Short Questions

1. What is a finite automaton?
2. Describe the structural representation of automata.
3. How does automata theory relate to computational complexity?
4. Define an alphabet in automata theory.
5. What is a string in the context of automata theory?
6. Explain the concept of languages in automata.
7. What problems does automata theory aim to solve?
8. Give an example of a problem solved by finite automata.
9. How does automata theory apply to computational problems?
10. Define deterministic finite automaton (DFA).
11. What is a nondeterministic finite automaton?
12. Provide a formal definition of NFA.
13. Discuss an application of NFA.
14. How is text search an application of NFA?
15. Explain finite automata with epsilon-transitions.
16. How do epsilon-transitions work in NFAs?
17. What are the key differences between DFA and NFA?
18. Can NFAs have multiple transitions for the same input?
19. How do NFAs handle epsilon-transitions?
20. Discuss the conversion of NFA with epsilon-transitions to NFA without epsilon-transitions.
21. How does a DFA process strings?
22. Describe the language recognized by a DFA.
23. Explain the process of converting an NFA without epsilon-transitions to a DFA.
24. What is the significance of the state transition diagram in DFA?
25. How do you design a DFA for a given language?
26. Discuss the process of minimizing a DFA.
27. What are dead states in DFA, and how are they identified?
28. How does DFA differ from NFA in terms of computational efficiency?
29. Provide an example of a language that can be recognized by a DFA.
30. Explain the significance of the start state in DFA.
31. Describe the subset construction algorithm for NFA to DFA conversion.
32. How do you handle epsilon-transitions during NFA to DFA conversion?
33. What challenges arise in converting NFA to DFA?
34. Compare the states of the original NFA with the states of the resulting DFA.
35. Discuss the importance of the acceptance state in the conversion process.
36. How do you determine the final states in the converted DFA?
37. What is the impact of nondeterminism on the conversion process?
38. Explain how the transition table is used in NFA to DFA conversion.

39. Provide an example of converting a simple NFA to a DFA.
40. Discuss the efficiency of the resulting DFA compared to the original NFA.
41. Why is automata theory important in compiler design?
42. How do finite automata contribute to lexical analysis in compilers?
43. What is the role of DFA in pattern matching algorithms?
44. How can NFAs be used in the optimization of regular expressions?
45. Discuss the limitations of finite automata in recognizing certain languages.
46. How does the concept of states facilitate computation in automata theory?
47. Why are epsilon-transitions significant in automata theory?
48. What are the computational limitations of deterministic finite automata?
49. How do automata theory concepts apply to software engineering problems?
50. Explain how automata theory helps in understanding the foundations of computer science.

Unit - II

51. What is the relationship between finite automata and regular expressions?
52. Provide examples of applications for regular expressions.
53. Discuss the algebraic laws for regular expressions.
54. How can finite automata be converted to regular expressions?
55. Explain how regular expressions are used in text search algorithms.
56. What is the significance of concatenation in regular expressions?
57. Describe the role of union and Kleene star in regular expressions.
58. How do regular expressions handle optional elements?
59. Give an example of a complex regular expression and explain its components.
60. Discuss the limitations of regular expressions in pattern matching.
61. State the pumping lemma for regular languages.
62. How can the pumping lemma be applied to prove non-regularity of a language?
63. Provide an example where the pumping lemma is used to show a language is not regular.
64. Explain the significance of the pumping length in the pumping lemma.
65. Discuss the conditions under which the pumping lemma holds for a regular language.
66. What are the common mistakes made when applying the pumping lemma?
67. How does the pumping lemma relate to the concept of regularity in languages?
68. Provide an application of the pumping lemma outside proving non-regularity.
69. What are the limitations of the pumping lemma in language theory?
70. Discuss the role of the pumping lemma in automata theory and compiler design.

71. Define context-free grammars (CFGs).
72. Explain the process of derivation in a context-free grammar.
73. What is the difference between leftmost and rightmost derivations in CFGs?
74. Describe the concept of the language of a grammar.
75. Explain the structure and purpose of parse trees in CFGs.
76. What is ambiguity in grammars, and why is it significant?
77. How can ambiguity in grammars affect language processing?
78. Provide an example of an ambiguous grammar and discuss how to resolve the ambiguity.
79. Discuss the importance of context-free grammars in compiler design.
80. How are context-free grammars used in parsing algorithms?
81. Explain how regular expressions contribute to the functionality of lexical analyzers in compilers.
82. Discuss the challenges in converting finite automata to regular expressions.
83. How does the concept of ambiguity in CFGs impact compiler design?
84. What are the practical applications of context-free grammars in software development?
85. How do regular expressions and CFGs compare in expressing languages?
86. Discuss the limitations of context-free grammars in representing natural languages.
87. How is the pumping lemma used in optimizing compilers?
88. What role do parse trees play in the syntax analysis phase of compilers?
89. Explain the significance of algebraic laws in optimizing regular expressions for search algorithms.
90. How do ambiguities in grammars affect the parsing process in compilers?
91. Discuss the impact of non-regular and context-free languages on computational theory.
92. How can the study of regular expressions and CFGs enhance understanding of programming language syntax?
93. What are the implications of the pumping lemma for the design of programming languages?
94. How do context-free grammars facilitate the understanding of language hierarchies in computational theory?
95. Discuss the relationship between parse trees and abstract syntax trees in compiler construction.
96. How does the study of automata theory and compiler design contribute to advancements in artificial intelligence?
97. What is the role of context-free grammars in the development of domain-specific languages?
98. Explain how advancements in understanding regular expressions and CFGs have influenced modern computing technologies.
99. Discuss the future trends in automata theory and their potential impact on compiler design.

100. How do regular expressions and context-free grammars contribute to the field of computational linguistics?

Unit - III

101. Define a Pushdown Automaton (PDA).
102. Describe how a PDA processes input strings.
103. Explain the concept of acceptance by final state in PDA.
104. How do the languages of a PDA compare to regular languages?
105. Discuss the equivalence of PDAs and Context-Free Grammars (CFGs).
106. Provide an example of a language that can be recognized by a PDA but not by a finite automaton.
107. How can a PDA be constructed from a given CFG?
108. What role do stack operations play in a PDA?
109. Explain the significance of non-determinism in PDAs.
110. How does acceptance by empty stack differ from acceptance by final state in PDAs?
111. Introduce the concept of a Turing Machine (TM).
112. Provide a formal description of a Turing Machine.
113. Explain the role of the tape and the head in a Turing Machine.
114. Discuss the concept of instantaneous description in the context of TMs.
115. How does a TM differ from finite automata and PDAs in terms of computational power?
116. Describe the language of a Turing Machine.
117. Provide an example of a problem solvable by a Turing Machine that is not solvable by a PDA.
118. Discuss the significance of Turing Machines in the theory of computation.
119. How do Turing Machines model the concept of algorithms?
120. Explain the concept of decidability in relation to Turing Machines.
121. Define undecidability in the context of computational theory.
122. Provide an example of an undecidable problem.
123. Discuss the significance of the halting problem in the study of undecidability.
124. Explain why some languages are not recursively enumerable.
125. How do undecidable problems about Turing Machines highlight limitations in computation?