**REVIEW QUESTIONS**

**1. What are three reasons why syntax analyzers are based on grammars?**

- Using BNF descriptions of the syntax of programs are clear and concise
- Can be used as the direct basis for the syntax analyzer
- Implementations based on BNF are relatively easy to maintain because of modularity

**2. Explain the three reasons why lexical analysis is separated from syntax analysis.**

- Simplicity = techniques for lexical analysis are less complex than those required for syntax analysis
- Efficiency = Although it pays to optimize the lexical analyzer, lexical analysis requires a significant portion of total compilation time
- Portability = the lexical analyzer reads input program files and often includes buffering of that input, it is somewhat platform dependent

**3. Define lexeme and token.**

- Lexeme = logical groupings that the lexical analyzer collects characters into
- Token = internal codes for categories of these groupings

**4. What are the primary tasks of a lexical analyzer?**

- Skipping comments and white space outside lexemes
- Inserts lexemes for user-defined names into the symbol table
- Detects syntactic errors in tokens such as ill-formed floating point literals and reports it to the user

**5. Describe briefly the three approaches to building a lexical analyzer.**

- Write a formal description of the token patterns of the language using a descriptive language related to regular expression
- Design a state transition diagram that describes the token patterns of the language and hand-construct a table-driven implementation of the state diagram
- Design a state transition diagram that describes the token patterns of the language and write a program that implements the diagram

**6. What is a state transition diagram?**

- A directed graph which the representations of a class of mathematical machine that is used for lexical analyzers

**7. Why are character classes used, rather than individual characters, for the letter and digit transitions of a state diagram for a lexical analyzer?**

- 52 characters require 52 transitions = too much

- Define a character class named LETTER for all 52 letters and use a single transition on the first letter of any name

**8. What are the two distinct goals of syntax analysis?**

- To detect syntax errors in a given program
- To produce a parse tree or possibly the information required to build such a tree for a given program

**9. Describe the differences between top-down and bottom-up parsers.**

- Top-down = builds from root downward to leaves
- Bottom-up = builds from the leaves upward to the root

**10. Describe the parsing problem for a top-down parser.**

- to determine whether a given input string can be generated by a given grammar and, if so, to construct a leftmost derivation or parse tree for the input string using a top-down parsing algorithm

**11. Describe the parsing problem for a bottom-up parser.**

- Can only identify and process the low level details of the structure of the text prior to the secondary level and leave the overall structure of the highest level

**12. Explain why compilers use parsing algorithms that work on only a subset of all grammars.**

- Complex and inefficient algorithm for unambiguous grammar

**13. Why are named constants used, rather than numbers, for token codes?**

- Used to simplify the reading of lexical and syntactic analysis

**14. Describe how a recursive-descent parsing subprogram is written for a rule with a single RHS.**

- Each terminal symbol in RHS is compared with nextToken terminal symbol
- If not match = syntax error. If match = lexical analyzer is called to get next input

**15. Explain the two grammar characteristics that prohibit them from being used as the basis for a top-down parser.**

- Left recursion directly or indirectly

**16. What is the FIRST set for a given grammar and sentential form?**

- FIRST( ) = {a => a } (If => , is in FIRST( ))
  in which =>* means 0 or more derivation steps

**17. Describe the pairwise disjointness test.**

- a test of non-left recursive grammar that indicates whether left recursion can be done. It requires the ability to compute a set based on the RHSs of a given nonterminal symbol in a grammar.

**18. What is left factoring?**

- the action taken when a grammar leads backtracking while marking parsing/syntax tree.

**19. What is a phrase of a sentential form?**

- a subsequence of a sentential form that is eventually reduced to a single non-terminal

**20. What is a simple phrase of a sentential form?**

- a phrase that takes a single derivation step from it's root non-terminal node

**21. What is the handle of a sentential form?**

- a substring of a sentential form that can be reduced to a non-terminal symbol by applying a production rule.

**22. What is the mathematical machine on which both top-down and bottomup parsers are based?**

- PDA (PushDown Automation) machine uses both because it recognizes context free language

**23. Describe three advantages of LR parsers.**

- Built for all programming languages
- Detect syntax errors ASAP in a left to right scan
- LR class is a proper superset of the class parsable by LL parsers

**24. What was Knuth's insight in developing the LR parsing technique?**

- to recognize that parsing can be viewed as a process of building a rightmost derivation of a grammar, where the non-terminals are replaced in a right-to-left order.

**25. Describe the purpose of the ACTION table of an LR parser.**

- The ACTION part specifies what the parser should do, given the state symbol on top of the parse stack and the next token of input.

**26. Describe the purpose of the GOTO table of an LR parser.**

- The rows of the GOTO part of the LR parsing table have state symbols as labels. This part of the table has nonterminals as column labels.

The values in the GOTO part of the table indicate which state symbol should be pushed onto the parse stack after a reduction has been completed, which means the handle has been removed from the parse stack and the new nonterminal has been pushed onto the parse stack.

**27. Is left recursion a problem for LR parsers?**

- can lead to infinite recursion or left recursion cycles during the parsing process.