**Project 1: Lexical Analyzer for the**

**STAR Programming Language**

*STAR language is a small scripting language that can be used for performing basic arithmetical operations. It supports only two data types: integers and strings. Integers can only take positive values and would take the value zero when forced to take a negative value. Assignments in STAR can have arithmetic expressions with two, operands only. Complicated expressions are not allowed.*

*Basic Star Example:*

int a,b,c.

read “First:” a.

newLine.

read “Second:” b.

newLine.

c is a+b.

c is c/2.

write “Result:” c.

*Lexical rules for the programming language STAR are as follows:*

**1**- **Identifiers:**

* Maximum identifier size is 10 characters. If you use an identifier larger than that, the lexical analyzer issues an error message.
* STAR is case sensitive and all the keywords are lower case. Identifiers can use both upper/lower case.
* Identifiers start with an alphabetic character (a letter) and are composed of one or more letters/digits/\_ (underscore)
* Example Token: Identifier(my\_var\_1)

**2**- **Integer constants:**

* Maximum integer size is 8 digits. If you use an integer value longer than that, the lexical analyzer issues an error message.
* Negative values are not supported.
* Example Token: IntConst(1290)

**3**- **Operators:**

* Valid operators of the language are +,-,\*,/
* Example Token: Operator(+)

**4**- **Brackets:**

Brackets are used for creating blocks of code for the loops.

* LeftCurlyBracket: { RightCurlyBracket: }
* Example Token: LeftCurlyBracket

**5**- **String constants:**

* String constants of STAR are delimited by double quotes (ASCII code 34)as in “this is a string“.
* Maximum string constant size is 256 characters
* String constants cannot contain the double quote character. when you reach one, the string terminates.
* If a string constant cannot terminate before the file end, there should be a lexical error issued.
* If a string constant exceeds 256 characters there should be a lexical error issued.
* Example Token: String(“Hello world!“)

**6- Keywords:** (all are case sensitive)

* Keywords are: int, text, is, loop, times, read, write, newLine
* Example Token: Keyword(int)

**7- End of line:** .

Period character ”.” is used to mark the end of line.

* Example Token: EndOfLine

**8- Comma:**

* Comma ‘,’ is used as a separator whenever multiple attributes are possible in any command. These include multiple declarations and write statements.

**9- Comments:** Anything between /\* and \*/ is a comment.

* If a comment fails to terminate before the file end, there should be a lexical error issued.
* Comments are just like blank space and they create no tokens.

**------------------------------------------------------------------------**

**1st Project Defn: The Program should accept a source file called code.sta and produce a text file named as code.lex that contains all the tokens listed one after the other.**

**Example: The line**

c is c/2.

Would produce:

Identifier(c)

Keyword(is)

Identifier(c)

Operator(/)

EndOfLine

***Teams:*** 1..2 students

***Deadline:*6th May 2024**

***Deliverables:***

1. Source code (Hard copy)
2. Executable file(s) (Electronic)
3. Demonstration (Please make an appointment with one of your lab instructors.)