CS 4308 – Concepts of Programming Languages

Course Project

The project for this course is the development of an interpreter of a language implemented in any of the following **programming languages: C / C + + , Java, Python, or Ada.**

Interpreter Project

This project consists of **developing an interpreter** for a **minimal form (subset) of the Lua language**. This minimal form of Lua has only 1 data type, integer, and the only identifiers are single letters. Lua is not case sensitive. The specification of the grammar is given below.

Download Lua http://www.lua.org/download.html

Reference Manual http://www.lua.org/manual/5.1/manual.html

The interpreter will process a Lua program and build some intermediate data structures. These data structures will then be interpreted to execute the program. All tokens in this language are separated by white space. The parsing algorithm should detect any syntactical or semantic error. The first such error discovered should cause an appropriate error message to be printed, and then the interpreter should terminate. Run-time errors should also be detected with appropriate error messages being printed.

Deliverables (see course schedule for due dates):

1. Module_3 – 1st Deliverable

Develop a complete scanner. You must show the execution of this program by using a relevant source line as input, the program must show a list of the tokens in that line. Write a short report describing the work performed.

2. Module_5 – 2nd Deliverable

Develop a complete parser that executes with the scanner. You must show the execution of this program by using a relevant source line as input, the program must show the corresponding statement recognized. Write a short report describing the work performed.

3. Module_7 – 3rd Deliverable

Develop a complete interpreter that executes with the scanner and parser. You must show the

execution of this program by using a relevant source line as input, the program must show the results after executing the statement recognize by the parser. Write a short report describing the work performed.

Grammar for the (subset of Lua) language

Syntax Analyzer

<program> → function id ( ) <block> end

<block> → <statement> | <statement> <block>

<statement> → <if_statement> | <assignment_statement> | <while_statement> | <print_statement> | <repeat_statement>

<if_statement> → if <boolean_expression> then <block> else <block> end

<while_statement> → while <boolean_expression> do <block> end

<assignment_statement> -> id <assignment_operator> <arithmetic_expression>

<repeat_statement> -> repeat <block> until <boolean_expression>

<print_statement> → print (<arithmetic_expression>)

<boolean_expression> → <relative_op> <arithmetic_expression> <arithmetic_expression>

<relative_op> → le_operator | lt_operator | ge_operator | gt_operator | eq_operator | ne_operator

<arithmetic_expression> → <id> | <literal_integer> | <arithmetic_op> <arithmetic_expression> <arithmetic_expression>

<arithmetic_op> → add_operator | sub_operator | mul_operator | div_operator

Lexical Analyzer

id → letter

literal_integer → digit literal_integer | digit

assignment_operator → =

le_operator → <=

lt_operator → <

ge_operator → >=

gt_operator → >

eq_operator → = =

ne_operator → ~=

add_operator → +

sub_operator → -

mul_operator → *

div_operator → /