1. What is a compiler?

- A) A program that converts machine language to high-level language
- B) A program that translates high-level language to machine language
- C) A program that checks spelling errors in source code
- D) None of the above
  Answer: B

---

2. What is the primary function of a translator?

- A) To optimize source code
- B) To convert one language to another
- C) To execute the program directly
- D) To generate syntax trees
  Answer: B

---

3. Which of the following is not a type of translator?

- A) Compiler
- B) Interpreter
- C) Assembler
- D) Emulator
  Answer: D

---

4. Which of these translates line by line?

- A) Compiler
- B) Interpreter
- C) Assembler
- D) Preprocessor
  Answer: B

---

5. What does an assembler do?

- A) Converts machine code to assembly code
- B) Converts assembly code to machine code
- C) Converts high-level language to assembly code
- D) None of the above
  Answer: B

---

6. What is a high-level language?

- A) Machine-understandable language
- B) User-friendly language
- C) Assembly-level language
- D) Language used in hardware circuits
  Answer: B

---

7. Which language uses mnemonics?

- A) High-level language
- B) Assembly language
- C) Machine language
- D) Middle-level language
  Answer: B

---

8. What is the output of the preprocessor?

- A) Object code
- B) Source code
- C) Machine code
- D) Relocatable machine code
  Answer: B

---

9. Which compiler phase generates tokens?

- A) Semantic analysis
- B) Syntax analysis
- C) Lexical analysis
- D) Code generation
  Answer: C

---

10. Which phase ensures type checking?

- A) Syntax analysis
- B) Semantic analysis
- C) Lexical analysis
- D) Code optimization
  Answer: B

---

11. What is intermediate code?

- A) Machine-independent code
- B) Machine-specific code
- C) High-level language code
- D) None of the above
  Answer: A

---

12. Which phase involves creating a syntax tree?

- A) Semantic analysis
- B) Syntax analysis
- C) Lexical analysis
- D) Code optimization
  Answer: B

---

13. What is three-address code?

- A) High-level language code
- B) Assembly language code
- C) Intermediate representation code
- D) Machine code
  Answer: C

---

14. What is the purpose of code optimization?

- A) To reduce memory and execution time
- B) To detect and fix errors
- C) To generate syntax trees
- D) To improve lexical analysis
  Answer: A

---

15. Which phase produces the final machine code?

- A) Code optimization
- B) Intermediate code generation
- C) Code generation
- D) Syntax analysis
  Answer: C

---

16. What is the role of the linker?

- A) To detect errors in the source code
- B) To link machine code with libraries
- C) To convert assembly code to machine code
- D) To optimize the intermediate code
  Answer: B

---

17. Which loader type modifies relocatable machine code?

- A) Absolute loader
- B) Relocating loader
- C) Compile-and-go loader
- D) None of the above
  Answer: B

---

18. What does static linking do?

- A) Links at runtime
- B) Links during program execution
- C) Links at compile time
- D) None of the above
  Answer: C

---

19. What is the smallest meaningful unit in lexical analysis?

- A) Token
- B) Syntax tree
- C) Intermediate code
- D) Bytecode
  Answer: A

---

20. Which of the following is machine-understandable?

- A) High-level language
- B) Assembly language
- C) Machine language
- D) Middle-level language
  Answer: C

---

21. What type of errors does the semantic analysis detect?

- A) Syntactical errors
- B) Logical errors
- C) Type compatibility errors
- D) Compiler errors
  Answer: C

---

22. Which is an example of a high-level language?

- A) Assembly
- B) C++
- C) Binary code
- D) Mnemonics
  Answer: B

---

23. What is the input for an assembler?

- A) High-level language code
- B) Assembly code
- C) Machine code
- D) Intermediate code
  Answer: B

---

24. Which type of compiler translates directly to machine code?

- A) Cross-compiler
- B) Native compiler
- C) Incremental compiler
- D) JIT compiler
  Answer: B

---

25. Which compiler type generates machine code for another architecture?

- A) Native compiler
- B) Cross-compiler
- C) Bootstrap compiler
- D) Source-to-source compiler
  Answer: B

---

26. Which phase produces a symbol table?

- A) Lexical analysis
- B) Syntax analysis
- C) Semantic analysis
- D) All of the above
  Answer: A

---

27. Which of these uses macro processing?

- A) Preprocessor
- B) Linker
- C) Loader
- D) Code generator
  Answer: A

---

28. Which of the following statements is correct?

- A) Interpreters create object code.
- B) Assemblers generate high-level code.
- C) Compilers are faster than interpreters in execution.
- D) Preprocessors execute code.
  Answer: C

---

29. What is the key feature of dynamic linking?

- A) Linking at compile time
- B) Linking during execution
- C) Linking before preprocessing
- D) None of the above
  Answer: B

---

30. Which translator produces an error report line by line?

- A) Compiler
- B) Interpreter
- C) Assembler
- D) Preprocessor
  Answer: B

Unit 2

1. **What is the main purpose of a lexical analyser?**
   A) Parse the source code
   B) Generate assembly code
   C) Identify tokens in the source code
   D) Optimize the code
   Answer: C
2. **What does the lexical analyser output?**
   A) Abstract Syntax Tree
   B) Tokens
   C) Machine code
   D) Error log
   Answer: B
3. **Which of the following tasks is NOT performed by the lexical analyser?**
   A) Removing comments
   B) Syntax validation
   C) Generating tokens
   D) Handling errors
   Answer: B
4. **Lexical errors are often resolved using which method?**
   A) Syntax Parsing
   B) Panic Mode Recovery
   C) Semantic Analysis
   D) Context-Free Grammar
   Answer: B

