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> -> <**firstChar**> | <digit>
<digit> -> 0 | 1 | … | 9
<const> -> <digit>{<digit>}
<decl**aration**> -> 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
stores the symbol table functions and structure.

bnf.cpp
is the parser that displays syntax errors if it exists.

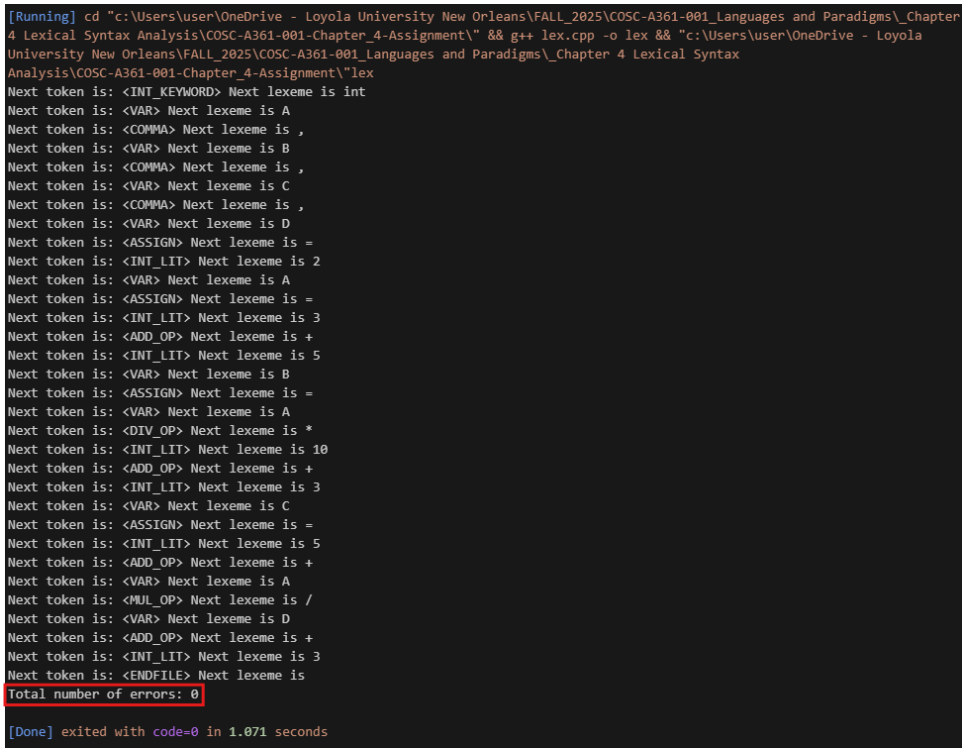
Programs in the following repository (run the lex.cpp to execute the program):
https://github.com/smacias116/COSC-A361-001-Chapter_4-Assignment

1) **Create a new file** called **prg1.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 "prg1.in" and run the parser. Paste a screenshot of the output of the parser.

The full BNF has been parsed with no issues:



2) Create a new file called **prg2.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.

The parser has detected an issue in line 4. Following is the explanation.
The parser works fine up to a point, but it fails after character D on line 3 due to a space.
At "A / D 3", the expression is detected as "<factor> { (* | /) <term>}", due to the operator "/".

"A" <factor> → <var> → <firstChar> → A
"/" (* | /)
"D 3" <term> → <factor> { (* | /) <term>}.
As there's no operator "*" or "/", term "D 3" is a <factor>
The option at this point are 3: <factor> → <var> | <const> | (<expr>)
1. ~~(<expr>)~~ Not a solution
2. ~~<const> → <digit>{<digit>}~~ Not a solution
3. <var> → <firstChar> {<restChars>} Let's see:
"D" <firstChar> → A | … | Z | a | … | z |_+ → D
"3" <restChars> → ~~<firstChar> |~~ <digit> → 0 | 1 | … | 9 → 3
The issue is that D and 3, there's a space and the parser cannot deal with it.

