Home Assignment #1: Syntax Analysis in Compiler Design

Part 1: Theoretical Questions

1. Define syntax analysis in the context of compiler design. Why is it important?

2. Explain the difference between lexical analysis and syntax analysis.

3. What is a context-free grammar (CFG)? Provide an example of a simple CFG for arithmetic expressions.

4. Differentiate between top-down and bottom-up parsing techniques. Provide at least one example of each.

5. Describe the concept of a parse tree and its significance in syntax analysis.

6. List and briefly explain commonly used parsing algorithms (e.g., Recursive Descent, LL(1), LR(0), SLR, LR(1)).

Part 2: Practical Tasks – Bison and flex

Choose A or B

A. Grammar Construction

1. Design a context-free grammar for the following language constructs:

- Simple assignment statements (e.g., variable = expression;)
- Basic if-else statements

- While loops

2. Identify whether your grammar is ambiguous. If yes, suggest ways to eliminate ambiguity.

B. Parsing Technique Implementation

1. Write a recursive descent parser (in your preferred programming language) for simple arithmetic expressions involving +, -, *, /, parentheses, and integer literals. Your parser should:

- Check whether an input string is a valid expression.
- Generate a parse tree or report syntax errors.

Submission Guidelines

- Submit your written answers to Part 1 in a PDF document – **up to 1 pager only.**
- Assignment can be done **individually, as couples or gorups of 3 max.**
- Provide your parser implementation code with proper comments – we'll schedule time to review.
- Include sample input strings and parsing outputs.
- Provide your parsing table diagrams.
- Reflective answers should be concise, well-structured, and demonstrate understanding.