5. **What is a lexeme?**
   A) A rule to define tokens
   B) A sequence of characters that matches a token pattern
   C) A symbol in the source code
   D) A type of error in lexical analysis
   Answer: B
6. **Which of the following is NOT a type of token?**
   A) Identifier
   B) Constant
   C) Comment
   D) Delimiter
   Answer: C
7. **What is a pattern in lexical analysis?**
   A) A sequence of tokens
   B) A rule defining a set of lexemes
   C) A source code segment
   D) An error-detection technique
   Answer: B

Input Buffering

8. **What is the purpose of input buffering?**
   A) Optimize source code
   B) Store intermediate results
   C) Reduce disk I/O operations
   D) Parse the program faster
   Answer: C
9. **In buffer pairs, how many pointers are used?**
   A) 1
   B) 2
   C) 3
   D) 4
   Answer: B
10. **Which technique uses a special character as a sentinel?**
    A) Buffer Pairs
    B) Single Buffering
    C) Sentinel Buffers
    D) Look-Ahead Buffers
    Answer: C

Specification and Recognition of Tokens

11. **A language is defined as a set of which of the following?**
    A) Tokens
    B) Characters
    C) Strings
    D) Lexemes
    Answer: C
12. **What is Kleene Closure in regular expressions?**
    A) One occurrence of a string
    B) Zero or more occurrences of a string
    C) Zero or one occurrence of a string
    D) Concatenation of two strings
    Answer: B

Regular Expression and Regular Definition

13. **What does the regular expression `a+` mean?**
    A) One or more occurrences of `a`
    B) Zero or more occurrences of `a`
    C) Zero or one occurrence of `a`
    D) Exactly one occurrence of `a`
    Answer: A

14. **Which regular expression represents all binary strings?**
    A) (0|1)*
    B) (0|1)+
    C) 0|1
    D) (0|1)?
    Answer: A
15. **What is the precedence of the Kleene star (*) operator?**
    A) Highest
    B) Lowest
    C) Same as Concatenation
    D) Undefined
    Answer: A

Finite Automata

16. **Which automaton can accept regular languages?**
    A) Pushdown Automaton
    B) Deterministic Finite Automaton (DFA)
    C) Turing Machine
    D) Linear Bounded Automaton
    Answer: B
17. **How does an NFA differ from a DFA?**
    A) DFA allows ε-transitions; NFA does not
    B) NFA allows ε-transitions; DFA does not
    C) Both are equivalent in computational power
    D) NFA is more powerful than DFA
    Answer: B, C

Regular Expression to NFA

18. **Which method is used to convert a regular expression to an NFA?**
    A) Subset Construction
    B) Thompson's Construction
    C) DFA Minimization
    D) Grammar Reduction
    Answer: B
19. **What is the initial state of an NFA constructed from a regular expression?**
    A) Always an accepting state
    B) Has transitions for every symbol
    C) Contains ε-transitions
    D) None of the above
    Answer: C

NFA to DFA Conversion

20. **Which method is used for NFA to DFA conversion?**
    A) Subset Construction
    B) DFA Minimization
    C) State Elimination
    D) Reverse Substitution
    Answer: A
21. **What does each state of the resulting DFA represent after subset construction?**
    A) A single NFA state
    B) A set of NFA states
    C) A lexeme
    D) A symbol
    Answer: B

DFA Optimization

22. **Why is DFA minimization performed?**
    A) To improve accuracy
    B) To reduce the number of states
    C) To handle more languages
    D) To convert it into an NFA
    Answer: B
23. **What is used to identify equivalent states during DFA minimization?**
    A) State Partitioning
    B) Transition Table Analysis
    C) ε-Closure
    D) Regular Expressions
    Answer: A

Miscellaneous

24. **Which of the following is NOT part of lexical analysis?**
    A) Removing white spaces
    B) Generating parse trees
    C) Identifying tokens
    D) Error handling
    Answer: B
25. **What does the statement `a|b` represent in a regular expression?**
    A) Either `a` or `b`
    B) Both `a` and `b`
    C) Only `a`
    D) Only `b`
    Answer: A

26. **Which of the following operations is NOT valid on regular languages?**
    A) Union
    B) Intersection
    C) Complementation
    D) Exponentiation
    Answer: D
27. **What does ε represent in regular expressions?**
    A) A space
    B) A single character
    C) An empty string
    D) None of the above
    Answer: C
28. **What is the primary limitation of regular expressions?**
    A) They cannot represent infinite languages
    B) They cannot handle nested constructs
    C) They are not deterministic
    D) They require DFA for conversion
    Answer: B
29. **Which technique is faster for token recognition?**
    A) NFA
    B) DFA
    C) Regular Expressions
    D) Grammar Parsing
    Answer: B
30. **What is the role of the "forward" pointer in buffering?**
    A) Mark the end of the buffer
    B) Track the beginning of a lexeme
    C) Indicate the next character to read
    D) Store the buffer content
    Answer: C

Set 1: Understanding States and Transitions

1. **What is the initial ε-closure of state 000 in the NFA for (a|b)∗abb(a|b)*abb(a|b)∗abb?**
    - (A) {0,1,2,4,7}\{0,1,2,4,7\}{0,1,2,4,7}
    - (B) {0,1,2,3}\{0,1,2,3\}{0,1,2,3}
    - (C) {0}\{0\}{0}
    - (D) {1,2,4,7}\{1,2,4,7\}{1,2,4,7}
    - **Answer:** (A)
2. **After processing input 'a' from state A={0,1,2,4,7}A = \{0,1,2,4,7\}A={0,1,2,4,7}, what is the resulting set of states?**
    - (A) {3,8}\{3,8\}{3,8}
    - (B) {5}\{5\}{5}
    - (C) {1,2,3,4,6,7,8}\{1,2,3,4,6,7,8\}{1,2,3,4,6,7,8}
    - (D) {0}\{0\}{0}
    - **Answer:** (A)
    -
3. **What does the ε-closure of {3,8}\{3,8\}{3,8} result in?**
    - (A) {3,6,7,1,2,4,8}\{3,6,7,1,2,4,8\}{3,6,7,1,2,4,8}

