1. Name some translator sofwaares.
   Ans.compiler, interpreter , assembler

2. What is a compiler?

   Ans Compiler is a translator that converts source code written in a high-level language into a low-level machine language.

3. **List various types of compilers.**

   There are three types of compilers

   - Single-Pass Compilers
   - Two-Pass Compilers
   - Multipass Compilers

4. What are the two parts of a compiler(complation)

   Ans. Analysis(FRONT END) and Synthesis(BACK END) are the two parts of compilation.

   Analysis phase

   - Lexical analysis
   - Syntax analysis
   - Semantic analysis
   - Intermediate code generation

   Synthesis phase

   - Code optimization
   - Code generation

5. What tools are used for writing (constructing) compilers in Python?
   - Scanner Generator E.g. **Lex** tool
   - Parser Generator e.g **YAAC**
   - Data-flow analysis engines
   - Automatic code generators
   - Compiler construction toolkits
   - Syntax-directed translation engines

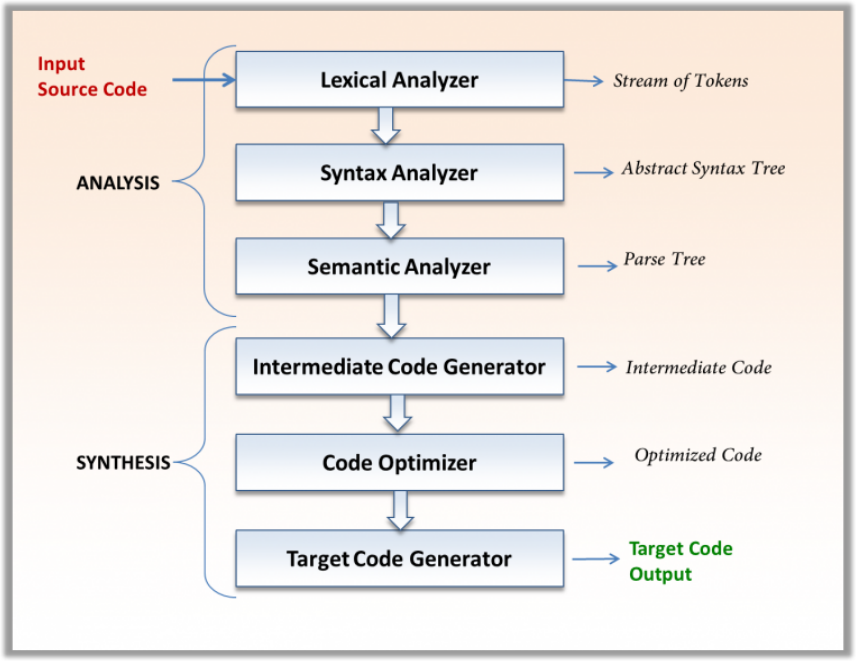
6. **What is LEx and YACC**

   Ans.Lex is a ***lexical analysis tool*** used to identify tokens

   YACC(Yet Another Compiler-Compiler) is ***parser generator***(syntax analyzer).

It determines whether or not the source code for a program is **valid** by the syntactic rules . YACC is developed by Steve Johnson. Yacc is used to **give some structure to those tokens.**

7. Phases of compiler



I**Lexical Analysis:** This phase is also known as scanning. The source code is broken down into a sequence of tokens.

- **Syntax Analysis:** This phase checks the grammar of the code. A parser reads the token generated by the scanner and builds an abstract syntax tree.

- **Semantic Analysis:** It is also known as context analysis. The semantic analyzer reads the AST produced during syntax analysis. Then it performs type checking, scope analysis, and other error.

- **Code Optimization:** This involves improving the final code, which is done by reducing file size, increasing speed, reducing power consumption, etc.

- **Code Generation:** It involves generating machine code from the intermediate after all the phases.

8. **Differentiate Tokens, Patterns, and Lexeme.**

- Tokens: Tokens are character sequences that have significance when taken together as a whole.The fundamental units of any language are called tokens.
- Patterns: are set of rules to determine tokens. .
- Lexeme: It is a string of characters in the source code used to determine whether or not a token should be granted access.
- Eg statement

  int a=10;

Lexeme	Token	Pattern
int	keyword	int
a	identifier	letter(letter| digit)*
=	operator	=
10	constant	digit+
;	Special symbol semicolon	;

9. **What is a Symbol Table?**

Ans. A symbol table is a database in which each identifier is represented by a record that includes fields for the identifier's attributes.

