Compiler Design – Unit 2 (Syntax Analysis) – Simplified Exam Notes

All important questions with simple, exam-ready answers.

# 1. What is the role of a parser in compiler design?

The parser checks whether the input program follows the correct syntax of the language. It takes tokens from the lexical analyzer and creates a parse tree or syntax tree. It also reports any syntax errors.

# 2. What is the difference between top-down and bottom-up parsing?

Top-down parsing starts from the start symbol and tries to derive the input string.  
Bottom-up parsing starts from the input string and tries to reach the start symbol by reducing.

# 3. Define top-down parsing with an example.

Top-down parsing creates the parse tree from top (start symbol) to bottom using grammar rules. Example: Recursive Descent Parser.

# 4. What is recursive descent parsing?

It is a top-down parsing method that uses a set of recursive functions to process the input. It works well for LL(1) grammars.

# 5. What is LL(1) parsing?

LL(1) stands for Left-to-right scanning, Leftmost derivation, and 1 symbol lookahead. It is a type of top-down parser that uses a table to predict which rule to apply.

# 6. What are FIRST and FOLLOW sets?

FIRST(X) = Set of terminals that begin strings derivable from X.  
FOLLOW(X) = Set of terminals that can appear immediately to the right of X in some derivation.

# 7. Rules to compute FIRST and FOLLOW sets?

FIRST Rules:  
1. If X is a terminal, FIRST(X) = {X}  
2. If X -> ε, then ε ∈ FIRST(X)  
3. For X -> Y1Y2...Yn, add FIRST(Y1), then Y2, etc.  
FOLLOW Rules:  
1. $ in FOLLOW(Start Symbol)  
2. For A -> aB, add FOLLOW(A) to FOLLOW(B) if B is at end.

# 8. How to construct a predictive parsing table?

Steps:  
1. For each production A -> α, add it to M[A, a] for all a in FIRST(α).  
2. If ε in FIRST(α), add A -> α in M[A, b] for all b in FOLLOW(A).  
3. Set other entries to error.

# 9. What is an LL(1) grammar?

A grammar with no ambiguity or left recursion and only one production per table entry is LL(1). It is suitable for predictive parsing.

# 10. What is left recursion and left factoring?

Left Recursion: When a non-terminal calls itself on the left (A → Aα). Left Factoring: When common prefixes are factored out to make parsing easier.

# 11. What is panic mode error recovery?

When an error is found, parser skips input symbols until it finds a synchronizing token (like ; or }) and continues parsing.

# 12. What is an LR parser?

An LR parser is a bottom-up parser that reads input left-to-right and produces a rightmost derivation in reverse. It uses a stack and parsing tables (ACTION, GOTO).

# 13. Why is LR parsing powerful?

It can handle almost all programming language constructs and detects syntax errors early. It is more powerful than LL(1) parsing.

# 14. What is operator precedence parsing?

This bottom-up parsing method uses precedence relations (<·, =·, ·>) to determine when to shift or reduce without ambiguity.

# 15. What is shift-reduce parsing?

It uses a stack. Symbols are pushed (shifted), and if top of stack matches right-hand side of a rule, it's replaced by left-hand side (reduced).

# 16. What is an action table and goto table?

ACTION table tells whether to shift, reduce, accept or error.  
GOTO table tells which state to go to after a reduction.

# 17. What is SLR parsing?

SLR stands for Simple LR. It uses LR(0) items and FOLLOW sets to construct parsing tables. It's easier but less powerful than full LR.

# 18. What are LR(0) items, closure, and goto functions?

LR(0) items: Productions with a dot showing position. Closure: Expands items. GOTO: Moves dot to next symbol and finds closure again.

# 19. What is canonical LR parsing?

It uses LR(1) items with lookahead symbols for precise parsing. More powerful than SLR, but uses more memory and time.

# 20. What is an LALR parser?

LALR = Look-Ahead LR. It merges LR(1) states with same cores to reduce size. It’s used in most practical compilers (like yacc, bison).

# 21. How does LR parser handle errors?

When no valid action is found in table, LR parser reports error and can use panic-mode or phrase-level recovery to continue parsing.

# 22. What is the dangling else problem?

It happens when an 'else' can match more than one 'if'. Resolved by associating 'else' with nearest unmatched 'if' or using grammar rules to remove ambiguity.