- (B) {5,6,7}\{5,6,7\}{5,6,7}
- (C) {8}\{8\}{8}
- (D) {3,7,9}\{3,7,9\}{3,7,9}
- **Answer:** (A)

4. **Which state set is labeled as BBB during the construction?**
   - (A) {1,2,3,4,6,7,8}\{1,2,3,4,6,7,8\}{1,2,3,4,6,7,8}
   - (B) {5,6,7,9}\{5,6,7,9\}{5,6,7,9}
   - (C) {3,8}\{3,8\}{3,8}
   - (D) {0,1,2,4,7}\{0,1,2,4,7\}{0,1,2,4,7}
   - **Answer:** (A)
5. **What is the result of Move(A,b)\text{Move}(A, b)Move(A,b), where A={0,1,2,4,7} A = \{0,1,2,4,7\}A={0,1,2,4,7}?**
   - (A) {5}\{5\}{5}
   - (B) {1,2,4}\{1,2,4\}{1,2,4}
   - (C) {8}\{8\}{8}
   - (D) {9,10}\{9,10\}{9,10}
   - **Answer:** (A)

Set 2: Closure Operations

6. **The ε-closure of Move(A,b)={5}\text{Move}(A, b) = \{5\}Move(A,b)={5} is:**
   - (A) {5,6,7,1,2,4}\{5,6,7,1,2,4\}{5,6,7,1,2,4}
   - (B) {1,2,4}\{1,2,4\}{1,2,4}
   - (C) {9,10}\{9,10\}{9,10}
   - (D) {6,7}\{6,7\}{6,7}
   - **Answer:** (A)
7. **State CCC is defined as:**
   - (A) {1,2,4,5,6,7}\{1,2,4,5,6,7\}{1,2,4,5,6,7}
   - (B) {3,6,7}\{3,6,7\}{3,6,7}
   - (C) {0,1,2,4,7}\{0,1,2,4,7\}{0,1,2,4,7}
   - (D) {1,3,8}\{1,3,8\}{1,3,8}
   - **Answer:** (A)
8. **When transitioning from CCC on input 'a', what is the next set of states?**
   - (A) {3,8}\{3,8\}{3,8}
   - (B) {1,3,8}\{1,3,8\}{1,3,8}
   - (C) {7,9,10}\{7,9,10\}{7,9,10}
   - (D) {6,7}\{6,7\}{6,7}
   - **Answer:** (A)
9. **The ε-closure of Move(C,b)={5}\text{Move}(C, b) = \{5\}Move(C,b)={5} results in:**
   - (A) {5,6,7,1,2,4}\{5,6,7,1,2,4\}{5,6,7,1,2,4}
   - (B) {5,7,9}\{5,7,9\}{5,7,9}
   - (C) {7,8,9}\{7,8,9\}{7,8,9}
   - (D) {0,3,8}\{0,3,8\}{0,3,8}
   - **Answer:** (A)
10. **How many states are needed in the DFA equivalent of the NFA for (a|b)∗abb(a|b)*abb(a|b)∗abb?**
    - (A) 4
    - (B) 5
    - (C) 6

- (D) 7
- **Answer:** (B)

Set 3: DFA Construction

11. **State DDD is defined as:**
    - (A) {1,2,4,5,6,7,9}\{1,2,4,5,6,7,9\}{1,2,4,5,6,7,9}
    - (B) {6,8,9}\{6,8,9\}{6,8,9}
    - (C) {1,3,7}\{1,3,7\}{1,3,7}
    - (D) {9,10}\{9,10\}{9,10}
    - **Answer:** (A)
12. **Which set represents the final accepting state in the DFA?**
    - (A) {9,10}\{9,10\}{9,10}
    - (B) {8,9}\{8,9\}{8,9}
    - (C) {7,9}\{7,9\}{7,9}
    - (D) {9}\{9\}{9}
    - **Answer:** (D)
13. **On input 'b' from state BBB, the resulting state is:**
    - (A) D={5,6,7,9}D = \{5,6,7,9\}D={5,6,7,9}
    - (B) C={1,2,4,5,6,7}C = \{1,2,4,5,6,7\}C={1,2,4,5,6,7}
    - (C) {8,9}\{8,9\}{8,9}
    - (D) B={1,2,3,4,6,7,8}B = \{1,2,3,4,6,7,8\}B={1,2,3,4,6,7,8}
    - **Answer:** (A)
14. **The transition Move(D,a)\text{Move}(D, a)Move(D,a) leads to:**
    - (A) {3,8}\{3,8\}{3,8}
    - (B) {5,7,9}\{5,7,9\}{5,7,9}
    - (C) {8,9}\{8,9\}{8,9}
    - (D) B={1,2,3,4,6,7,8}B = \{1,2,3,4,6,7,8\}B={1,2,3,4,6,7,8}
    - **Answer:** (A)
15. **How many transitions are there in the final DFA?**
    - (A) 8
    - (B) 9
    - (C) 10
    - (D) 11
    - **Answer:** (B)

