**Name: Iasam KaKaKhail**

**Reg\_No: 2020490**

**Compiler Construction Lab Mid**

**Code Structure**:

The parser.y code consists of the following main sections:

* %{...%}: This section includes the necessary header files and declares the external functions and variables used in the code.
* %union {...}: This section defines the union structure to hold the different data types used in the parser.
* %token and %type: These sections define the token types and the types of the non-terminal symbols used in the grammar.
* %%: This section contains the grammar rules for the language being parsed.
* main(): This is the entry point of the program, which initializes the parser and starts the parsing process.
* yyerror(): This function is called when an error occurs during the parsing process and prints an error message.

**Grammar Rules**:

The grammar rules defined in the %% section specify the syntax of the language being parsed. The rules cover the following:

* program\_input: Defines the overall structure of the input, which can be multiple lines.
* line: Defines the structure of a single line, which can be either a calculation or an assignment.
* calculation: Defines the structure of a calculation, which can be either an expression or an assignment.
* expr: Defines the structure of an expression, including arithmetic operations and variable references.
* assignment: Defines the structure of a variable assignment.

The grammar rules follow the BODMAS (Brackets, Order, Division, Multiplication, Addition, Subtraction) order of operations.

Lex Code

Code Structure

The lex code consists of the following main sections:

* %{...%}: This section includes the necessary header files and declares the external functions and variables used in the code.
* %option noyywrap: This option tells the lex generator not to generate a yywrap() function, which is not needed in this case.
* %%: This section contains the regular expressions and associated actions for the lexical analyzer.

**Lexical Rules**:

The regular expressions and actions defined in the %% section specify the lexical structure of the language being parsed. The rules cover the following:

* Whitespace: Skips over any whitespace characters.
* Numbers: Recognizes and returns numeric values.
* Arithmetic operators: Recognizes and returns the arithmetic operators (+, -, \*, /).
* Equals sign: Recognizes and returns the equals sign (=).
* Parentheses: Recognizes and returns the left and right parentheses.
* Variable keywords: Recognizes and returns the "var" and "VAR" keywords.
* Variables: Recognizes and returns variable names, which are added to the variable store.
* Newline: Recognizes and returns the newline character.
* Undefined symbols: Recognizes and reports any undefined symbols.

The lex code works in conjunction with the parser.y code to provide a complete implementation of the calculator language.

**Functionality**

- Basic arithmetic following BODMAS rules e.g, 4 \\* (3 + 2) = 20

- Standard functions (modulo, ceil, abs, floor)

- Variable stores (create and use your own variables) i.e. (var a = ?? or VAR b = ??)

Compile and execute

bison -d parser.y or yacc -d parser.y

flex lex.l

gcc parser.tab.c lex.yy.c -lm -o calc

calc