Lexical Analysis
Tokens, patterns, lexemes
Regular expressions
Transition Diagrams
NFA
DFA

Syntax Analysis
Context-Free Grammar
Derivation
Parse-Tree
Writing Grammar
Eliminating Ambiguity
Eliminating Left-Recursion
Left-Factoring

Top-Down Parsing
Recursive-Descent
Predictive-Parsing
LL(1) Parsing
LL(1) Grammar
FIRST and FOLLOW
Predictive Parsing table

Bottom-Up Parsing
Shift-Reduce Parsing
LR(0) Parsing
Constructing LR(0) Automaton
LR Parsing Algorithm
Constructing SLR Parsing Table

LR(1) parsing
LALR

Syntax-Directed Translation
Syntax-Directed Definition
Inherited Attributes
Synthesized Attributes
Syntax-Directed Translation Schemes

Intermediate Code Generation
Three-Address Code
Syntax Tree
Handling: declarations, expressions, control-flow, arrays, …

Run-Time Environment
Storage Organization
Stack: activation records
Heap

Code Generator
For the following regular expression: ((c|a)b*)*

a. Give example of expressions generated by this expression
b. Derive NFA from the regular expression
c. Give the transition table for the NFA
d. Derive a DFA from the NFA
e. Is the resulting DFA optimized? or we can minimize it?
f. Does NFA always need to be larger than DFA?

For the following grammar: S->SS+ | SS* | a

a. Construct the LR(0) automaton (similar to Fig 4.31)
b. From the automaton construct the SLR parsing table (see Fig 4.37)
c. Show the action of your parsing table for the input: aa*a+
d. Construct the GOTO graph of LR(1) sets of items

What is an activation record? Why do we need it?

If a program has X functions/procedures, what is the minimum and maximum number of activation records it can have?

What are basic blocks? Why do we need them?