1. What is the initial state of a DFA constructed from an NFA?
    - (A) All accepting states of the NFA
    - (B) ε-closure of the NFA's initial state
    - (C) A single state
    - (D) None of the above

    Answer: (B)
2. In the subset construction algorithm, how many states will a DFA have at most if the NFA has nnn states?
    - (A) 2n2^n2n
    - (B) n2n^2n2
    - (C) nnn
    - (D) n+1n+1n+1

    Answer: (A)

3. What does the ε-closure of a state in an NFA represent?
   - (A) All states reachable by one ε transition
   - (B) All states reachable by zero or more ε transitions
   - (C) Only the starting state
   - (D) None of the above

   Answer: (B)
4. How are the final states of the DFA determined when constructing from an NFA?
   - (A) Any state containing at least one NFA final state in its subset
   - (B) Only the NFA's final state
   - (C) Union of all NFA states
   - (D) All reachable states

   Answer: (A)
5. What is the correct ε-closure of state q0q_0q0 if q0q_0q0 has ε-transitions to q1q_1q1 and q2q_2q2?
   - (A) {q0,q1,q2}\{q_0, q_1, q_2\}{q0,q1,q2}
   - (B) {q1,q2}\{q_1, q_2\}{q1,q2}
   - (C) {q0}\{q_0\}{q0}
   - (D) None of the above

   Answer: (A)
6. Which of the following is **true** about deterministic finite automata (DFA)?
   - (A) A DFA may have multiple transitions for the same input symbol
   - (B) A DFA can have ε-transitions
   - (C) A DFA must have exactly one transition for each symbol from every state
   - (D) None of the above

   Answer: (C)
7. In the DFA A={0,1,2,4,7}A = \{0, 1, 2, 4, 7\}A={0,1,2,4,7}, what does Move(A,b)Move(A, b)Move(A,b) represent?
   - (A) States reachable from AAA with ε-transitions
   - (B) States reachable from AAA with input bbb
   - (C) States reachable from AAA with input aaa
   - (D) None of the above

   Answer: (B)
8. Which of the following is **true** for a DFA?
   - (A) It can represent any NFA
   - (B) It is always equivalent to an NFA
   - (C) DFA states can be a power set of NFA states
   - (D) All of the above

   Answer: (D)
9. How is the DFA transition table constructed?
   - (A) Using ε-closures and subset construction
   - (B) By directly copying the NFA transitions
   - (C) By converting to a grammar first
   - (D) None of the above

   Answer: (A)
10. What is the time complexity of converting an NFA to a DFA in terms of states?

- (A) O(2n)O(2^n)O(2n)
- (B) O(n2)O(n^2)O(n2)
- (C) O(n)O(n)O(n)
- (D) O(1)O(1)O(1)

Answer: (A)

11. If a DFA has states A={0,1,2,4,7}A = \{0, 1, 2, 4, 7\}A={0,1,2,4,7}, and Move(A,a)={3,8}Move(A, a) = \{3, 8\}Move(A,a)={3,8}, what is the ε-closure of Move(A,a)Move(A, a)Move(A,a)?

- (A) {3,8}\{3, 8\}{3,8}
- (B) {1,2,3,4,6,7,8}\{1, 2, 3, 4, 6, 7, 8\}{1,2,3,4,6,7,8}
- (C) {3,6,7,1,2,4,8}\{3, 6, 7, 1, 2, 4, 8\}{3,6,7,1,2,4,8}
- (D) None of the above
  Answer: (C)

12. If a DFA state B={1,2,4,6,7}B = \{1, 2, 4, 6, 7\}B={1,2,4,6,7} transitions on input bbb to {5,9}\{5, 9\}{5,9}, the next DFA state is:

- (A) {5,9}\{5, 9\}{5,9}
- (B) εεε-closure of {5,9}\{5, 9\}{5,9}
- (C) {1,2,4,5,6,7,9}\{1, 2, 4, 5, 6, 7, 9\}{1,2,4,5,6,7,9}
- (D) None of the above
  Answer: (B)

13. How many distinct states are there in the minimized DFA for the language (a|b)∗abb(a|b)*abb(a|b)∗abb?

- (A) 4
- (B) 3
- (C) 5
- (D) 6
  Answer: (A)

14. In the language (a|b)∗abb(a|b)*abb(a|b)∗abb, what does the DFA final state represent?

- (A) Acceptance of strings ending with abbabbabb
- (B) Acceptance of strings starting with abbabbabb
- (C) Strings containing abbabbabb as a substring
- (D) None of the above
  Answer: (A)

15. Which of the following is not possible with a DFA?

- (A) Recognizing regular languages
- (B) Recognizing context-free languages
- (C) Accepting only finite-length strings
- (D) Having a unique transition for each input
  Answer: (B)

- **What is the primary role of a parser in the compiler model?**

- (A) Translate code into assembly
- (B) Verify syntax and construct parse trees
- (C) Generate machine code
- (D) Optimize the code
  Answer: (B)

- **Which type of error does the parser handle?**

  - (A) Logical errors
  - (B) Syntax errors
  - (C) Runtime errors
  - (D) Lexical errors
    Answer: (B)

- **What does a parse tree represent?**

  - (A) Syntax rules
  - (B) Semantic structure
  - (C) Syntactic structure of tokens
  - (D) Compiler output
    Answer: (C)

- **Which grammar type is the most restrictive?**

  - (A) Type-0
  - (B) Type-1
  - (C) Type-2
  - (D) Type-3
    Answer: (D)

- **Which grammar is known as Context-Free Grammar (CFG)?**

  - (A) Type-0
  - (B) Type-1
  - (C) Type-2
  - (D) Type-3
    Answer: (C)

