LAB 3: Parser (Association & Precedence)

Vo Hoang Nhat Khang

Examples:

Example 1:

Consider the expression:

```
expr1 + expr2 - expr3
```

where `expr1`, `expr2` and `expr3` are identifiers or another expression. Identifiers are strings consisting of alphanumeric characters (both uppercase and lowercase letters) and may include underscores (_). Write parser rules to match this expression. Note that we want to proceed the addition of expressions before proceeding the subtraction of expressions, left-to-right associativity.

Example 2:

Consider the expression:

```
expr1 * (expr2 - expr3)
```

where `expr1`, `expr2`, and `expr3` represent arbitrary expressions. An `expr` can be either an alphanumeric characters (both uppercase and lowercase letters) or an integer. Write parser rules to match this expression.

Exercises:

Exercise 1:

Consider the expression:

```
NUM1 * NUM2 + NUM3
```

where `NUM1`, `NUM2`, and `NUM3` represent integer values. Write parser rules to match this expression. Ensure that multiplication (*) has higher precedence than addition (+).

Exercise 2:

Consider the expression:

```
expr1 && expr2 || expr3 && expr4
```

where `expr1`, `expr2`, `expr3`, and `expr4` represent boolean values, or another expression. Write parser rules to match this expression. Ensure correct association and precedence for logical `AND` (&&) and logical `OR` (||), with `OR` having higher precedence than `AND`. Operation `OR` is left-to-right associativity, but `AND` operation is right-to-left associativity.