Compiler Construction- CS424 Assignment-1

Reg no: 2020517 Name :Warda Bibi

MiniLang Scanner Design Report

Design Decisions:

1. Token Definitions:
The scanner defines tokens based on the given specifications. Each token type corresponds to a particular pattern in the input stream, such as keywords, identifiers, literals, operators, parentheses, etc.

2. Regular Expressions:
Regular expressions are utilized to match patterns in the input stream. These patterns define the lexical structure of the programming language.

3. Ignoring Comments:
Comments, specified as lines starting with "//", are ignored by the scanner. This decision ensures that comments do not interfere with token recognition and parsing.

4. Error Handling:
Invalid tokens and characters that do not match any defined pattern are treated as errors and printed as such. This helps in identifying lexical errors in the input code.

Scanner Structure:

The scanner is implemented using Flex (Lexical Analyzer Generator). It consists of the following components:

1. Header Section: Includes necessary header files and declarations.

2. Options Section: Contains options for the scanner, such as `noyywrap`, which disables default input buffering.

3. Rules Section: Defines rules for recognizing tokens and patterns in the input stream. Each rule consists of a pattern and an associated action.

4. C Code Section: Contains additional C code, such as the `main()` function, to initialize the scanner and call the lexer function (`yylex()`).

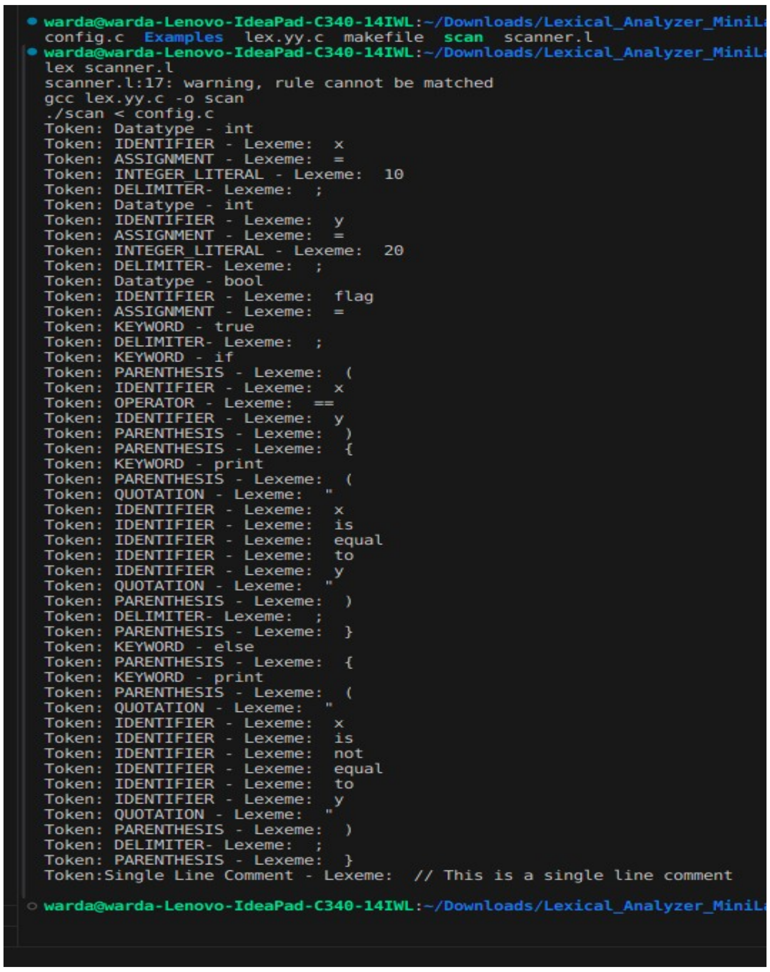
Running the Program:

Run make to run the make file created for scanner

Test Cases:

Example 1:

```
int x = 10;
int y = 20;
bool flag = true;
if (x == y) {
print("x is equal to y");
} else {
print("x is not equal to y");
}
// This is a single line
comment
```

