CS424

Assignment 2 – REPORT

Ghulam Ishaq Khan Institute of Science and Technology

Name: Shaheer Nashad

REG#: 2020442

INSTRUCTOR: USAMA ARSHAD

Lexer:

- The lexer tokenizes the input text into tokens based on defined token types.
- It skips whitespace characters and handles integer literals, arithmetic operators, parentheses, etc.
- Error handling is implemented to raise exceptions for invalid characters.

Parser:

- The parser constructs an abstract syntax tree (AST) from the tokens generated by the lexer.
- It defines grammar rules for arithmetic expressions using recursive descent parsing.
- Grammar rules include factors (integers and parentheses), terms (multiplication and division), and expressions (addition and subtraction).
- Error handling is implemented to catch syntax errors and provide meaningful error messages.

AST Nodes:

- AST nodes represent different constructs in the MiniLang language, such as binary operations and numbers.

Main Program:

- The main program takes input from the user, initializes the lexer and parser, constructs the AST, and prints it.

How to Run the Program:

- Run the Python script.
- Enter MiniLang expressions at the prompt, and the program will parse and print the corresponding abstract syntax tree.

Test Cases:

Case 1: Simple Arithmetic Expression

Input:

```
MiniLang> 3 + 5 * 2;
```

Output:

```
ADD
  3
  MULTIPLY
    5
    2
```

Case 2: Parentheses and Precedence

Input:

```
MiniLang> (3 + 5) * 2;
```

Output:

```
MULTIPLY
  ADD
    3
    5
  2
```

Case 3: Invalid Syntax (Missing Operand)

Input:

```
MiniLang> 10 / ;
```

Output:

```
Syntax Error: Invalid syntax
```

Conclusion:

The MiniLang parser successfully parses arithmetic expressions, handles variables, and performs error handling for invalid syntax and characters. The provided test cases demonstrate various scenarios, including valid expressions, invalid syntax, and error handling.