Course: CECS 4200 **Section: 08** **Term: Winter 2016**

Project

Topic: Interpreter

Due date: February 16, 2017

Professor Luis A. Ortiz Ortiz

Project Description:

Design and implement an interpreter that executes the code written in *LAO* language. The statements in *LAO* language are:

- <comment statement> ::= **REM** | **REM** <any string>
- <assignment statement> ::= <variable> **=** <arithmetic expression>
- <print statement> ::= **PRINT** | **PRINT** <variable> | **PRINT** <number> | **PRINT** <string>
- <read statement> ::= **READ** <variable>
- <if statement> ::= **IF** <conditional expression> **THEN** <then statement>
- <end statement> ::= **END.**

Where:

<variable> ::= <variable> <letter> | <letter>

<letter> ::= **a-z** | **A-Z**

<number> ::= <integer> | <real>

<integer> ::= <unsign integer> | <sign> < unsign integer>

<unsign integer> ::= <unsign integer> <digit> | <digit>

<real> ::= <integer> <decimal part> | <integer> <decimal part> <exponent part>

<decimal part> ::= <decimal point> <unsign integer>

<decimal point> ::= **.**

<exponent part> ::= <exponent> <integer> <decimal part>

<exponent> ::= **e** | **E**

<digit> ::= **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

<sign> ::= **+** | **-**

<string> ::= **"**<any string>**"** | **""**

<any string> ::= <any sequence of characters>

<relational operator> ::= **.gt.** | **.lt.** | **.eq.** | **.ge.** | **.le.** | **.ne.**

<logical operator> ::= **.or.** | **.and.** | **.not.**

<then statement> ::= <assignment statement> | <read statement> | <print statement>

A conditional expression must include logical or relational operators. Examples:

- `3 .gt. 5`
- `.not. 3 .gt. 5`
- `3 .gt. 5 .and. 3 .eq. var`
- `.not. 3 .gt. 5 .and. 3 .eq. var`

The *keywords* are reserved words. They are: **IF**, **THEN**, **READ**, **PRINT**, and **END**. In addition, the language is not case sensitive. An instruction with only the *keyword* **PRINT** is equivalent to a new line when the instruction is executed by the interpreter. Moreover, the interpreter will ignore the comments statements.

The alphabet of the language is the characters in the *ASCII table*. Additionally, the type of a variable is determined by the first letter of the variable name. The following table shows the previous:

First letter of the variable name	**Variable type**
A-F	integer
G-N	real
O-Z	string

The following table shows the results of the arithmetic operators according to the different types (integer, real, and string):

Arithmetic Operator	**Operands**	**Result**	**Example**
.add.	integer vs. integer	integer	`2 .add. 2 => 4`
	real vs. real	real	`2.0 .add. 2.0 => 4.0`
	real vs. integer	real	`2.0 .add. 2 => 4.0`
	string vs. string	string	`"ab" .add. "c" => "abc"`
	integer vs. string	string	`3 .add. "ab" => "3ab"`
	real vs. string	string	`"ab" .add. 3.5 => "ab3.5"`
.sub.	integer vs. integer	integer	`2 .sub. 2 => 0`
	real vs. real	real	`2.0 .sub. 2.0 => 0.0`
	real vs. integer	real	`2.0 .sub. 2 => 0.0`
.mul.	integer vs. integer	integer	`2 .mul. 2 => 4`
	real vs. real	real	`2.0 .mul. 2.0 => 4.0`
	real vs. integer	real	`2.0 .mul. 2 => 4.0`
.div.	integer vs. integer	integer	`2 .div. 2 => 1`
	real vs. real	real	`2.0 .div. 2.0 => 1.0`
	real vs. integer	real	`2.0 .div. 2 => 1.0`

The following table describes the associative and precedence of the logical, relational, and arithmetic operators:

Operator	**Associative**	**Precedence (low to high)**
.or.	left to right	1
.and.	left to right	2
.not.	right to left	3
.eq. .ne.	left to right	4
.lt. .le. .gt. .ge.	left to right	5
.add. .sub.	left to right	6
.mul. .div.	left to right	7

The two operands of a relational operator will be:

- integer vs. integer
- real vs. real
- real vs. integer
- string vs. string

The interpreter must read from a file the instructions of a program written in *LAO*. In case of an error in an instruction, the interpreter must stop and show:

- the line number of the instruction that provoked the error
- the instruction that provoked the error
- the type of instruction that provoked the error, what is the error, and possible solution (the interpreter must only show unknown statement if the instruction does not match the beginning of any valid statement in the language *LAO*)

Sample program in *LAO* language:

```
w = "HOLA"
y = " MUNDO"
if 3 .gt. 2 then a = 2 .add. 1 .add. 3 .mul. 4
z = w .add. y
print z
print
print
if z .eq. "HOLA MUNDO" then print z
cc = 0
c2 = 9
c = 4
print
print
if c .eq. 4 then a = c .add. 3 .div. 3
n = 3.0 .add. 2 .mul. c
print "n = "
print n
print
print
a = 3 .mul. c .add. 2
print "a = "
print a
print
print
print "ENTER AN INTEGER NUMBER: "
read b
print
print
print "YOU ENTERED: "
print b
print
print
print
if b .gt. 7 .and. b .le. 9 .or. b .eq. 8 then print "a = "
if b .gt. 7 .and. b .le. 9 .or. b .eq. 8 then print a
print
print
end.
```