Translation?