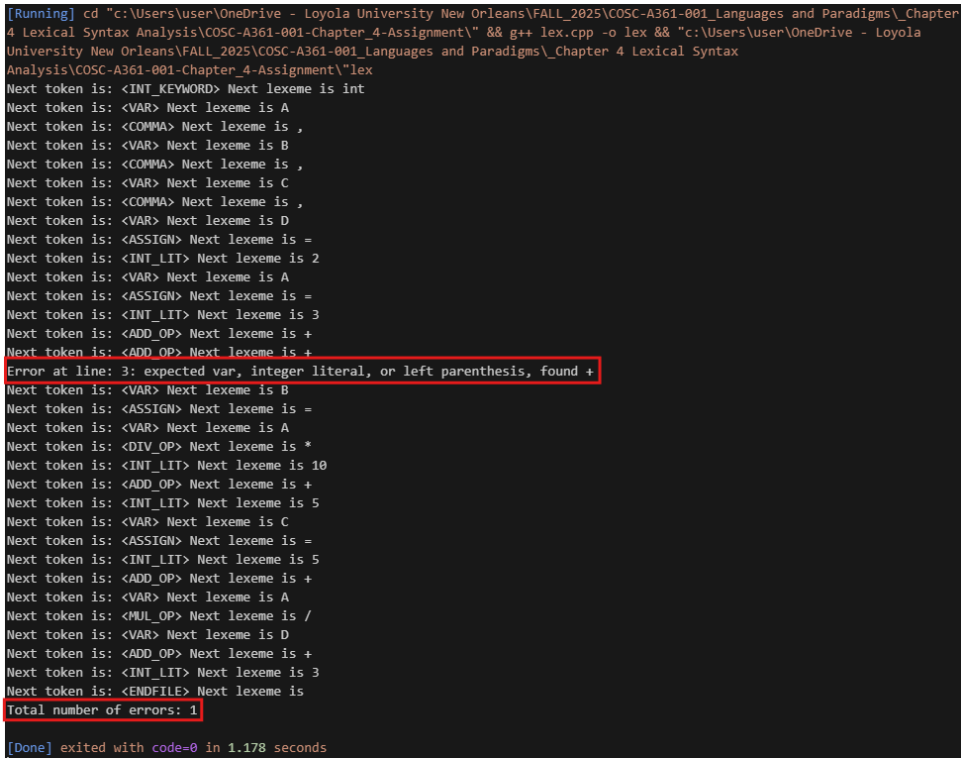
```
[Running] cd "c:\Users\user\OneDrive - Loyola University New Orleans\FALL_2025\COSC-A361-001_Languages and Paradigms\_Chapter
4 Lexical Syntax Analysis\COSC-A361-001-Chapter_4-Assignment\" && g++ lex.cpp -o lex && "c:\Users\user\OneDrive - Loyola
University New Orleans\FALL_2025\COSC-A361-001_Languages and Paradigms\_Chapter 4 Lexical Syntax
Analysis\COSC-A361-001-Chapter_4-Assignment\"lex
Next token is: <INT_KEYWORD> Next lexeme is int
Next token is: <VAR> Next lexeme is A
Next token is: <COMMA> Next lexeme is ,
Next token is: <VAR> Next lexeme is B
Next token is: <COMMA> Next lexeme is ,
Next token is: <VAR> Next lexeme is C
Next token is: <COMMA> Next lexeme is ,
Next token is: <VAR> Next lexeme is D
Next token is: <ASSIGN> Next lexeme is =
Next token is: <INT_LIT> Next lexeme is 2
Next token is: <VAR> Next lexeme is A
Next token is: <ASSIGN> Next lexeme is =
Next token is: <INT_LIT> Next lexeme is 3
Next token is: <ADD_OP> Next lexeme is +
Next token is: <INT_LIT> Next lexeme is 5
Next token is: <VAR> Next lexeme is B
Next token is: <ASSIGN> Next lexeme is =
Next token is: <VAR> Next lexeme is A
Next token is: <DIV_OP> Next lexeme is *
Next token is: <INT_LIT> Next lexeme is 10
Next token is: <ADD_OP> Next lexeme is +
Next token is: <INT_LIT> Next lexeme is 2
Next token is: <VAR> Next lexeme is C
Next token is: <ASSIGN> Next lexeme is =
Next token is: <INT_LIT> Next lexeme is 5
Next token is: <ADD_OP> Next lexeme is +
Next token is: <VAR> Next lexeme is A
Next token is: <MUL_OP> Next lexeme is /
Next token is: <VAR> Next lexeme is D
Next token is: <INT_LIT> Next lexeme is 3
Error at line: 4: Variable expected, found 3
Next token is: <ENDFILE> Next lexeme is
Total number of errors: 1

[Done] exited with code=0 in 1.081 seconds
```

3) Create a new file called **prg3.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.

The parser has detected an issue in line 3. Like before, there's no solution for this grammar As <term> { (+ | -) <expr>}, the parser expects <expr> after the operator +.
However, there's no BNF for a <expr> which allows "+" without setting a <term> before the operator (<expr> → <term> { (+ | -) <expr>})



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 'prg4.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.

The parser has detected an issue in line 2: After any lexeme of a's only a (token <A>) only a single c (toke <C>) is allowed.

```
[Running] cd "c:\Users\user\Desktop\T_Desktop\" && g++ Tlex.cpp -o Tlex && "c:\Users\user\Desktop\T_Desktop\"Tlex
Next token is: <A> Next lexeme is aa
Next token is: <C> Next lexeme is c
Next token is: <B> Next lexeme is bb
Next token is: <A> Next lexeme is aaa
Next token is: <B> Next lexeme is bb
Error at line: 2: After token <A>, expected token <C>
Next token is: <ENDFILE> Next lexeme is
Total number of errors: 1

[Done] exited with code=0 in 2.179 seconds
```

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.