10. What is the use of intermediate code generation

- It is Machine Independent. It can be executed on different platforms.
- It creates the function of code optimization easy.
- It can perform efficient code generation.

- From the existing front end, a new compiler for a given back end can be generated.

11. Mention some cousins of Compiler?

    aPreprocessors ,Assemblers Loaders

12. regular expression for an identifier?

    letter (letter| digit)*

13. What is Lexical analysis?

    Lexical analysis is the first phase of a compilation that is used to convert the input from a simple sequence of characters into a list of tokens of different kinds, such as numerical and string constants, variable identifiers, and programming language keywords.

14. Define parser.

    syntax analysis (Hierarchical analysis) or parsing is one in which the tokens are grouped hierarchically into nested collections with collective meaning.

15. . Mention the basic issues in parsing. There are two important issues in parsing.

    1. Specification of syntax

    2. Representation of input after parsing

16. What does a semantic analysis do?

    Semantic analysis is one in which certain checks are performed to ensure that components of a program fit together meaningfully.

    Mainly performs **type checking.**

17. List the various error recovery strategies
    - Deleting an extraneous character
    - Inserting a missing character
    - Replacing an incorrect character by a correct character
    - Transposing two adjacent charactersCompiler
18. What is bootstrapping in compiler design?

    In compiler design, Bootstrapping is a process in which simple language is used to translate more complicated programs. This complicated program can further handle even more complicated programs and so on.

Bootstrapping is the process of **writing a compiler for a programming language using the language itself**

19. What is cross compiler

    A cross compiler is a compiler that *runs on one machine* (platform) but capable of *creating executable code for another platform (Machine)*

    For example, a compiler that runs on a PC but generates code that runs on an Android smartphone is a cross compiler.

20. Which is a process of finding a parse tree for a string of tokens.?

    Parsing

21. What is ambiguous grammar

    Some strings in this grammar has more than one derivation , more than one parse tree

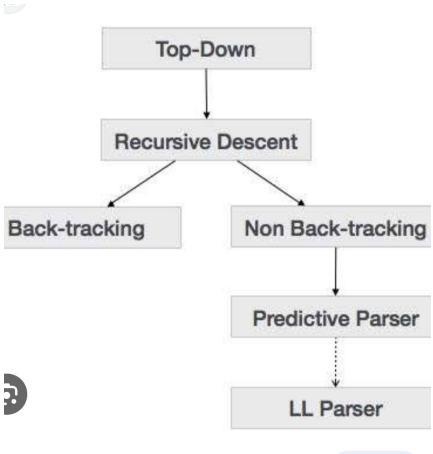
22. Powerful parser

    Canonical LR(CLR)

23. What is top down parsing? Examples?

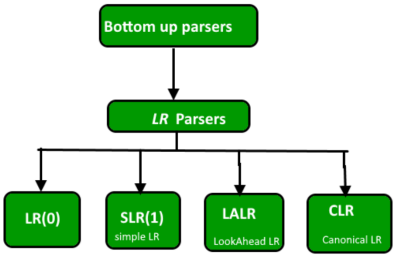
    Recursive Descent Parsing –

    Predictive Parser



24. What is bottom up parsing? Examples?

Shift reduce parsing



25. What is the input and output of the lexical analyzer

    Input Source program

    Output Tokens

26. What is the compiler called which runs on one machine and produces code for a different machine?

    Cross compiler

27. What is used for grouping characters into tokens.

    scanner (lexical analyzer)

28. What are the actions in shift reduce parsing

    shift reduce accept error

29.
30.

31. The basic limitation of finite automata is that
    Ans. It can't remember arbitrary large amount of information.
32. NFA, in its name has 'non-deterministic' because of
    Ans. The choice of path is non-deterministic
33. Number of states require to simulate a computer with memory capable of storing '3' words each of length '8'.

Ans. $2^{(3*8)}$

34.
35. . . .
36.