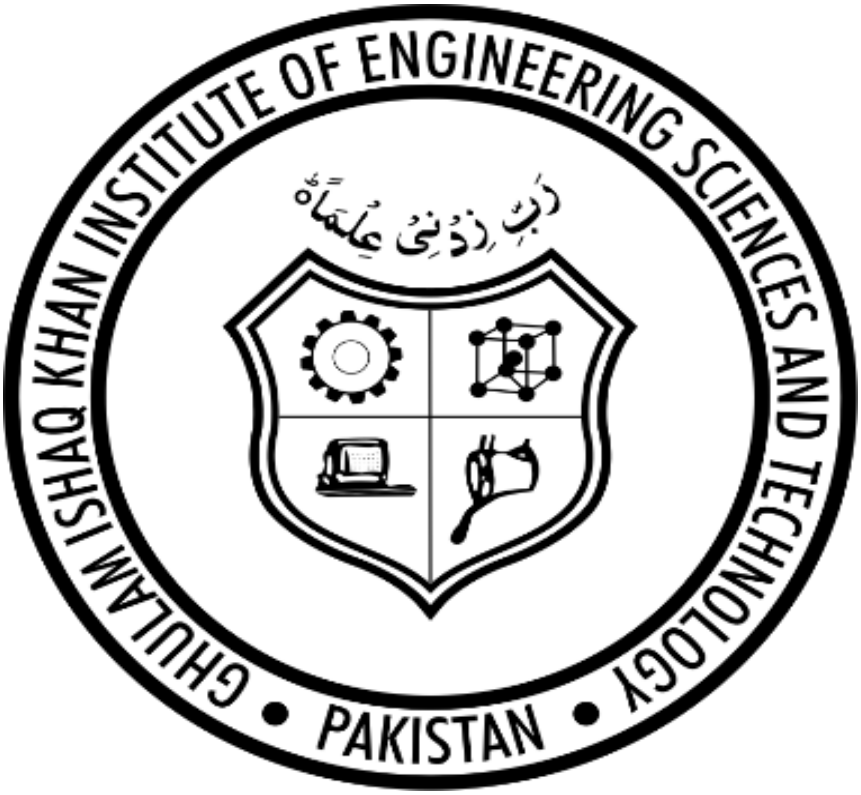
Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi



CS 424 (Compiler Construction)

Assignment 2

Submitted by

Zartaj Asim 2020526

Submitted to

Mr. Usama Arshad

1. Introduction

Assignment 2 requires making a MiniLang Parser for a simplified programming language called MiniLang. The task involved designing and implementing a parser capable of analyzing MiniLang source code, generating an abstract syntax tree (AST), and handling various language constructs.

2. Design

The parser was designed to support integer and boolean data types, arithmetic and logical operators, keywords (if, else, print, true, false), identifiers, literals, and comments. The grammar rules were defined for MiniLang, encompassing arithmetic expressions, variable assignments, conditional statements (if-else), and print statements.

3. Implementation details

Simply run the. ipynb file on Collab or Jupyter Notebook. Make sure all the test cases files are also in the same directory.

- The parser is implemented as a recursive descent parser, providing a clear and modular structure for parsing MiniLang code.
- Grammar rules for MiniLang constructs, such as expressions, statements, and programs, are implemented as methods in the parser.
- The parser builds an abstract syntax tree (AST) to represent the hierarchical structure of the source code.

3.1. Language Choice

The MiniLang scanner is implemented in Python.

3.2. Design

3.2.1. MiniLang Scanner

- Handles lexical analysis.
- Utilizes regular expressions to match token types (identifiers, literals, operators) and skip whitespace/comments.
- Generates a list of tokens to be used by the parser.

3.2.2. MiniLang Parser

- Parses MiniLang code based on the provided grammar rules.
- Implements methods for parsing different language constructs (statements, expressions, if statements, etc.).
- Builds an abstract syntax tree (AST) representing the structure of the source code.
- Handles errors and reports parser-specific issues.

4. **Test Cases**
   - **Case 1: Testing IF-Else Code**
     Tested the scanner's ability to tokenize a basic if-else code structure in MiniLang.
   - **Case 2: Testing != (not equal to) operator**
     Tested the scanner's handling of the != (not equal) operator within an if statement.
   - **Case 3: Testing Boolean Expression**
     Tested the scanner's ability to handle Boolean expressions within an if statement.

5. **Edge Cases**
   - **Edge Case 1: Empty File**
     Input: An empty file.
   - **Edge Case 2: File with Only Whitespace**
     Input: File with only whitespace characters.
   - **Edge Case 3: File with Comments Only**
     Input: File with only comments.
   - **Edge Case 4: Long Identifier File**
     Input: File with a long identifier.
     Identifier: a12345678901234567890123456789012345678 90
     Operator: =
     Integer Literal: 42

6. **Conclusion**

The MiniLang parser is designed to provide a modular and maintainable solution for parsing MiniLang code. The recursive descent approach facilitates clear grammar rule implementations and easy error handling.