Unit I

1. Make an NFA for the regular expression (a|b)*abb and convert it intoan equivalent DFA.

2. Describe the major data structures used in a compiler and their roles.

3. Explain in brief

   (i) The role of the lexical analyzer.

   (ii) Lexeme, Token & Pattern.

4. How are tokens specified and recognized in lexical analysis?

5. What is bootstrapping in compiler design? Provide an example.

Unit - II

1. Differentiate between top-down parsing and bottom-up parsing with suitable examples.

2. Construct LL(1) parsing table for the following grammar :

   SaBDh BcC CbC|

   DEF Eg| Ff|

3. Consider the grammar:

   S AaAb| BbBa A B

   Show that the grammar is LL (1) but not SLR.

4. Construct LALR parsing table for the following set of productions.

   SCC CcC Cd

5. Define syntax-directed definitions (SDDs) and their role in compilers.

Unit - III

1. Compare and contrast static and dynamic type checking in terms of error detection, performance, and flexibility. How do these differences impact compiler design?

2. Evaluate the impact of different storage allocation strategies (static, stack, heap) on runtime performance and memory management. Provide examples where each strategy is most appropriate.

Unit - IV

1. Write Three Address Code for the following expression-

If A < B and C < D then t = 1 else t = 0

2. Translate the following expression to quadruple, triple and indirect triple-

a + b x c / e ↑ f + b x c

Unit - V

1. Consider the following expression and construct a DAG for it-

(a + b) x (a + b + c)

2. Consider the following expression and construct a DAG for it-

(((a + a) + (a + a)) + ((a + a) + (a + a)))