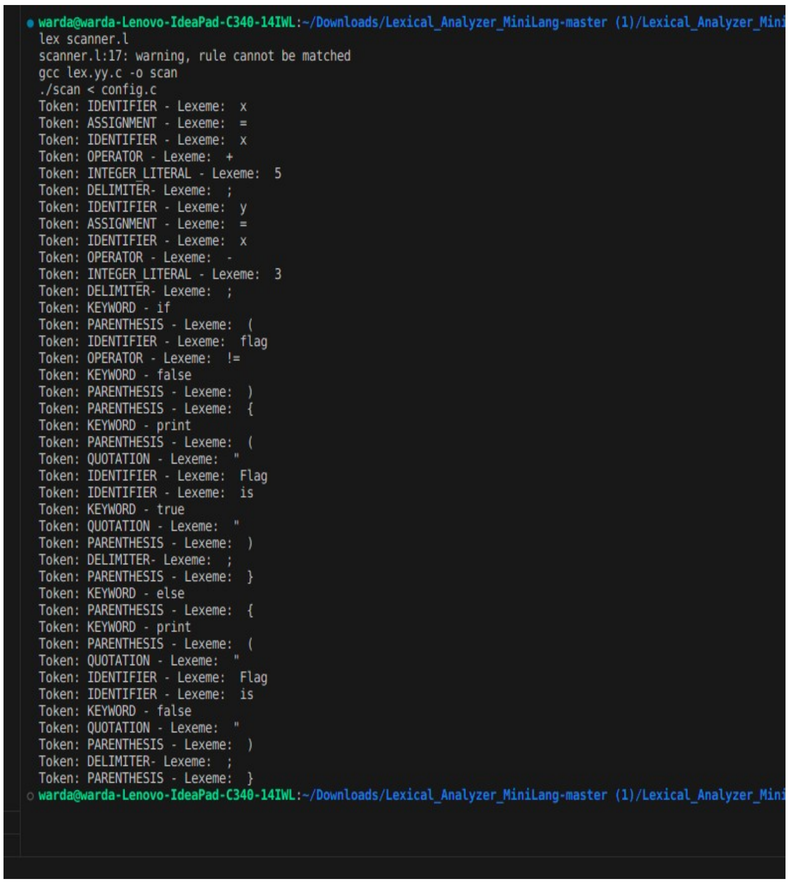


```
● warda@warda-Lenovo-IdeaPad-C340-14IWL:~/Downloads/Lexical_Analyzer_MiniLa
  config.c  Examples  lex.yy.c  makefile  scan  scanner.l
● warda@warda-Lenovo-IdeaPad-C340-14IWL:~/Downloads/Lexical_Analyzer_MiniLa
  lex scanner.l
  scanner.l:17: warning, rule cannot be matched
  gcc lex.yy.c -o scan
  ./scan < config.c
  Token: Datatype - int
  Token: IDENTIFIER - Lexeme:  x
  Token: ASSIGNMENT - Lexeme:  =
  Token: INTEGER_LITERAL - Lexeme:  10
  Token: DELIMITER- Lexeme:  ;
  Token: Datatype - int
  Token: IDENTIFIER - Lexeme:  y
  Token: ASSIGNMENT - Lexeme:  =
  Token: INTEGER_LITERAL - Lexeme:  20
  Token: DELIMITER- Lexeme:  ;
  Token: Datatype - bool
  Token: IDENTIFIER - Lexeme:  flag
  Token: ASSIGNMENT - Lexeme:  =
  Token: KEYWORD - true
  Token: DELIMITER- Lexeme:  ;
  Token: KEYWORD - if
  Token: PARENTHESIS - Lexeme:  (
  Token: IDENTIFIER - Lexeme:  x
  Token: OPERATOR - Lexeme:  ==
  Token: IDENTIFIER - Lexeme:  y
  Token: PARENTHESIS - Lexeme:  )
  Token: PARENTHESIS - Lexeme:  {
  Token: KEYWORD - print
  Token: PARENTHESIS - Lexeme:  (
  Token: QUOTATION - Lexeme:  "
  Token: IDENTIFIER - Lexeme:  x
  Token: IDENTIFIER - Lexeme:  is
  Token: IDENTIFIER - Lexeme:  equal
  Token: IDENTIFIER - Lexeme:  to
  Token: IDENTIFIER - Lexeme:  y
  Token: QUOTATION - Lexeme:  "
  Token: PARENTHESIS - Lexeme:  )
  Token: DELIMITER- Lexeme:  ;
  Token: PARENTHESIS - Lexeme:  }
  Token: KEYWORD - else
  Token: PARENTHESIS - Lexeme:  {
  Token: KEYWORD - print
  Token: PARENTHESIS - Lexeme:  (
  Token: QUOTATION - Lexeme:  "
  Token: IDENTIFIER - Lexeme:  x
  Token: IDENTIFIER - Lexeme:  is
  Token: IDENTIFIER - Lexeme:  not
  Token: IDENTIFIER - Lexeme:  equal
  Token: IDENTIFIER - Lexeme:  to
  Token: IDENTIFIER - Lexeme:  y
  Token: QUOTATION - Lexeme:  "
  Token: PARENTHESIS - Lexeme:  )
  Token: DELIMITER- Lexeme:  ;
  Token: PARENTHESIS - Lexeme:  }
  Token:Single Line Comment - Lexeme:  // This is a single line comment

○ warda@warda-Lenovo-IdeaPad-C340-14IWL:~/Downloads/Lexical_Analyzer_MiniLa
```

