Question set

1. Eliminate the left recursion and left factoring from the given grammar.

2. Design a DFA directly from the regular expression.

3. Draw a transition diagram to identify the keywords (THEN, ELSE, DO etc.), relational operator, and identifier.

4. Construct predictive parsing table.

5. Construct the sets of LR (0) items for the given grammar.

6. Compute the FIRST and FOLLOW sets for each nonterminal of the grammar.

7. Construct the sets of LR (1) items for the grammar.

8. NFA to an equivalent DFA conversion.

9. Design a Finite Automata that accepts set of strings such that every string ends with 00, over alphabets {0,1}.

10. Why the use of CFG is not preferred over regular expressions for defining the lexical syntax of a language?

11. Prove the grammar is ambiguous. E→ E + E | E * E | (E) | id

12. What is top-down parsing? Explain with the help of an example. Name the different parsing techniques used for top-down parsing.

13. Explain shift-reduce parsing with stack implementation.

14. Consider the following grammar and show the handle of each right sentential form for the string (b, (b, b)).

E →(A) | b
A → A,E | E

15. Explain ACTION and GOTO function in LR parsing.

16. Give the algorithm for the construction of canonical LR parsing table.