Ercan Mert Esen

21290195

Description Of The Rules:

Integers can be created by a single or more digits and can be signed or unsigned. "string" is defined for strings, "int" is defined for integer values, "flt" is defined for floating values and "identifier" is defined as a data type to use for variable names. Strings can be created by char data types.

Usage of "for loop":

The initialization, condition and updation must be in between two brackets and there must be semicolons in between them. The body of the for loop must be written after the "then" keyword.

Usage of "while loop":

The condition for the while loop must be in between two brackets and the body of the while loop must be written after the "then" keyword.

Usage of "if statement":

The condition for if statement must be in between two brackets and the body of the if statement to be executed must be written after the "then" keyword.

Backus–Naur form

```
<Comparison> ::= == | != | < | > | <= | >=

<LogicOperator>::= && | ! | ||
```

```
<digits>::= <digit>

                  | <digits><digit>

<digit>::= 0

                  | <non zero digit>

<non zero digit>::= <digits>.<digit>

                  | <exponent part>
```

```
                    | <digits><exponent part>
<floating>::= <digits>.<digits>
                    | <exponent part>
                    | <digits><exponent part>
<exponent part>::= <exponent indicator><integer>
< exponent indicator>::= e
<integer>::= <sign>?<digits>
<number>::= <integer>
                    | <floating>
<sign>::= + | -
<null>::= NULL
<letters>::= a| b| c| d| e| f| g| h| i| j| k| l| m| n| o| p| r| s| t| u| v| w| x| y| z| A| B| C|
D| E| F| G| H| I| J| K| L| M| N| O| P| R| S| T| U| W| V| X| Y| Z
<chars>::= <digits>
                    | <letters>
<special chars>::= \n | \r | \t
<punctuation> ::= .| ?| ,| ‘| “| (| )| ;| :| !| {| }| [| ]| %| $| #| -| *| =



<text> ::= <text>
                    |<chars>
                    | <punctuation>
<string> ::= “<text>”
<boolean expression> ::= <expression><conditional operators><expression>
                    | <expression>
                    | <string><check strings><string>
<expression> ::= <variable name>
                    | <number>
<any statement> ::= <if statement>
                    | <assignment statement>
```

| <loop statement>

| <arithmetic operations>

<variable name> ::= <variable name>

| <letters><chars>

| <letters>

<declaration statement> ::= <variable type> <variable name> ;

| <variable type> <assignment statement>

<assignment statement> ::= <*left hand side*> <*assignment operator*> <*assignment*> ;

<*left hand side*> ::= <*variable name*>

| <*array access*>

| <graph access>

<*assignment operator*> ::= **=** |**+=** | **-=**

<assignment> ::= <integer>

| <floating>

| <null>

| <string>

| <variable name>

|<arithmetic statement>

<ArithmeticStatement0> ::= <ArithmeticStatemen0t> + <ArithmeticStatement1>

| <ArithmeticStatement1>

<ArithmeticStatement1> ::= <ArithmeticStatement1> - <ArithmeticStatement2>

| <ArithmeticStatement2>

*<ArithmeticStatement2> ::= <ArithmeticStatement2> * <ArithmeticStatement3>*

| <ArithmeticStatement3>

<ArithmeticStatement3> ::= <ArithmeticStatement3> / <ArithmeticStatement4>

| <ArithmeticStatement4>

<ArithmeticStatement4> ::= (<ArithmeticStatement>)

| <integer>

| <floating>

<loop statement> ::= <while loop>

| <for loop>

<while loop> ::= while (<boolean expression>) then <any statement>

<for loop> ::= for (<assignment statement> ; <boolean expression>; <assignment statement>) then <any statement>

<any statement> ::= <any statement>

| <any nonif statement>

| <if statement>

<if statement> ::= <matched>

| <unmatched>

<matched> ::= if (<boolean expression>) then <matched> else <matched>

| <any nonif statement>

<unmatched> ::= if (<boolean expression>) then <if statement>

| if <boolean expression> then <matched> else <unmatched>

<any nonif statement> ::= <any nonif statement>

| <assignment statement>

| <loop statement>

| <assignment statement>

| <function call>

Contents Of Files:

"mpl.l" file:

This file is for lexical analysis. In this file, input is broken down into tokens and a symbol table is created with these tokens.

"mpl.y" file:

This file is for syntax analysis. The tokens gathered after lexical analysis are examined to make syntactically correct sentences and the line containing syntax errors are printed out after execution.