21CS8056
Sayan Paul

Compiler Design Lab-5

Assignment 5: Implementing Recursive Descent Parsing

Build a C program to parse and evaluate arithmetic expressions with the following rules:

- **The expressions can include addition (+), subtraction (-), multiplication (*), and division (/) operations.**
- **Parentheses can be used to group expressions.**
- **Variables can be represented by single letters (e.g., x, y).**
- **Numbers can be integers or floating-point values.**

Your task is to:

- **Implement a lexer to tokenize the input expression into a sequence of tokens.**
- **Implement a recursive descent parser to parse the tokens and build an abstract syntax tree (AST).**
- **Evaluate the expression represented by the AST.**

Design suitable inputs of your own to check your program.
Extra credit will be given to modular/function based coding styles.

The reason for my inability to complete the assignment in the compiler design lab is primarily due to the lack of knowledge regarding a linear time pattern matching algorithm capable of accurately detecting identifiers such as numeric literals (e.g., 1, 10, 200) within the given context. Despite extensive research efforts, I have not been able to identify a suitable algorithm that meets the requirements of our assignment. Furthermore, while exploring, I found that the gcc compiler uses tools like lex, to do so. I did not find relevant information or resources that could assist in resolving this challenge effectively.