**Sameer Arif Khan 2020430**

**Compiler Construction CS-424**

**Assignment 1**

**MiniLang Scanner Documentation**

**Design Decisions**

The MiniLang scanner is designed to tokenize source code written in the MiniLang programming language according to its specifications. To achieve this, the scanner utilizes a finite state machine approach, where regular expressions are used to match and tokenize different types of tokens such as integers, identifiers, keywords, operators, etc. The scanner also handles single-line comments and reports lexical errors when encountering invalid symbols or malformed identifiers.

**Structure of the Scanner**

The scanner consists of the following components:

1. **Token Types:** A class defining different types of tokens such as INTEGER, BOOLEAN, OPERATOR, etc.

2. **Token Class**: A class representing individual tokens with attributes for type and lexeme.

3. **Scan Function**: The main function responsible for scanning MiniLang source code from a file and tokenizing it. It utilizes regular expressions to match tokens based on the language specifications.

4. **scan\_file Function**: A helper function within the scan function that reads from a file line by line and tokenizes each line accordingly. It also handles comments and whitespace.

5. **Main Function**: The entry point of the program that demonstrates the usage of the scanner by providing a file name containing MiniLang source code.

**How to Run the Program**

To run the MiniLang scanner program, follow these steps:

**1.** Ensure you have Python installed on your system.

**2.** Save the provided MiniLang scanner code in a Python file (e.g., `minilang\_scanner.py`).

**3.** Create a MiniLang source code file (e.g., `**example**.**minilang**`) containing valid MiniLang code or use one of the provided test cases.

**4.** Open a terminal or command prompt.

**5.** Navigate to the directory containing the Python file and the MiniLang source code file.

**6.** Run the Python script by executing the command: `python minilang\_scanner.py`

The scanner will tokenize the MiniLang source code and display the token type along with the lexeme for each token.

**Test Cases**

The scanner is tested with the following test cases:

1. **tc\_1.minilang**: Tests simple arithmetic and variable assignment.

2. **tc\_2.minilang**: Tests conditional statements.

3. **tc\_3.minilang**: Tests boolean expressions and comments.

These test cases cover various aspects of the MiniLang language, including arithmetic operations, conditional statements, boolean expressions, and comments. They ensure that the scanner correctly tokenizes different constructs of the MiniLang language and handles edge cases such as whitespace, comments, and invalid symbols.