- **What is the output of leftmost derivation for the string `id + id * id` using the grammar S→S+S|S∗S|idS \rightarrow S + S | S * S | idS→S+S|S∗S|id?**

  - (A) S→id+S∗idS \rightarrow id + S * idS→id+S∗id
  - (B) S→id+id∗idS \rightarrow id + id * idS→id+id∗id
  - (C) S→id+id+idS \rightarrow id + id + idS→id+id+id
  - (D) S→id∗id+idS \rightarrow id * id + idS→id∗id+id
    Answer: (B)

- **In top-down parsing, which issue needs to be eliminated?**

  - (A) Ambiguity
  - (B) Left recursion
  - (C) Semantic errors
  - (D) Lexical errors
    Answer: (B)

- **Which is not a type of left recursion?**
  - (A) Direct
  - (B) Hidden
  - (C) Indirect
  - (D) Associative
    Answer: (D)

- **Which technique is used to address ambiguity in grammar?**
  - (A) Left factoring
  - (B) Recursive descent parsing
  - (C) Semantic analysis
  - (D) Lexical analysis
    Answer: (A)

- **What is the purpose of syntax error handling in parsers?**
  - (A) Debugging semantic errors
  - (B) Detecting logical errors
  - (C) Reporting and recovering from syntax errors
  - (D) Improving code efficiency
    Answer: (C)

- **Which parser uses lookahead to make decisions?**
  - (A) Top-down parser
  - (B) Bottom-up parser
  - (C) Recursive descent parser
  - (D) Predictive parser
    Answer: (D)

- **What is the input to a parser?**
  - (A) Source code
  - (B) Syntax tree
  - (C) Tokens from the lexical analyzer
  - (D) Intermediate code
    Answer: (C)

- **Which LR parser is the most powerful?**
  - (A) LR(0)
  - (B) SLR
  - (C) LALR
  - (D) CLR
    Answer: (D)
  -

- **Which grammar production is not allowed in regular grammar?**
  - (A) A→aBA \rightarrow aBA→aB
  - (B) A→aA \rightarrow aA→a
  - (C) A→BaA \rightarrow BaA→Ba
  - (D) A→ϵA \rightarrow \epsilonA→ϵ
    Answer: (C)

- **Which parser constructs a derivation tree using a stack?**
  - (A) LL Parser
  - (B) LR Parser
  - (C) Recursive Descent Parser
  - (D) Syntax Directed Translator
    Answer: (B)

- **What is the root of a parse tree for a given grammar?**
  - (A) A terminal symbol
  - (B) A non-terminal symbol
  - (C) Start symbol
  - (D) Lexeme
    Answer: (C)

- **Which derivation replaces the leftmost non-terminal first?**
  - (A) Leftmost derivation
  - (B) Rightmost derivation
  - (C) Bottom-up derivation
  - (D) Top-down derivation
    Answer: (A)

- **What is the main disadvantage of ambiguous grammar?**
  - (A) It generates no parse trees
  - (B) It generates multiple parse trees
  - (C) It generates logical errors
  - (D) It generates syntax errors
    Answer: (B)

- **How can left recursion be eliminated?**
  - (A) By removing non-terminals
  - (B) By using rightmost derivation
  - (C) By rewriting the grammar rules
  - (D) By simplifying the lexical analyzer
    Answer: (C)

- **What is the common solution for resolving ambiguities in predictive parsing?**

- (A) Lookahead
- (B) Backtracking
- (C) Left factoring
- (D) LL(1) Parsing
  Answer: (C)

- **What is the role of reduction in parsing?**

  - (A) Replace a non-terminal with a terminal
  - (B) Replace a substring by its production rule
  - (C) Replace the start symbol with terminals
  - (D) Replace errors with correct syntax
    Answer: (B)

- **Which of the following is a bottom-up parsing approach?**

  - (A) LL Parser
  - (B) SLR Parser
  - (C) Recursive Descent Parser
  - (D) Predictive Parser
    Answer: (B)

- **What is Left Factoring?**

  - (A) Removing ambiguity from grammar
  - (B) Eliminating left recursion
  - (C) Refactoring production rules to common prefixes
  - (D) Optimizing parse trees
    Answer: (C)

- **How many types of LR parsers exist?**

  - (A) Two
  - (B) Three
  - (C) Four
  - (D) Five
    Answer: (C)

- **What is a distinguishing feature of LALR parsers?**

  - (A) Less memory usage than CLR parsers
  - (B) Handles ambiguous grammar efficiently
  - (C) Constructs parse trees recursively
  - (D) Generates fewer conflicts than SLR parsers
    Answer: (A)

- **What is the primary limitation of Type-0 Grammar?**

  - (A) It is ambiguous

- (B) It is too general for programming languages
- (C) It cannot generate infinite strings
- (D) It lacks context sensitivity
  Answer: (B)

- **Which method allows a grammar to be predictive?**

  - (A) Left recursion
  - (B) Recursive descent
  - (C) Left factoring
  - (D) Ambiguity removal
    Answer: (C)

- **What type of derivation does an LL parser use?**

  - (A) Leftmost derivation
  - (B) Rightmost derivation
  - (C) Reverse derivation
  - (D) Mixed derivation
    Answer: (A)

- **What is the main difference between parse tree and syntax tree?**

  - (A) Parse tree represents precedence clearly
  - (B) Syntax tree eliminates unnecessary nodes
  - (C) Parse tree is generated after the syntax tree
  - (D) Syntax tree contains non-terminal nodes only
    Answer: (B)

