Important Questions

CHAPTER - 1: INTRODUCTION TO COMPILER DESIGN

1. What is the pass of a compiler? Explain how single pass and multi pass compilers work.
2. List out phases of a compiler. Write a brief not on Lexical Analyzer.
3. Explain linker & loader.
4. For a statement given below, write output of all phases of a compiler.
   a = a+b*c

CHAPTER - 2: LEXICAL ANALYZER

1. Draw deterministic finite automata for:
   (0+1)*101(0+1)*
   10(0+1)*1
2. Construct DFA for following regular expression without constructing NFA and optimize the same.
   (a/ϵ)*ab(a/b)*#
3. Construct a DFA without constructing NFA for the following regular expression.
   (a/b)*a
4. Explain subset construction method with an example.
5. Construct DFA without constructing NFA for the following regular expression.
   a*b*a(a/b)b*a# minimize the same.
6. Construct NFA for the following regular expression using Thompson's notation and then convert it into DFA.
   a(b/c)*a*c#
7. Draw the state transition diagram for the unsigned numbers.
8. Convert the (a/b/c)*d*(a*/b)ac+# regular expression to DFA directly and draw its DFA.

CHAPTER - 3: PARSING THEORY

1. Implement the following grammar using table driven parser and check whether it is LL(1) or not

   S->aBDh

   B->cC

   C->bC/ϵ

   D->EF

   E->g/ϵ

   F->f/ϵ
2. Implement the following grammar using Recursive Descent Parser.
3. What is bottom-up parsing? Discuss Shift Reduce parsing technique in brief.
4. What is a handle?
5. Write a syntax directed definition of a simple desk calculator and draw an annotated parse tree for 4*3+2*5 n.
6. Define an operator precedence grammar. Also write down the rules to find relationship between each pair of terminal symbols.
7. Construct SLR parsing table for the following grammar.

   E -> E+T/T

   T -> T*F/F

   F -> (E)/a
8. Differentiate Synthesized and Inherited attributes.
9. Write a brief note on input buffering techniques.
10. How do the parser and scanner communicate? Explain the block diagram of communication between them.
11. Construct a SLR parsing table for following grammar.

    S -> aAb/bB

    A -> Aa/ϵ

    B -> Bb/ϵ
12. Check whether the given grammar is LL(1) or not?

S -> aAC/bB

A -> Abc/Abd/e

B -> f/g

C -> h/i

13. Construct the LALR parsing table for the following grammar.

    S -> CC

    C -> aC

    C -> d

14. Write a syntax directed definition for desk calculator. Justify whether this is an S attributed definition or L-attributed definition. Using definition draw annotated parse tree for 3*5+4n.
15. Consider the grammar S -> SS+/SS*/a
    Show that the string aa+a* can be generated by the grammar. Construct the parse tree for the grammar. Is the grammar ambiguous?
16. Write unambiguous production rules for if then else construction.
17. Compare top-down and bottom-up parser.
18. Explain right-most-derivation –in-reverse with the help of an example.
19. Explain SLR parser. How is its parser table constructed?
20. Construct a precedence graph, precedence table for operator precedence parser to be used for parsing a string consisting of id, -, *, $. Parse following string.
    $ id – id * id $
21. Explain synthesized attributes with the help of an example.
22. Explain left factoring with the help of an example.
23. Design the FIRST SET and FOLLOW SET for the following grammar.

    E -> E+T/T

    T -> T*F/F

    F -> (E) / id

24. Differentiate SLR, Canonical LR and LALR.

CHAPTER - 4: ERROR RECOVERY

1. Error Recovery strategies of compiler.
2. Write down short note on Error-Recovery strategies.

CHAPTER - 5: INTERMEDIATE CODE GENERATION

1. What is an Intermediate Code? Benefits of Intermediate Code. Explain different intermediate forms
2. Explain the three address code with example and its merits & demerits.

CHAPTER - 6: RUN TIME MEMORY MANAGEMENT

1. Explain Storage allocation Strategies.
2. What is an Activation Record ? Explain how they access local & global variables.
3. Explain Parameter Passing.
4. What is Symbol table? Explain all the values in symbol table.
5. Explain Dynamic storage allocation techniques.

CHAPTER – 7: CODE OPTIMIZATION

1. What is Code Optimization ? Explain Classification and its techniques.

CHAPTER – 8: CODE GENERATION

1. What is Code Generation ? Explain its issues.
2. Explain the DAG representation of basic blocks with example and its application.
3. Explain Peephole Optimization
4. Explain Dynamic programming code generation algorithm.