ANKARA UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

2024-2025 FALL SEMESTER

COM2043 PROGRAMMING LANGUAGE CONCEPTS

PROJECT-1

"Lexical Analysis and Parsing"

Deadline: 27 December 2024 (**You can only write in c programming language**)

Objective:
To learn the lexical and syntax analysis of context-free grammars using lex&yacc and to design a syntax analyzer for your own programming language.

Background Information:
Read "lexyacc.pdf" in the course homepage.

Implementation:
You will design a lexical and syntax analyzer for your own programming language. Give some rules for your programming language and call it as **MPL** (**M**y **P**rogramming **L**anguage). **For example**, your rules **may be** as;

- all programs must start with "begin" and end with "end"
- all commands must be ended by "**;**"
- all variables must be declared just after "begin" in the following format;
  int: i num;
  float: fl;
- three types exist; "int", "float", "char"
- variables names may be any alphanumeric string, but they must start with a letter
- statements between "begin" and "end" can be either a variable declaration or an assignment
- an assignment includes four type of operators; +,-,*,/.
- the number of variables or constants in an expression is unlimited
- the presedence of operators given as.....
- .....
- .etc

The minimum requirements are: a well-defined regular grammar for variable names, rules for variable declerations, at least 4 arithmetic, 2 logical and 2 comparison operators with their presedence and associativity rules, at least one conditional statement (like if, switch) and at least one loop structure (like for, while, do-while, repeat-until...). Instead of using the grammar for C language, try to bring new ideas for your language.
After writing CFG rules for your language, design and implement a syntax analyzer for it. This analyzer will include a lex source (e.g ***mpl.l***), a yacc source (e.g ***mpl.y***) and example inputs (source codes written in your language, e.g ***myprog.mpl***).

To test your analyzer, follow the steps below;

```
$ lex mpl.l
$ yacc mpl.y
$ gcc –o mpl y.tab.c –lfl
$ mpl < myprog.mpl
```

In the last step, if there is no syntax error in ***myprog.mpl***, the program should give an "OK" message, otherwise, a "syntax error" message. Check your program for correct and wrong inputs.

Note: Do not deal with any semantic rule, just give syntax rules!

Report:

Your report should include;

(1) Description of the rules which you designed (in your own words)

(2) Details of your grammar in BNF or EBNF notation,

(3) Code descriptions,

(4) The content of ***mpl.l***, ***mpl.y*** files and an example ***myprog.mpl*** file,

(5) Any explanation which clarifies that you made your assignment with your own effort