GitHub: https://github.com/JuheonChu/lumber_84

84 LUMBER

Code Modernization and Documentation

Dickinson CS student research Group

Outline

- **Project 1:** Modernization of the corporate codebase via the conversion of C codebase into Java, Python, and .NET

  by Boosung, Maximo, William, Shahir, Fox.

- **Project 2:** Scan the corporate codebase and produce documentation and specifications that can facilitate training new developers.

  by John, Pranav, Ahnaf, Youssif

Project Scheme

- Aims to discover the optimal approach to initiate them.

- Divide the team into multiple subgroups to experiment with various approaches.

Subgroups

Subgroup 1: Maximo, Shahir, Fox

Design and implement a backend system to support code analysis.

Subgroup 2: John, Pranav, Ahnaf, Youssif

Construct an NLP pipeline to generate documentation based on codebases.

Subgroup 3: Boosung, William

Experimentation with existing open-source platforms such as Copilot.

Subgroup 1 (Maximo & Shahir & Fox)

Subgroup 2 (John, Pranav, Ahnaf, Youssif)

Approach

- Construct an NLP pipeline to generate documentation based on the input codebases.

Objective

- Address Project 1 & Project 2 details as denoted in Outline.

Motivation

- Delve into NLP, Programming Language Structure, and low-level programming languages (e.g. Assembly language, Low C/C++, BASIC)

Project settings

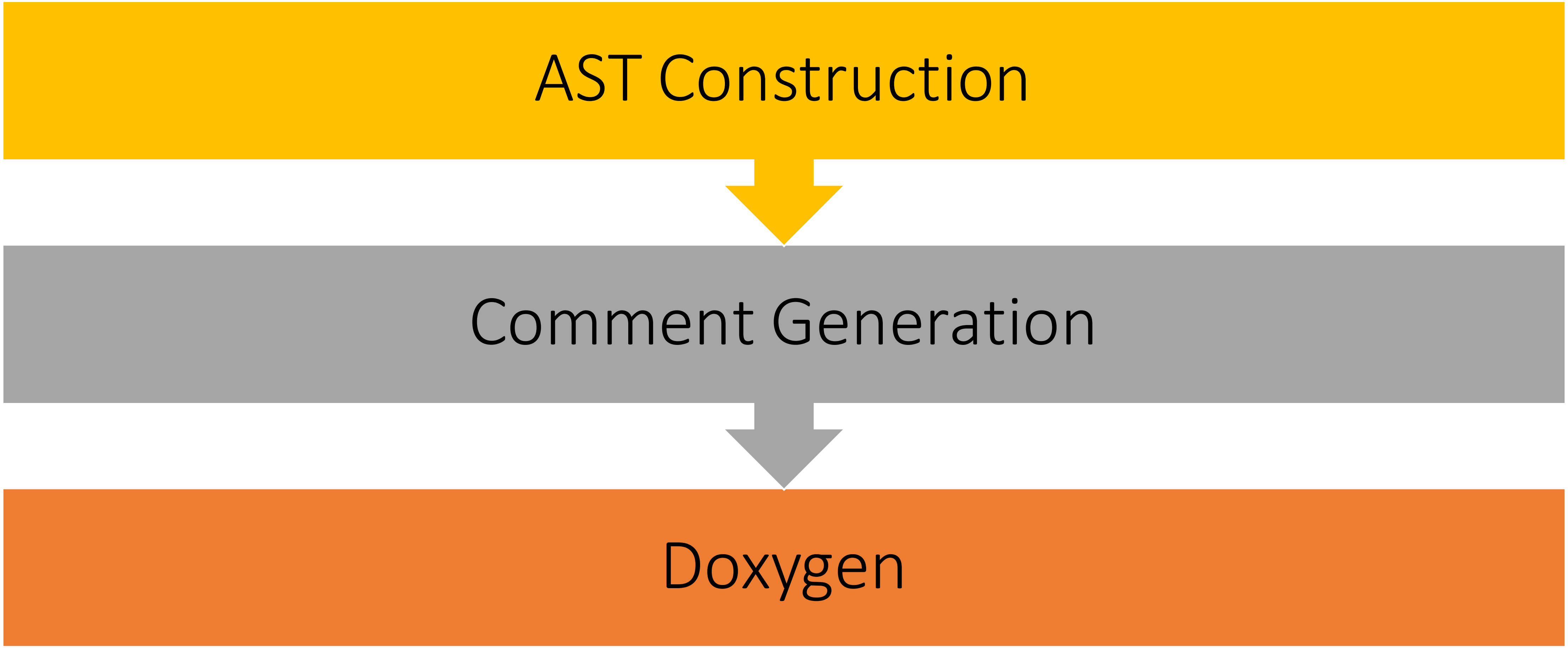
Software: WSL Ubuntu (John), Multipass (Ahnaf), Pranav & Youssif (Docker, Jupyter), NLTK, graphviz

Programming Languages: BASIC, C/C++, Python, Java, ANTLR4

Deep Learning models: Hugging Face Transformer language models (e.g. GPT, BERT)

Subgroup Meeting: Weekly subgroup meeting to discuss the direction of our progress.

