SER 502: Milestone 1 (Team 18)

Language Name:

MOWA Language

GitHub Repo: https://github.com/Vijay-Giduturi/SER502-Spring2022-Team18

Team Members: Team 18

Vijay Ram Giduturi - 1223292304
Nikhil Alapati - 1222336349
Phani Teja Inaganti - 1223458470
Krishna Chandra Sen Dadi - 1222678613
Srikruthi Vedantham - 1223040832

Language Design:

- The language we are implementing, **MOWA**, is an **imperative language**.

- Data type implementation:
  We represent integers using the datatype **num**.
  Strings are implemented in this language with the datatype **ntr**.
  Boolean values(True/False) can be represented with the datatype **bool**.

- The following operators are supported in MOWA:
  Arithmetic: +,-,*,/
  Unary: ++,--
  Relational: <,>,<=,>=,==,~=
  Logical: ^(and),|(or),~(not)

- Assignment operator can be represented as →

- Conditional constructs to be implemented in the language are:
  1). Traditional **if-then-else** statements.

```
If(bool) then
{Block of Code}
Else if(bool) then
        {Block of Code}
Else
        {Block of Code}
```

  2). Ternary operator - **?!** (Condition**?**True**!**False)

- Looping structures would be:
    1). Traditional **for** loop –
        for(Initialization; bool;Unary)
            {Block of Code}
    2). Traditional **while** loop–
        while(bool)
            {Block of Code}
    3). **for Identifier in range(num,num)**
        {Block of Code}

- Print statements can be implemented as:
    show()- Prints on the output screen**,**
    shownl()- Prints on the output screen in a new line.

- Every statement must end with ****.

- Comments in the code can be done with:
    - Single line comments- Start with #
    - Multi line comments- Start with ## end with ##

- Variable declaration:
    - Numbers can be declared as **num <variable name>**
    - Boolean values can be declared as **bool <variable name>**
    - Strings can be declared as **ntr <variable name>**
    - A variable can be declared anywhere in the program (except loops) and can be used from its declaration to the end of the program.
    - Once a variable is declared, it cannot be declared in any other part of the program.

- If a variable is not initialized, the default values are:
    - num- 0
    - ntr- “”
    - bool - false

- Compiler/Interpreter Procedure and Runtime environment:
    - A lexer is implemented to convert the input program into tokens. In the next step, syntactic analysis is done which checks if the expressions are following the rules by generating a parse tree from the input tokens. This parse tree is interpreted to give us an output if it is semantically correct, else an error is displayed.

- We'll use the tool ANTLR to generate parser in Java.

Grammar:

P – PROGRAM
K – BLOCK
S - STATEMENT
D – DECLARATION
IN – INITIALIZATION
U – UNARY EXPRESSION
I – IDENTIFIER
E – EXPRESSION
B – BOOLEAN EXPRESSION
C – CONDITION
T – TERNARY OPERATOR

P ::= K

K ::= S K | S

S ::= D | IN | IF | FOR | WHILE | Print | U

D::= num I\ | bool I\ | ntr I\

IN::= I → E\ | I → B\ | I → "I"\ | D → E\| D →B\ | D → "I"\

U::= I++\ | I--\

IF::= if C { K } ELSE_IF | if C { K } ELSE

ELSE_IF::= else if B { K } ELSE_IF

ELSE::=else { K }| empty

FOR::= for (IN ; C ; U) { K } | for I in range (N , N) { K }

WHILE::= while(C) {K}

Print::= show(“V”)\ | show(I)\ | show(“V”,I)\ | shownl(“V”)\ | shownl(I)\ | shownl(“V”,I)\

B ::= true | false | ~ B | B | B | B ^ B

C ::= E < E | E > E | E<= E | E >= E | E == E | E ~= E | B

E ::= E + E | E - E | E * E | E / E | (E) | I | N | T

I::= [a-z] Ids | [A-Z] Ids

Ids ::= [a-z] | [A-Z] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | empty

N::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

T::= (C) ? (E ! E)

Example Code:

```
num x=2\
num y\
# This is a comment starting with the mentioned symbol
y=3\
num z=x+y\
show(“The value of z is : ”,z)\
```

Output:

```
The value of z is : 5
```