**UNIT I**

1. Discuss about Chomsky hierarchy.
2. Explain about programs related to compilers
3. Explain the translation process with diagram.
4. Explain about major data structures in a compiler.
5. Differentiate between analysis and synthesis parts of compiler.
6. Compare Front end and Back end view of complier.
7. Compare pass and phase of a compiler.
8. Define compiler pragmatics. (Page No 17)
9. Explain bootstrapping with a neat diagram.
10. Explain porting with a neat diagram.
11. Briefly describe the TINY language (Page No 22)
12. Define pattern, token, lexeme.
13. Explain the scanning process with a neat diagram.
14. Define regular expression.
15. Discuss the operations, properties of regular expressions.
16. Discuss about the extensions to regular expressions.
17. Write regular expression for signedNatural number (Page No 43 ).
18. Write regular expression for whitespace (Page No 46).
19. Write regular expression for identifier (Page No 46).
20. Construct finite automata for identifier ( Page No 48)
21. Construct finite automata for identifier with error transitions ( Page No 50)
22. Construct finite automata for floating point numbers ( Page No 52 , fig 2.3)
23. Construct finite automata for C-style comments ( Page No 53)
24. Construct finite automata for identifier with delimiter and return value ( Page No 54)
25. Construct an NFA for regular expression ab|a
26. Construct an NFA for the regular expression **letter(letter|digit)\*** using Thompson’s Construction ( Page No 68 Fig 2.9)
27. Convert the above NFA to DFA using subset construction method (Page No 72)
28. List the tokens of TINY language ( Page No 75, Table 2.1)
29. Construct DFA for the TINY scanner (Page No 77 , fig 2.10)
30. Give structure / format of **Lex** input file. ( Page No 83)
31. Write a program in **Lex** to add line numbers to the input file ( Page No 84 )

**UNIT II**

1. Define a CFG (Context Free Grammar) ( Page No 128)
2. Discuss about scanning process ( Page No 96 )
3. Define left recursive grammar ,right recursive grammar.
4. Discuss how to remove left recursion from the given grammar with an example.
5. Define abstract syntax tree.

or

1. Define syntax tree.
2. Define left most derivation and right most derivation.
3. Discuss about EBNF ( page No 123) [ IMP ]
4. Give equivalent EBNF for A🡪Aα| β (Page No 123) [ IMP ]
5. Discuss about Syntax Diagrams ( Page No 125) [ IMP ]
6. Construct syntax diagram for factor🡪(exp) | **number** ( Page No 125)
7. Construct syntax diagram for A🡪{B} and A🡪[B] (Page No 126)
8. Convert the following BNF into EBNF and then construct syntax diagram for the same (Page No 127) [ IMP ]

Statement 🡪if-stmt |**others**

If-stmt🡪**if** (exp) statement | **if** (exp) statement **else** statement

Exp🡪 **0** | **1**

1. Define sentential form, language generated by a Grammar. ( page NO 129)
2. Define leftmost derivation, rightmost derivation. ( page NO 129)
3. Define parse tree ( page NO 129)
4. Define ambiguous grammar.
5. Discuss about Chomsky and the 4 types of languages.
6. Construct syntax trees for **if ,repeat** , **assign** ,**write**,**op**<opkind> statements

(Page no 136-137)

1. Discuss about top-down parsers.
2. Explain construction of RD (recursive descent) parser with an example.
3. List the rules for computation of FIRST(X) or FIRST.
4. List the rules for computation of FOLLOW (B) or FOLLOW.
5. Write the algorithm for construction of predictive LL (1) parser.
6. Explain the procedure for elimination of left recursion with an example. ( Page No 158)
7. Explain the procedure for left factoring with an example. ( Page No 162 )
8. Construct LL(1) parsing table for

**exp🡪term exp ‘**

**exp ‘🡪 addop term exp’ | ε**

**addop🡪 + | -**

**term 🡪factor term’**

**term’🡪mulop factor term’ | ε**

**mulop🡪 \***

**factor 🡪 (exp) | number**

1. Discuss error recovery in RD parsers (page no 183 )
2. Discuss error recovery in LL(1) parsers. (page no 186)
3. Discuss panic mode error recovery.( page no 183)
4. Explain Error recovery in Top-Down parsers.