Example 2:

```
x = x + 5;
y = x - 3;
if (flag != false) {
print("Flag is true");
} else {
print("Flag is false");
}
```



```
● warda@warda-Lenovo-IdeaPad-C340-14IWL:~/Downloads/Lexical_Analyzer_MiniLang-master (1)/Lexical_Analyzer_Mini
  lex scanner.l
  scanner.l:17: warning, rule cannot be matched
  gcc lex.yy.c -o scan
  ./scan < config.c
  Token: IDENTIFIER - Lexeme:  x
  Token: ASSIGNMENT - Lexeme:  =
  Token: IDENTIFIER - Lexeme:  x
  Token: OPERATOR - Lexeme:  +
  Token: INTEGER_LITERAL - Lexeme:  5
  Token: DELIMITER- Lexeme:  ;
  Token: IDENTIFIER - Lexeme:  y
  Token: ASSIGNMENT - Lexeme:  =
  Token: IDENTIFIER - Lexeme:  x
  Token: OPERATOR - Lexeme:  -
  Token: INTEGER_LITERAL - Lexeme:  3
  Token: DELIMITER- Lexeme:  ;
  Token: KEYWORD - if
  Token: PARENTHESIS - Lexeme:  (
  Token: IDENTIFIER - Lexeme:  flag
  Token: OPERATOR - Lexeme:  !=
  Token: KEYWORD - false
  Token: PARENTHESIS - Lexeme:  )
  Token: PARENTHESIS - Lexeme:  {
  Token: KEYWORD - print
  Token: PARENTHESIS - Lexeme:  (
  Token: QUOTATION - Lexeme:  "
  Token: IDENTIFIER - Lexeme:  Flag
  Token: IDENTIFIER - Lexeme:  is
  Token: KEYWORD - true
  Token: QUOTATION - Lexeme:  "
  Token: PARENTHESIS - Lexeme:  )
  Token: DELIMITER- Lexeme:  ;
  Token: PARENTHESIS - Lexeme:  }
  Token: KEYWORD - else
  Token: PARENTHESIS - Lexeme:  {
  Token: KEYWORD - print
  Token: PARENTHESIS - Lexeme:  (
  Token: QUOTATION - Lexeme:  "
  Token: IDENTIFIER - Lexeme:  Flag
  Token: IDENTIFIER - Lexeme:  is
  Token: KEYWORD - false
  Token: QUOTATION - Lexeme:  "
  Token: PARENTHESIS - Lexeme:  )
  Token: DELIMITER- Lexeme:  ;
  Token: PARENTHESIS - Lexeme:  }
○ warda@warda-Lenovo-IdeaPad-C340-14IWL:~/Downloads/Lexical_Analyzer_MiniLang-master (1)/Lexical_Analyzer_Mini
```

Conclusion:

The scanner has been designed to effectively recognize tokens and patterns in the input stream based on the given specifications. It successfully identifies keywords, identifiers, literals, operators, parentheses, and ignores comments. Additionally, it handles invalid tokens and characters gracefully, providing a robust lexical analysis component for the overall compiler system.