BNF Grammar for the language “**Lua**”

Shivam Bhagat 2015B5A70460H
Abhinav Kumar 2015B5A70674H
Yashdeep Thorat 2015B5A70675H
Rohan Jain 2015B4A70676H

<Program> ::= <Statements>

SEQUENTIAL STATEMENTS

```
<Statements> ::= <Statement> `;´ <Statements> | <Statement>
<Statement>   ::= <Conditional-statement> | <Loop-statement> | <Assign-statement> |
                  <Function-statement> | <expression>
```

CONDITIONAL STATEMENTS

```
<Conditional-statement> ::= `if´( <expression> ) <Statements> `end´ |
                            `if´ ( <expression> ) <Statements> `else´ <Statements> `end´
```

LOOP CONSTRUCTS

```
<Loop-statement>      ::= <while-stmt> | <repeat-stmt>
<while-stmt>          ::= `while´ `(´ <expression> `)´ `do´ <Statements> `end´
<repeat-stmt>         ::= `repeat´ <Statements> `until´ `(´ <expression> `)´
```

ASSIGNMENT STATEMENTS

```
<Assign-statement> ::= <identifier> `=´ <expression>
```

LOGICAL/ARITHMETIC EXPRESSIONS (operators according to precedence)

```
<expression>   ::= <expression> `or´ <or-term> | <or-term>
<or-term>      ::= <or-term> `and´ <and-term> | <and-term>
<and-term>     ::= <and-term> `<´ <rel-term> | <and-term> `>´ <rel-term> |
                   <and-term> `<=´ <rel-term> | <and-term> `>=´ <rel-term> |
                   <and-term> `~=´ <rel-term> | <and-term> `==´ <rel-term> | <rel-term>


<rel-term>     ::= <rel-term>  `+´ <term> | <rel-term>  `-´ <term> | <term>
<term>         ::= <term> `*´ <factor> | <term> `/´ <factor> | <term> `%´ <factor> | <factor>
<factor>       ::= `not´ <un-term> | `#´ <un-term> | `-´ <un-unterm> | <un-term>
<un-term>      ::= <un-term> `^´ <power-term> | <power-term>
<power-term>  ::= `(´ <expression> `)´ | <identifier> | <literal> | <Function-call>
```

FUNCTION DECLARATION

```
<Function-statement> ::= `function´ <identifier> <func_body>
<func_body>          ::= `(´<arguments>`)´ <Statements> <return-statement> `end´ |
                         `(´ `)´ <Statements> <return-statement> `end´

<arguments>          ::= <identifier> `,´ <arguments> | <identifier>
<return-statement>   ::= `return´ <expression-list>
<expression-list>    ::= <expression>`,´<expression-list> | <expression>
```

FUNCTION CALL

<Function-call>	::= <identifier> \`(´<arguments>\`)´ | <identifier> \`(´ \`)´

VARIABLE DECLARATION

<identifier>	::=<letter><word>
<word>	::= <letter><word> | <digit><word> | ε
<literal>	::= <number-literal> | \`true´ | \`false´
<number-literal>	::= <digit><number-literal> | <digit>
<letter>	::= \`a´ | \`b´ | \`c´ | ……… | \`z´ | \`A´ | \`B´ | \`C´ | ……… | \`Z´ | \`_´
<digit>	::= \`0´ | \`1´ | \`2´ | …. | \`9´