**UNIT III**

1. Explain about bottom up parsers .

or

1. Explain about shift reduce parsers.
2. Define handle.
3. Define handle pruning.
4. Define viable prefix.
5. Define LR (0) item.
6. Discuss about closure(X) and goto() functions.
7. Write algorithm for construction of SLR parser.
8. Write algorithm for construction of LR parser.
9. Write algorithm for construction of LALR parser.
10. Give structure / format of **YACC** input file.
11. Write a program for a simple calculator ( Page No 228)
12. Explain error recovery in Bottom-up parsers.
13. Discuss panic mode error recovery in Bottom up parsers ( Page No 245)
14. Discuss error recovery in YACC (Page No 247).
15. Define an attribute.
16. Define attribute grammar.
17. Define attribute equation or semantic rule.
18. Explain attribute grammar with an example ( Page no 263 table 6.1 ,fig 6.1)
19. Define meta language for the attribute grammar ( Page No 268 )
20. Discuss about dependeny graph with an example ( Page No 271,for example fig 6.6)
21. Discuss about topological sort (Page No 275 , example 6.9)
22. Define and Explain about synthesized and inherited attributes.( Page No 277)
23. Explain about attributes as parameters and return values.
24. Explain the use of external data structures to store attribute values
25. Explain about symbol table.
26. Explain the structure of symbol table with a neat diagram ( Page No ,fig 6.12)
27. Compare separate chaining and open addressing. ( Page No 296)
28. Discuss about / kinds declarations in symbol table.
29. Explain about scope rules and Block structures in symbol table.
30. Explain about data types and type checking.
31. Explain different type expressions and type constructors.
32. Discuss about different Type equivalences.
33. Discuss Type inferences and type checking.
34. Discuss a semantic analyzer for the TINY language.

NOTE : Also practice **ALL** the problems solved in the class (both top down & bottom up )

**UNIT IV**

1. Explain memory organization during program execution.

2. Explain about procedure activation record.

3. Explain calling sequence. (Page no 359)

4. Explain return sequence. (Page no 359)

5. Explain static runtime environments.

6. Explain Stack Based Runtime Environments.

7. Explain Stack Based Runtime Environments without local procedures.

8. Explain activation trees.

9. Explain Stack Based Runtime Environments with local procedures.

10. Describe access chaining. (Page no 367)

11. Describe access link.

12. Describe is control link.

13. Define closure.

14. Explain Stack Based Runtime Environments with procedure parameters.

15. Explain about Fully dynamic Runtime Environments.

16. Define dangling reference problem.

17. Discuss about garbage collection (Page no 374)

18. Explain Dynamic memory in Object Oriented Languages.

19. Define inheritance graph.

20. Define virtual function table.

21. Explain Heap Management.

22. Explain Automatic Management of Heap Memory.

23. Define memory compaction.

24. Discuss Mark and Sweep garbage collection.

25. Discuss Stop and Copy garbage collection.

Or

26. Discuss two space garbage collection.

27. Discuss generational garbage collection.

28. Explain different parameter passing mechanisms.

29. Explain with example pass by value parameter passing mechanism.

30. Explain with example pass by reference parameter passing mechanism.

31. Explain with example pass by value-result parameter passing mechanism.

32. Explain with example pass by name parameter passing mechanism.

33. Explain runtime environment for the TINY Language

**UNIT V**

1. Explain intermediate code and data structures for code generation.
2. Discuss about three address codes. ( or 3AC or TAC )
3. Discuss about data structures for implementing 3AC.
4. Write P-code for the following code 2\*a+(b-3) (page no 404 )
5. Compare TAC and P-code.
6. Explain basic code generation techniques.
7. List the differences between instructions **sto** and **stn** (page no 407)
8. Write attribute grammar of P-code as a synthesized string attribute ( Page no 408 table 8.1)
9. Write attribute grammar for TAC as a synthesized string attribute ( Page no 409 table 8.2)
10. Explain generation of target code from the intermediate code.
11. Compare Macro expansion and static simulation ( Page no 413)
12. Write P-code for a=b+c.
13. Explain code generation of Data Structure reference.
14. Discuss about address calculation ,Three-Address code for Address Calculations, and P-code for
15. Address Calculations.
16. Discuss about instruction **ind** and **ixa** with an example. ( Page No 417)
17. Discuss about Array References.
18. Explain three address code for array references
19. Explain record structure and pointer references.
20. Explain code generation of control statements and logical expressions.
21. Explain about code generation of IF and WHILE statements.
22. Define fall through ( Page No 429)
23. Explain Backpatching.
24. Explain code generation of logical expression
25. Explain code generation of procedure and function calls.
26. Explain code generation for Borland 3.0 C compiler for 80x86.
27. Explain code generation for Sun 2.0 C compiler for Sun SPARCstation.
28. Discuss about common subexpression elimination with an example.
29. Discuss about unreachable code with an example
30. Discuss about dead code elimination with an example
31. Discuss about strength reduction with an example
32. Discuss about copy propagation with an example
33. Discuss about constant folding with an example
34. Discuss about procedure in lining ( Page No 470)
35. Explain principal sources of code optimization.
36. Differentiate source level and target level optimization.
37. Define a Basic block.
38. Define a Leader.
39. List the rules to recognize a Leader.
40. Discuss data flow analysis.
41. Define a induction variable.
42. Explain construction of flow graph with an example.
43. Explain construction of DAG with an example