Multiple Choice Question

Compiler Design

1. Top-down parsing is a technique to find ——————.

(a) Leftmost derivation (b) Rightmost derivation
(c) Leftmost derivation in reverse (d) Rightmost derivation in reverse

2. Predictive parsing is possible only for ——————.

(a) LR(k) grammar (b) LALR(1) grammar
(c) LL(k) grammar (d) CLR(1) grammar

3. Which two functions are required to construct a parsing table in predictive parsing technique?

(a) CLOSURE() and GOTO () (b) FIRST() and FOLLOW()
(c) ACTION() and GOTO() (d) None of these

4. Non-recursive predictive parser contains ——————.

(a) An input buffer (b) A parsing table
(c) An output stream (d) All of these

5. Which of these parsing techniques is a kind of bottom-up parsing?

(a) Shift-reduce parsing (b) Reduce-reduce parsing
(c) Predictive parsing (d) Recursive-decent parsing

6. Which of the following methods is used by the bottom-up parser to generate a parse tree?

(a) Leftmost derivation (b) Rightmost derivation
(c) Leftmost derivation in reverse (d) Rightmost derivation in reverse

7. Handle pruning forms the basis of ——————.

(a) Bottom-up parsing (b) Top-down parsing
(c) Both (a) and (b) (d) None of these

8. In shift-reduce parsing, accept action occurs ——————.

(a) When we have the right end of the handle at the top of the stack
(b) When we have the left end of the handle at the top of the stack
(c) When parser declares the successful completion of parsing
(d) When the parser finds a syntax error in the input and calls an error recovery Routine

9. Which of the following operators is hard to handle by the operator precedence parser?

(a) Plus (+) (b) Minus (-)
(c) Multiply (*) (d) Divide (/)

10. Given a grammar G:

T → BCTd | Bcd
CB → BC
Cc → cc
Bc → bc

Bb → b
Which of the following sentences can be derived by G?
(a) bcd (b) bbc
(c) bcdd (d) bccd

11. The simplest LR parsing technique is ——————.
(a) CLR parser (b) SLR parser
(c) LALR parser (d) LL parser

12. The most common non-backtracking shift-reduce parsing technique is known as ————

—.
(a) LL parsing (b) LR parsing
(c) Top-down parsing (d) Bottom-up parsing

13. X → A.BC, the given item indicates that ——————.
(a) a string derivable from ABC is expected next on the input.
(b) a string derivable from BC has already been seen and now a string derivable from A is expected on the input.
(c) a string derivable from A has already been seen on the input and now a string derivable from BC is expected.
(d) the body of the production has already been seen, and now it is time to reduce it to X.

14. Shift-reduce and reduce-reduce conflicts occur in ——————.
(a) SLR parser (b) LALR parser
(c) CLR parser (d) None of these

15. A parser that accommodates some extra information in the form of a terminal symbol, as a second component is known as ——————.
(a) SLR parser (b) LALR parser
c) CLR parser (d) LL parser

16. —————— parsers are specialized form of LR parsers that lie in between SLR parsers and canonical LR parsers in terms of power of parsing grammars.
(a) LALR parser (b) LR(0) parser
(c) CLR(1) parser (d) LR(1) parser

17. Which of the following is not true for SDT?
(a) It is an extension of CFG.
(b) Parsing process is used to do the translation.
(c) It does not permit the subroutines to be attached to the production of a CFG.
(d) It generates the intermediate code.

18. A parse tree with attribute —————— at each node is known as an annotated parse tree.
(a) Name (b) Value
(c) Label (d) None of these

19. Which of the following is true for a dependency graph?
(a) The dependency graph helps to determine how the attribute values are computed.
(b) It depicts the flow of information among the attribute instances in a parse tree.

(c) Both (a) and (b)
(d) None of these

20. An SDD is S-attributed if every attribute is ———————.
(a) Inherited (b) Synthesized
(c) Dependent (d) None of these

21. In L-attributed definitions, the dependency graph edges can go from ——————— to ——————.
(a) Left to right (b) Right to left
(c) Top to bottom (d) Bottom to top

22. Which of the following is not true for an abstract syntax tree?
(a) It is a compressed form of a parse tree.
(b) It represents the syntactic structure of the source program.
(c) The nodes of the tree represent the operands.
(d) None of these

23. Which of the following is not true for syntax-directed translation schemes?
(a) It is a CFG with program fragments embedded within production bodies.
(b) The semantic actions appear at a fixed position within a production body.
(c) They can be considered as a complementary notation to syntax-directed definitions.
(d) None of these

24. Which of the following is not true for the intermediate code?
(a) It can be represented as postfix notation.
(b) It can be represented as syntax tree, and or a DAG.
(c) It can be represented as target code.
(d) It can be represented as three-address code, quadruples, and triples.

25. Which of the following is true for intermediate code generation?
(a) It is machine dependent.
(b) It is nearer to the target machine.
(c) Both (a) and (b)
(d) None of these

26. Which of the following is true in the context of high-level representation of intermediate languages?
(a) It is suitable for static type checking.
(b) It does not depict the natural hierarchical structure of the source program.
(c) It is nearer to the target program.
(d) All of these

27. Which of the following is true for the low-level representation of intermediate languages?
(a) It requires very few efforts by the source program to generate the low-level representation.
(b) It is appropriate for machine-dependent tasks like register allocation and instruction selection.
(c) It does not depict the natural hierarchical structure of the source program.

(d) All of these

28. The reverse polish notation or suffix notation is also known as ——————.

(a) Infix notation (b) Prefix notation

(c) Postfix notation (d) None of above

29. In a two-dimensional array A[i][j], where i is a element of width w1 and j is of width w2, the relative address of A[i][j] can be calculated by the formula ——————.

(a) i * w1 + j * w2 (b) base + i * w1 + j * w2

(c) base + i * w2 + j * w1 (d) base + (i + j) * (w1 + w2)