PPL - Spring 2025/Lab report

**Lab 8 - Abstract Syntax Tree and Code Runner - Report**

**Students:**

1. Phan Trần Thanh Huy – ITCSIU22056

**1 Overview**

*In this section, briefly listing the issues that the class today covers.*

# Results

**Q.1.** Trace the process of generating an AST for the input expression 3 + 5 \* 2. Check the resulting AST with your tracing in the given code. Sketch the workflow of your trace from input to the output AST (i.e., step-by-step).

Step 1: Run the command “python run.py test” to generate.

Step 2: Get the text from textcase.txt

For example: “3 + 5 \* 2”

Step 3: Running the Sample.g4 to create the rule before testing.

Step 4: It will separate the text into token

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

, check token syntax (skip off-channel tokens)

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

, check grammar and sentence syntax.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Then returning the parser. The general flow is shown below:  
Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, số

Mô tả được tạo tự động

Step 5: Based on the parser, it will analyze and generate the parse tree.



Step 6: Based on the parse tree, generate the abstract syntax tree (AST) by ASTGeneration library.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Step 7: Based on the AST, we will do the logic calculations and return the result from the CodeRunner file:

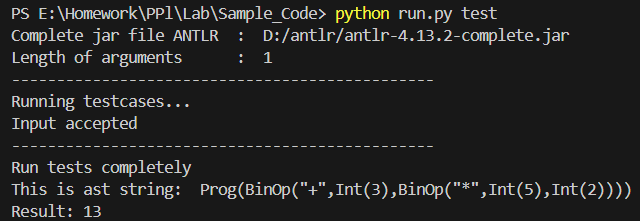
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

The result will show:



**Q.2.** From Q.1, explore and explain how the grammar ensures that operators \* and/have higher precedence than + and- operators?

The code below:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Overall explanation:

Step 1: The flow will run and check whether an expression or a term base on the operations. The expression will be “+”, “-“, and the term will be “\*”, “/”.

The BinOp is the function which will display the AST.

Ảnh có chứa ảnh chụp màn hình, Phần mềm đa phương tiện, văn bản, phần mềm

Mô tả được tạo tự động

So when it is an expression, the BinOp will display the sign, the number and its term if there are any terms. Otherwise, it will display the term. Likewise, the term will display the sign and 2 numbers inside it. If it is only factor, it will continue run the visitFactor, or integer in the tree.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

The result:



Moreoere, we will have a test below:

This is the initial code:  
Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

I will swap the mul and div to the expression, add and sub to the term below:  
Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động  
Technically, swapping the operations of expression and term. On the other words, it will prior the “+”, “-“ first, then is “\*”, “/”

So the result:



Conclusion: The explanation is satisfied.

**Q.3.** Extend the code in Q.1 to include the following scenarios. For each, test with 3 different input samples. Capture the code you made modification to handle the problem, input samples and corresponding output.

Step 1: Add modulo and exponentiation into Sample.g4

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

Also run python run.py gen to generate lexer and parser.

Step 2: Because it is modulo and exponentiation, it will be term. Modifying the visitTerm in ASTGeneration.py

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Step 3: Add the logic calculation in visitBinaryOp inside CodeRunner.py

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

*Q.3.1.* Scenario 1: include the modulo operator (%).

Sample input: 10 % 3.

Result:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Sample input: 10 % 3 + 5 \* 2.

Result:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Sample input: 10 % 3 + 5 % 2.

Result:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

*Q.3.2.* Scenario 2: include the ˆ operator for exponentiation.

Sample input: 2 ^ 3.

Result:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Sample input: 2 ^ 3 + 5 \* 2.

Result:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

Sample input: 2 ^ 3 + 5 ^ 2.

Result:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

## CSE, HCMIU - VNU 1