CS424

Assignment 1 - REPORT

Ghulam Ishaq Khan Institute of Science and Technology

REG#: 2020447

Name: Shahryar Mubashar
INSTRUCTOR: USAMA ARSHAD

Introduction:

MiniLang is a simple programming language designed to demonstrate fundamental programming concepts. This report outlines the design and implementation details of a scanner for MiniLang. The scanner is responsible for tokenizing MiniLang source code according to the language's specifications, handling lexical errors, and providing accurate output tokens. The report also includes a description of test cases used to validate the scanner's functionality.

Scanner Design:

The scanner is implemented in Python due to its simplicity and flexibility. It follows a modular structure consisting of the following components:

- Token Types: Enumeration constants are defined to represent different token types such as integer literals, boolean literals, identifiers, operators, keywords, parentheses/braces, comments, and lexical errors.

- Regular Expressions: Regular expressions are used to match tokens based on the language specifications. Patterns are defined for integer literals, boolean literals, identifiers, operators, keywords, parentheses/braces, and comments.

- Tokenize Function: The tokenize function reads MiniLang source code from a file, applies regular expressions to tokenize the code, and generates tokens. Error handling is incorporated within this function to detect and report lexical errors.

Implementation Details:

The scanner reads MiniLang source code from a file and tokenizes it according to the language's specifications. Regular expressions are used to efficiently match tokens in the input code. Error handling is implemented to detect invalid symbols or malformed identifiers during tokenization, ensuring the reliability and robustness of the scanner.

Test Cases:

Several test cases have been designed to validate the functionality of the scanner. These test cases cover various aspects of the MiniLang language, including arithmetic operations, boolean expressions, variable assignments, if-else conditions, comments, and identifiers. Each test case is carefully crafted to test specific functionalities of the scanner and ensure its correctness and robustness.

Example Test Cases:

1.

```
x = 10;
if (x == 10) {
print(true);
} else {
print(false);
}
```

2.

```
x = 5 + 3;
```

3.

```
a = 5;
b = true;
c = a * 2 + 1;
```

Conclusion:

The scanner for MiniLang is successfully designed and implemented, adhering to the language specifications. It efficiently tokenizes MiniLang source code, handles lexical errors, and produces accurate output tokens. The provided test cases demonstrate the scanner's capabilities in various scenarios, ensuring its correctness and reliability.