Project Workflow



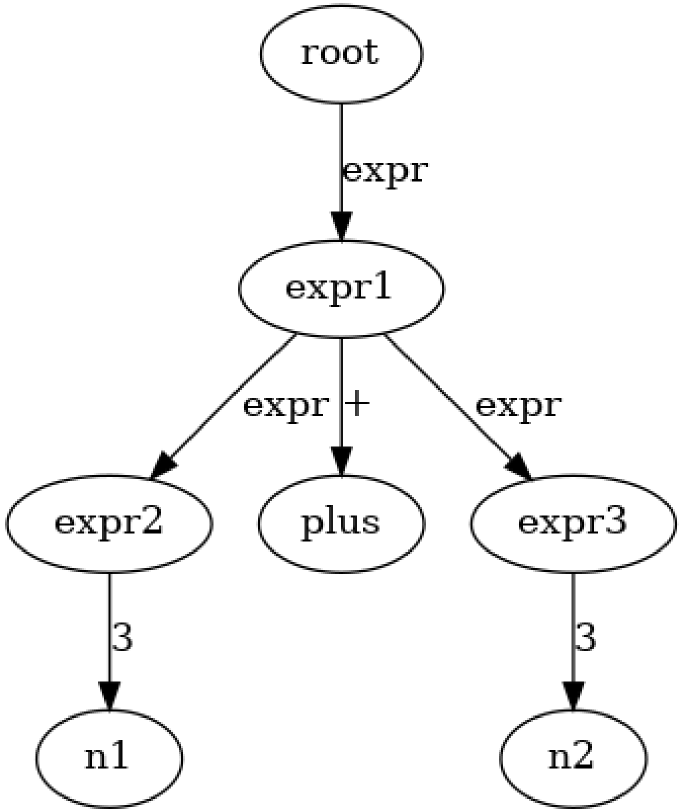
AST Construction

Comment Generation

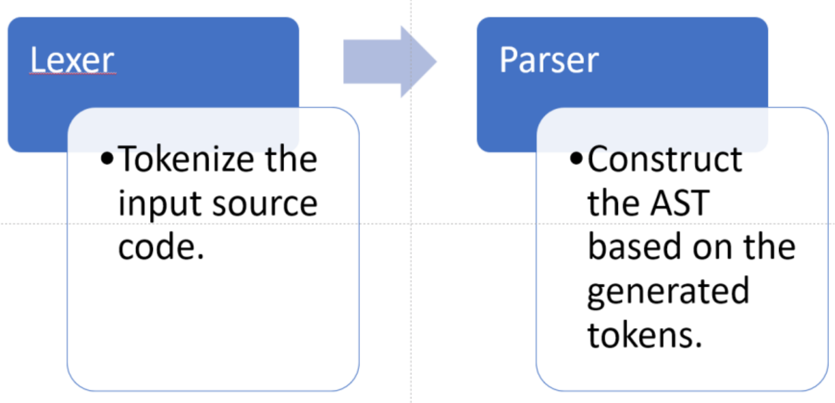
Doxygen

Abstract Syntax Tree (AST) Construction (John & Ahnaf)

- We aim to construct the Abstract Syntax Tree to store tokens that implicate pivotal program logics.



Abstract Syntax Tree


Lexer

- Tokenize the input source code.

