**Compiler Construction  
Assignment 1  
Report**

Name: M. Abdullah Farooq  
Reg: 2020022

Scanner Design

The scanner for the MiniLang programming language is designed to tokenize source code according to the language's specifications. It follows a finite state machine approach to recognize different types of tokens such as keywords, identifiers, literals, operators, and comments.

Implementation Details

The scanner is implemented in Python using object-oriented programming principles. Here are the key components of the implementation:

1. Token Class: Represents a token with attributes for token type and lexeme.
2. Scanner Class: Responsible for scanning MiniLang source code files and tokenizing them. It includes methods for scanning, reporting errors, and handling different token types.
3. Scan Method: Reads the source code file line by line, identifying tokens based on their lexical patterns. It recognizes keywords, identifiers, integer literals, operators, and comments. Lexical errors are reported when encountered.
4. Error Handling: The scanner reports lexical errors such as invalid characters or malformed identifiers with the line number and column number where the error occurs.

Test Cases

To validate the correctness of the scanner implementation, the following test cases can be used:

1. Valid Input: Provide a MiniLang source code file containing valid syntax and ensure that all tokens are correctly identified. Include various combinations of tokens to verify the scanner's ability to handle different scenarios.
2. Invalid Input - Lexical Errors: Introduce lexical errors such as invalid characters or incomplete tokens in the source code file and verify that the scanner reports them accurately. This ensures that the scanner can appropriately detect and report errors to the user.
3. Edge Cases: Test the scanner with edge cases such as empty files, files with only comments, or files containing only one type of token. These cases help ensure that the scanner behaves as expected in various boundary conditions and corner cases.

Conclusion

The scanner for the MiniLang programming language provides a robust and efficient tool for tokenizing source code files according to the language's specifications. Its implementation in Python leverages object-oriented programming principles and a finite state machine approach to achieve accurate and reliable tokenization. Through comprehensive testing with a variety of test cases, including valid inputs, invalid inputs, and edge cases, the scanner demonstrates its ability to handle different scenarios and report errors effectively.

Overall, the MiniLang scanner serves as a foundational component for further stages of the language processing pipeline, enabling syntactic and semantic analysis to be performed on MiniLang programs with confidence in the accuracy and correctness of the tokenization process.