- **What does an LR(0) parser lack compared to LALR?**

  - (A) Lookahead capability
  - (B) Efficient error recovery
  - (C) Handling of large grammars
  - (D) Grammar simplification
    Answer: (A)

1. **What does LL(1) stand for in LL(1) parsing?**
   a) Left-to-right scanning, Leftmost derivation, and 1 lookahead symbol
   b) Left-to-right scanning, Rightmost derivation, and 1 lookahead symbol
   c) Right-to-left scanning, Leftmost derivation, and 1 lookahead symbol
   d) Right-to-left scanning, Rightmost derivation, and 1 lookahead symbol
   Answer: a) Left-to-right scanning, Leftmost derivation, and 1 lookahead symbol
2. **Which data structures are commonly used by an LL(1) parser?**
   a) Tree and Stack
   b) Input buffer, Stack, and Parsing Table
   c) Queue and Tree
   d) List and Queue
   Answer: b) Input buffer, Stack, and Parsing Table

3. **What is the purpose of a predictive parsing table?**
   a) To parse regular expressions
   b) To predict the next token in the input
   c) To determine the appropriate grammar rule based on lookahead
   d) To optimize memory usage during parsing
   Answer: c) To determine the appropriate grammar rule based on lookahead
4. **Which step is NOT required when constructing an LL(1) parser?**
   a) Compute FIRST and FOLLOW sets
   b) Remove left recursion
   c) Create an operator precedence table
   d) Construct the predictive parsing table
   Answer: c) Create an operator precedence table
5. **What does the FOLLOW set of a non-terminal represent?**
   a) All terminals that can appear immediately after the non-terminal
   b) All terminals that can appear before the non-terminal
   c) All terminals in the grammar
   d) The last terminal in the production rule
   Answer: a) All terminals that can appear immediately after the non-terminal
6. **In LL(1) parsing, what does the '1' in LL(1) indicate?**
   a) One non-terminal in the grammar
   b) One lookahead symbol
   c) One production rule per grammar
   d) One step to derive the string
   Answer: b) One lookahead symbol
7. **How does an LL(1) parser handle an error during parsing?**
   a) It skips the current input symbol
   b) It halts parsing and throws an error
   c) It calls an error recovery routine
   d) It continues parsing with the next production
   Answer: c) It calls an error recovery routine
8. **Which of the following is a handle in a parse tree?**
   a) The root of the tree
   b) A substring that matches the RHS of a production rule
   c) A terminal symbol
   d) The leftmost non-terminal in the tree
   Answer: b) A substring that matches the RHS of a production rule
9. **What is the first step in constructing an LL(1) parser?**
   a) Compute the parsing table
   b) Compute FOLLOW sets
   c) Remove left recursion from the grammar
   d) Parse the input string
   Answer: c) Remove left recursion from the grammar
10. **What action does an LL(1) parser take if X = a ≠ $?**
    a) Halts parsing
    b) Pops X off the stack and advances the input pointer
    c) Calls the error recovery routine
    d) Pushes a onto the stack
    Answer: b) Pops X off the stack and advances the input pointer

11. **What is "handle pruning"?**
    a) Identifying and expanding a handle in the parse tree
    b) Identifying and reducing a handle to the left-hand side of a production
    c) Optimizing the input buffer for parsing
    d) Removing unnecessary symbols from the grammar
    Answer: b) Identifying and reducing a handle to the left-hand side of a production
12. **What is the main feature of operator precedence parsing?**
    a) No two adjacent terminals in a production
    b) Use of FIRST and FOLLOW sets
    c) Ability to handle left recursion directly
    d) Dependence on predictive parsing tables
    Answer: a) No two adjacent terminals in a production
13. **What does the relation "a <· b" indicate in operator precedence parsing?**
    a) a has lower precedence than b
    b) a has the same precedence as b
    c) a has higher precedence than b
    d) a is equivalent to b
    Answer: a) a has lower precedence than b
14. **Which of these parsers uses a stack to store partial parse trees?**
    a) Operator precedence parser
    b) Shift-reduce parser
    c) Recursive descent parser
    d) CYK parser
    Answer: b) Shift-reduce parser
15. **What is the key advantage of LL(1) parsing?**
    a) Supports ambiguous grammars
    b) Simple and efficient table-driven parsing
    c) Does not require a lookahead symbol
    d) Directly supports operator precedence parsing
    Answer: b) Simple and efficient table-driven parsing
16. **What is the starting stack symbol in an LL(1) parser?**
    a) The start symbol of the grammar
    b) The first terminal in the input string
    c) $ (end-of-input marker)
    d) The first non-terminal in the parsing table
    Answer: c) $ (end-of-input marker)
17. **What is the relationship between FIRST and FOLLOW sets in grammar analysis?**
    a) FIRST is derived from FOLLOW
    b) FOLLOW is derived from FIRST
    c) Both are computed independently
    d) FOLLOW is the inverse of FIRST
    Answer: c) Both are computed independently
18. **Which action in shift-reduce parsing moves a terminal from the input buffer to the stack?**
    a) Shift
    b) Reduce
    c) Accept
    d) Error
    Answer: a) Shift

19. **What happens when the parser stack contains only the start symbol, and the input is empty?**
    a) The parser shifts the start symbol
    b) The parser reduces the start symbol
    c) The parser accepts the input
    d) The parser throws an error
    Answer: c) The parser accepts the input
20. **In a grammar E → E + E, what type of recursion is present?**
    a) Left recursion
    b) Right recursion
    c) Indirect recursion
    d) No recursion
    Answer: a) Left recursion

unit 4

21. • **What is the main role of the intermediate code generator in a compiler?**

