Course: Principles of Programming Languages **Course ID:** IT092IU

INTERNATIONAL UNIVERSITY - VIETNAM NATIONAL UNIVERSITY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

PPL Lab 2

Parser Exercise

NAME	STUDENT ID
Vũ Nhật Duy	ITITIU17047

Write lexer rules in "BKIT.g4" that can accept the following tokens.
Example 1:
Program accepts expressions that are integers or identifiers.

Example 2:

- a + b
- Calculating must be performed from left to right.

Exercise 1:

- a + b
- a – b
- Calculating must be performed from left to right.

Exercise 2:

- a + b
- a – b
- a * b
- a / b
- Operator '*' and '/' have higher priority than '+' and '-'.
- Calculating must be performed from left to right in case operators have the same priority.

Solution:
Exercise 1:
With the provided Parser Code files, changing the program part to meet the requirements in Exercise1.g4

```
program
    : ( expression )* EOF
    ;

expression
    : expression '+' term
    | expression '-' term
    | term
    ;

term
    : Integer
    | Id
    ;

Integer: [0-9]+ ;
Id: [a-z]+ ;
```

```
program
    : ( expression )* EOF
    ;

expression
    : expression '+' term
    | expression '-' term
    | term
    ;

term
    : Integer
    | Id
    ;
```

This ensures + and - are left-associative so that the calculation is performed left to right.
Then after completing, running the code to compile and test with the testcase1.txt
35 + 47
7 + 9 + a
13 - 2
11 - 2 - 2

```
PS C:\MyFolder\Studies\Principles of Programming Language\Lab\Day 2\Parser\LexerGenerator> python gen.py Exercise1.g4
```

```
PS C:\MyFolder\Studies\Principles of Programming Language\Lab\Day 2\Parser\FinalProgram> python run_Exercise1.py testcase1.txt
successful
```

Exercise 2:

With the provided Lexer Code files, changing the program part to meet the requirements in Exercise2.g4

```
program
    : ( expression )* EOF
    ;

expression
    : expression '+' term
    | expression '-' term
    | term
    ;

term
    : term '*' factor
    | term '/' factor
    | factor
    ;

factor
    : Integer
    | Id
    ;

Integer: [0-9]+ ;
Id: [a-z]+ ;
```

program
: (expression)* EOF
;

expression
: expression '+' term
| expression '-' term
| term
;

term
: term '*' factor
| term '/' factor
| factor
;

factor
    : Integer
    | Id
    ;

Because * and / have higher priority, the rules for term involving * and / appear before the rules for expression involving + and -.

All operators are left-associative, ensuring calculation to be performed from left to right in case operators have the same priority.

Then after completing, running the code to compile and test with the testcase2.txt

35 + 47

7 + 9 + a

13 - 2

11 - 2 - 2

a * b

a * b / c + d - e

```
PS C:\MyFolder\Studies\Principles of Programming Language\Lab\Day 2\
Parser\LexerGenerator> python gen.py Exercise2.g4
```

```
PS C:\MyFolder\Studies\Principles of Programming Language\Lab\Day 2\P
arser\FinalProgram> python run_Exercise2.py testcase2.txt
successful
```