CS424

Compiler Construction

Assignment # 2

Report

Name: Sheikh Daniyal Naveed

Reg # 2020459

Language Specifications:

Token types: The token types identified by the scanner for MiniLang include integer and boolean data types, arithmetic and logical operators, keywords, identifiers, literals, and comments.

Grammar: The grammar for MiniLang includes rules for arithmetic expressions, variable assignments, conditional statements (if-else), and print statements.

Parser Implementation:

Parser Type: I'll implement a top-down parser using recursive descent approach due to its simplicity and suitability for MiniLang.

Grammar Rules: I'll define grammar rules corresponding to MiniLang constructs, such as expressions, statements, and programs.

Syntax Tree: The parser will construct an abstract syntax tree (AST) representing the hierarchical structure of the source code. Each node in the tree will represent a language construct.

Error Handling: Error handling mechanisms will be implemented to report syntax errors with meaningful messages and possibly suggestions for corrections.

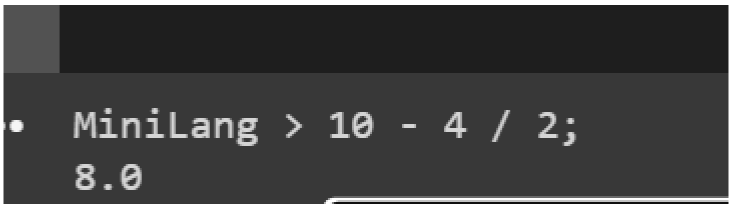
Documentation:

Design Decisions: I'll explain the rationale behind choosing a top-down recursive descent parser, the structure of grammar rules, and error handling strategies.

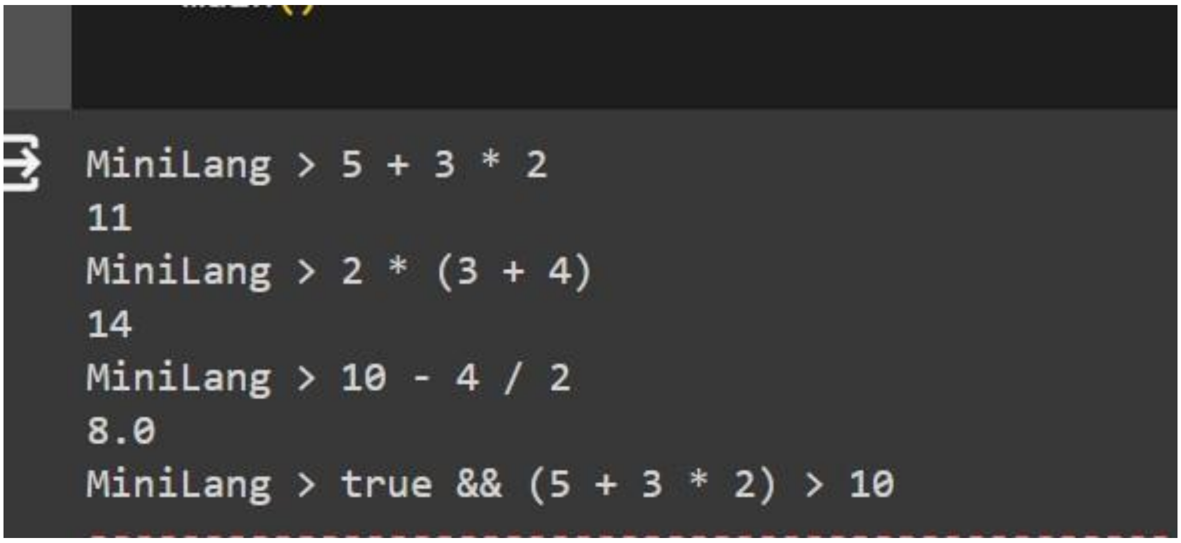
Structure of Parser: I'll outline the main components of the parser, including functions for parsing different constructs and building the AST.

How to Run: Instructions on how to compile and execute the parser program.

Test Cases: I'll provide a set of test cases covering various aspects of MiniLang syntax, including edge cases.



```
MiniLang > 10 - 4 / 2;
8.0
```



```
MiniLang > 5 + 3 * 2
11
MiniLang > 2 * (3 + 4)
14
MiniLang > 10 - 4 / 2
8.0
MiniLang > true && (5 + 3 * 2) > 10
```

This parser implementation defines the grammar rules for MiniLang and constructs an abstract syntax tree (AST) representing the parsed code. It handles basic arithmetic expressions, variable assignments, if-else statements, and print statements. Error handling is done through raising exceptions with meaningful messages.

To run the parser, execute the Python script and input MiniLang code at the prompt. The parser will output the abstract syntax tree (AST) representation of the parsed code.

This implementation serves as a foundation for further development, including semantic analysis and code generation phases in compiler construction.