- a) Translate source code directly into machine code
- b) Generate target code
- c) Translate source code into an intermediate representation
- d) Perform lexical analysis
  Answer: c

- **Which of the following is an advantage of using an intermediate representation in compilers?**

- a) Easier debugging
- b) Facilitates retargeting
- c) Improves source code readability
- d) Direct execution of the source program
  Answer: b

- **Which of the following is NOT a type of intermediate representation?**

- a) Abstract Syntax Tree
- b) Postfix Notation
- c) Three Address Code
- d) Machine Code
  Answer: d

- **What structure is used to identify common subexpressions in intermediate code?**

- a) Binary Tree
- b) Directed Acyclic Graph (DAG)
- c) Hash Table
- d) Linked List
  Answer: b

- **In postfix notation, how is the expression `a + b * c` represented?**

  - a) a + b * c
  - b) abc*+
  - c) ab+c*
  - d) a*bc+

  Answer: b

- **Which of the following is an example of Three Address Code?**

  - a) x := y op z
  - b) x := y + z * w
  - c) x + y + z
  - d) a := b + c + d

  Answer: a

- **What does the "Three" in Three Address Code refer to?**

  - a) The number of instructions
  - b) The number of operands in each statement
  - c) The number of temporary variables used
  - d) The number of address fields per instruction

  Answer: d

- **Which of the following represents a temporary value using the position of the statement that computes it?**

  - a) Quadruples
  - b) Triples
  - c) Indirect Triples
  - d) None of the above

  Answer: b

- **In a quadruple, the result of the statement `x := y op z` is stored in which field?**

  - a) op
  - b) arg1
  - c) arg2
  - d) result

  Answer: d

- **Which representation avoids storing temporary names in the symbol table?**

  - a) Quadruples
  - b) Triples
  - c) Indirect Triples
  - d) Abstract Syntax Tree

  Answer: b

- **What is an indirect triple?**

  - a) A representation that uses pointers to triples
  - b) A direct representation of syntax tree
  - c) A hybrid of Quadruples and Triples
  - d) None of the above
    Answer: a

- **A Directed Acyclic Graph (DAG) does not contain:**

  - a) Nodes
  - b) Edges
  - c) Cycles
  - d) Values
    Answer: c

- **In Syntax-Directed Definitions, what are attributes?**

  - a) Variables in source code
  - b) Semantic information associated with grammar symbols
  - c) Operators used in intermediate code
  - d) Rules for parsing
    Answer: b

- **A syntax-directed definition using only synthesized attributes is known as:**

  - a) Inherited Attribute Definition
  - b) L-attributed Definition
  - c) S-attributed Definition
  - d) Context-Free Grammar
    Answer: c

- **What type of attribute propagates information from parent to child in a parse tree?**

  - a) Synthesized
  - b) Inherited
  - c) Context-Free
  - d) Syntax-Directed
    Answer: b

- **Which attribute is used to pass information bottom-up in the parse tree?**

  - a) Synthesized
  - b) Inherited
  - c) Both a and b
  - d) None of the above
    Answer: a

- **Which of the following attributes is used only in L-attributed Syntax-Directed Definitions?**

  - a) Synthesized
  - b) Inherited
  - c) Both Synthesized and Inherited
  - d) None of the above
    Answer: b

- **Which of the following structures is used for parameter passing during runtime?**

  - a) Quadruples
  - b) DAG
  - c) Activation Record
  - d) Postfix Notation
    Answer: c

- **In an activation record, where are local variables stored?**

  - a) Heap
  - b) Stack
  - c) Global Memory
  - d) Symbol Table
    Answer: b

- **What is the purpose of an Annotated Parse Tree?**

  - a) To visualize the DAG
  - b) To compute attribute values at each node
  - c) To identify syntax errors
  - d) To optimize memory usage
    Answer: b

- **Which parameter-passing method copies the actual value of arguments?**

  - a) Call by Value
  - b) Call by Reference
  - c) Call by Name
  - d) Call by Copy-restore
    Answer: a

- **What does the `op` field in Quadruples contain?**

  - a) Temporary variable
  - b) Operation type
  - c) Operand name
  - d) Memory location
    Answer: b

- **What is the result of `t1 := uminus c` in a three-address code?**
  - a) The negative of c is stored in t1
  - b) The value of c is added to t1
  - c) The value of c is assigned to t1
  - d) None of the above
    Answer: a

- **In Syntax-Directed Translation, what does `E.val = E1.val + T.val` indicate?**
  - a) E.val inherits its value from E1.val and T.val
  - b) E.val is synthesized from E1.val and T.val
  - c) Both inherited and synthesized attributes are used
  - d) None of the above
    Answer: b

- **Which of the following is an intermediate language?**
  - a) Java Bytecode
  - b) Assembly Language
  - c) Machine Code
  - d) Python
    Answer: a

- **What does a DAG identify in an expression?**
  - a) Temporary variables
  - b) Common subexpressions
  - c) Syntax errors
  - d) Parameter dependencies
    Answer: b

- **In an activation record, the control link points to:**
  - a) Local variables
  - b) The previous activation record
  - c) The current function code
  - d) None of the above
    Answer: b

- **Which type of intermediate code is human-readable and close to assembly language?**
  - a) Postfix Notation
  - b) Abstract Syntax Tree
  - c) Three Address Code
  - d) Machine Code
    Answer: c
  -

- **What does the operator `[]` in Quadruples represent?**

  - a) Function Call
  - b) Array Access
  - c) Pointer Dereference
  - d) Assignment
    Answer: b

