Languages and Paradigms Chapter 4 Project

**Q1)** Given the following BNF:

<program> -> <stmt>{<stmt>}

<stmt> -> <assign> | <declaration>

<assign> -> <var> = <expr>

<expr> → <term> { (+ | -) <expr>}

<term> → <factor> { (\* | /) <term>}

<factor> → <var> | <const> | ( <expr> )

<var> -> <firstChar> {<restChars>}

<firstChar> -> A | … | Z | a | … | z | \_

<restChars> → <firstLetter> | <digit>

<digit> → 0 | 1 | … | 9

<const> → <digit>{<digit>}

<decl> -> int <var>{, <var>}

Write the programs on the slides:

* Globals.cpp: contains global variables declarations and the method errMsg that display any error messages.
* Lex.cpp is the tokenizer that converts a program into tokens.
* symbolTable.cpp store the symbol table functions and structure.
* Bnf is the parser that displays syntax errors if it exists.

1) **create a new file** called **prg.in**, then type the following code and save it:

int A, B, C, D=2

A = 3 + 5

B = A \* 10 + 3

C = 5 + A / D + 3

In the main function, set the filename to “prg.in” and run the parser. Paste a screenshot of the output of the parser.

2) Create a new file called prg1.in and type the following program:

int A, B, C, D=2

A = 3 + 5

B = A \* 10 + 2

C = 5 + A / D 3

Note the error at line 4 of the program.

Run the parser and paste a screenshot of the output.

3) Create a new file called prg1.in and type the following program:

int A, B, C, D=2

A = 3 + +

B = A \* 10 + 5

C = 5 + A / D + 3

Note the error at line 2. Run the parser and paste a screenshot of the output.

**Q2)** write a tokenizer (Q2Tokenizer.cpp) and parser (Q2Parser.cpp) for the following BNF:

S -> <A><C><B> | <A>

<A> -> a<A> | a

<B> -> b<B> | b

<C> -> c

Add any necessary files like globals.cpp and symbolTable.cpp.

Create a new file ‘prg3.in’ that includes the following lines in **figure 1**:

aacbb

aaabb

**Figure 1**, program test for Q2

Run your Q2Parser on the following ‘prg3.in’, and attach screenshots.

The first thing you need to do is to change the enum to include the tokens which are: A, B, C, ENDFILE, UNKNOWN. Then, the prt function should change to display the tokens. After that the lookupKeyword should change to get the token that corresponds to every symbol ‘a’, ‘b’ or ‘c’. Finally, the tokenizer and parser must be changed to every case: a’s, b’s and c.

**What to submit?**

* Create a github repository and upload all your programs, screenshots and this document with your answers.
* Submit on Canvas the github repository link.