Question Bank Compiler Design For Unit 3,Unit 4,Unit 5

Very short questions

1.What is a parser generator? Name a few commonly used parser generators.
2.Explain the term "syntax-directed definition" (SDD).
3. Describe the purpose of syntax trees in compiler design.
4. What are the main differences between S-attributed and L-attributed definitions?
5. Define translation schemes in the context of syntax-directed translation.
6. What is the role of semantic actions in a parser generator?
7. Explain how bottom-up evaluation works for S-attributed definitions.
8. What is the significance of attributes in syntax-directed definitions?
9. Describe the process of constructing a syntax tree from a given grammar.
10. What are the advantages of automatic parser generation?
11. Define error detection in the context of compilers. Why is it important?
12. What are the main types of syntax errors that can occur during parsing?
13. Explain the difference between ad-hoc and systematic error recovery methods.
14.What is a "panic mode" error recovery strategy? Provide an example.
15.Describe how semantic errors differ from syntax errors in a compiler.
16.What role does error reporting play in the error detection process?
17.Explain the term "error propagation" and its implications for error recovery.
18.What is the significance of the "error production" rule in grammar?
19.How can a compiler use a "phrase level" recovery strategy?
20.Define the term "rollback" in the context of error recovery.

Short questions

- Discuss the steps involved in converting a left-recursive grammar into a right-recursive grammar.
- Explain the top-down parsing process with an example using a simple CFG.
- Describe the construction and significance of an LR parsing table.
- What are the different types of LR parsers (e.g., SLR, LALR, LR(1))? Compare their capabilities.
- Explain how to detect and resolve ambiguities in a grammar.

- Discuss the steps involved in the automatic generation of parsers using parser generators.
- Explain the concept of L-attributed definitions with an example.
- Describe how syntax trees can be used to represent abstract syntax in compilers.
- Compare syntax-directed definitions (SDD) with syntax-directed translations (SDT).
- Provide an example of a syntax-directed definition and demonstrate how to evaluate it.

- Discuss various methods of error detection in compilers, including both static and dynamic methods.
- Explain the systematic error recovery approach and how it can be implemented in a compiler.
- Describe a case study of ad-hoc error recovery and its advantages and disadvantages.

- Explain how the use of error productions can improve error recovery in parsers.

Long Questions

- Construct an operator-precedence parsing table for the following grammar and demonstrate the parsing of the expression id + id * id:

- Describe the process of top-down parsing using recursive descent. Provide an example of a grammar and the corresponding parsing function.

- Discuss the implications of using ambiguous grammars in parsing. Provide an example and explain how different parse trees can be generated from it.

- Write a program that generates a syntax tree for a given arithmetic expression using a specified grammar. Include appropriate data structures and traversal methods.

- Implement a simple parser generator that takes a context-free grammar as input and produces an S-attributed definition for that grammar. Include examples and explanations of the output.

- Compare and contrast ad-hoc and systematic error recovery methods, providing examples for each approach.

- Provide a detailed explanation of panic mode error recovery, including its implementation and effectiveness in handling syntax errors.

- Discuss the concept of error reporting in compilers. How can effective error reporting enhance the user experience during compilation?

- Explain how the use of a recovery strategy can impact the overall efficiency of a compiler. Provide examples to illustrate your points.

- Discuss how semantic error detection and recovery can be implemented in a compiler, including the challenges involved.

- Implement an LR parser for a simple arithmetic expression grammar. Include the necessary components (grammar, parsing table, and parsing process) and demonstrate it with a sample input.