- **In parameter passing, what does Call by Reference do?**

  - a) Copies the value of the argument
  - b) Copies the address of the argument
  - c) Does not copy anything
  - d) None of the above
    Answer: b

unit 5

1. What is the primary purpose of code optimization?

A) To change the program output
B) To improve program readability
C) To make code more efficient in terms of time and space
D) To increase code complexity

Answer: C

---

2. Which of the following is NOT an optimization technique?

A) Constant Folding
B) Code Motion
C) Function Inlining
D) Syntax Highlighting

Answer: D

---

3. What does "Common Subexpression Elimination" achieve?

A) Removes duplicate variable names
B) Eliminates repeated computation of the same expression
C) Improves the readability of expressions
D) Eliminates unused variables

Answer: B

---

4. What is "Dead Code Elimination"?

A) Removing unreachable or unused code
B) Replacing inefficient loops
C) Modifying code structure for readability
D) Eliminating recursive function calls

Answer: A

---

5. Which of the following applies to "Peephole Optimization"?

A) Global data flow analysis
B) Loop optimization
C) Optimization of short sequences of code
D) Dead code removal

Answer: C

---

6. In loop optimization, "Code Motion" refers to:

A) Adding new variables inside a loop
B) Moving loop-invariant computations outside the loop
C) Eliminating redundant loops
D) Changing the order of loop instructions

Answer: B

---

7. "Reduction in Strength" replaces:

A) Complex instructions with simpler, faster ones
B) Multiple variables with single variables
C) Function calls with inline code
D) Recursive calls with iterative loops

Answer: A

---

8. Which of these is an example of "Compile Time Evaluation"?

A) A loop executed at runtime
B) Calculating constants during compilation
C) Allocating memory during runtime
D) Optimizing register usage

Answer: B

---

9. What is a "Basic Block"?

A) A sequence of code with multiple entry points
B) A section of code with a single entry and exit point
C) A group of unrelated statements
D) A set of functions

Answer: B

---

10. In "Global Data Flow Analysis," what does "kill[S]" represent?

A) Data generated by a statement
B) Data removed or invalidated by a statement
C) Errors encountered during execution
D) Constants eliminated during compilation

Answer: B

---

11. What is the goal of "Register Allocation"?

A) Assign variables to registers efficiently
B) Remove unnecessary registers
C) Increase the size of machine code
D) Simplify intermediate code

Answer: A

12. Which of these is an example of "Peephole Optimization"?

A) Removing unused variables
B) Eliminating unnecessary jumps
C) Reducing the strength of operators
D) All of the above

Answer: D

---

13. An expression is considered "Busy" when:

A) It is used frequently in a loop
B) It is computed but not used immediately
C) No operands are redefined before its use
D) It involves complex arithmetic

Answer: C

---

14. What type of instruction is optimized using "Machine Idioms"?

A) Arithmetic operations
B) High-level source code
C) Target-specific efficient instructions
D) Error-checking statements

Answer: C

---

15. What are the inputs to a Code Generator?

A) Assembly code
B) Optimized source code
C) Intermediate representation of source code
D) Machine-level code

Answer: C

---

16. Which of the following forms the output of a Code Generator?

A) Source code
B) Intermediate code
C) Executable machine code
D) Compiler errors

Answer: C

---

17. What is the primary objective of "Instruction Selection"?

A) Minimize register usage
B) Select appropriate target-machine instructions
C) Detect syntax errors
D) Allocate memory dynamically

Answer: B

---

18. "Live Variable" analysis is used to:

A) Identify unused variables
B) Determine if a variable's value is used along any path before being redefined
C) Track memory allocation for variables
D) Eliminate loop-invariant computations

Answer: B

---

19. What is an "Available Expression"?

A) An expression available in the symbol table
B) An expression not modified before its use
C) A computed value stored in memory
D) A statement with no side effects

Answer: B

---

20. The term "Instruction Ordering" refers to:

A) Arranging code instructions for optimal performance
B) Removing redundant instructions
C) Changing loop structures
D) Debugging machine code

Answer: A

---

21. What type of language is "Postfix Notation"?

A) High-level language
B) Assembly language
C) Intermediate representation
D) Machine language

Answer: C

22. "Unreachable Code" occurs when:

A) The program enters an infinite loop
B) A section of code cannot be executed
C) Variables are never initialized
D) A jump instruction is omitted

Answer: B

23. "Flow Graph" is used for:

A) Representing function calls
B) Tracking memory usage
C) Visualizing control flow in code
D) Optimizing variable names

Answer: C

24. DAG stands for:

A) Direct Allocation Graph
B) Directed Acyclic Graph
C) Data Allocation Graph
D) Dead Assignment Graph

Answer: B

25. What is the role of "Semantic Analysis" in code generation?

A) Detect syntax errors
B) Ensure type correctness and variable declarations
C) Optimize register allocation
D) Improve loop efficiency

Answer: B

---

26. Which of the following is NOT a property of optimized target code?

A) Correctness
B) High quality
C) Increased code size
D) Efficient resource utilization

Answer: C

27. Which optimization involves replacing "x^2" with "x*x"?

A) Dead code elimination
B) Reduction in strength
C) Constant folding
D) Peephole optimization

Answer: B

28. Register assignment is a sub-problem of:

A) Instruction selection
B) Register allocation
C) Code motion
D) Constant propagation

Answer: B

29. "Redundant Load and Store" optimization eliminates:

A) Unused variables
B) Extra memory access instructions
C) Dead code
D) Infinite loops

Answer: B

30. What does "Absolute Machine Language" represent?

A) Intermediate code
B) Code with fixed memory addresses ready for execution
C) Assembly code
D) Object files

Answer: B