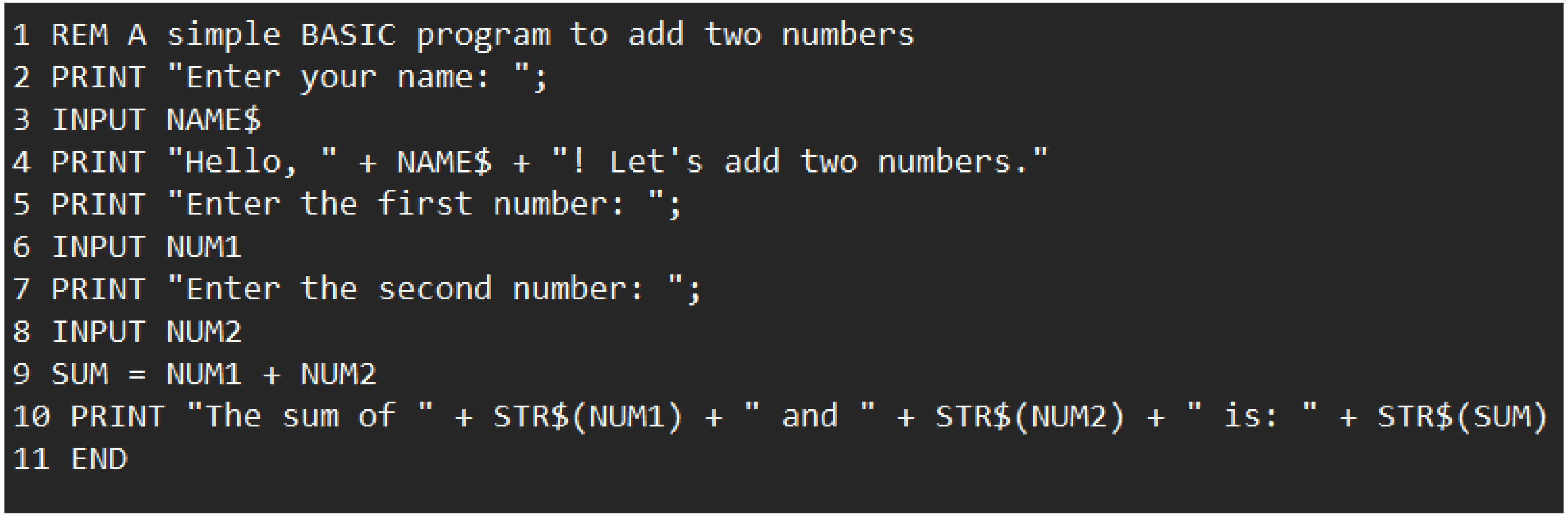
Parser

- Construct the AST based on the generated tokens.

ANTLR4 Pipeline

Grammar Definition (BASIC.g4)

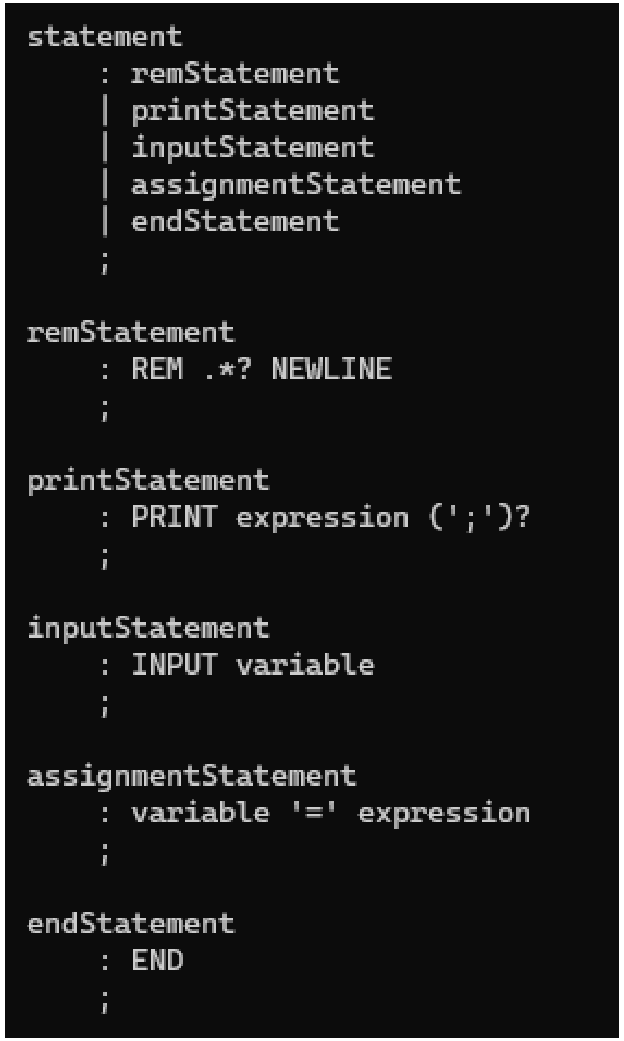
- Define a regular expression that can analyze the BASIC programming framework to construct the Abstract Syntax Tree.
- Experimented with a simple BASIC code.



```
1 REM A simple BASIC program to add two numbers
2 PRINT "Enter your name: ";
3 INPUT NAME$
4 PRINT "Hello, " + NAME$ + "! Let's add two numbers."
5 PRINT "Enter the first number: ";
6 INPUT NUM1
7 PRINT "Enter the second number: ";
8 INPUT NUM2
9 SUM = NUM1 + NUM2
10 PRINT "The sum of " + STR$(NUM1) + " and " + STR$(NUM2) + " is: " + STR$(SUM)
11 END
```

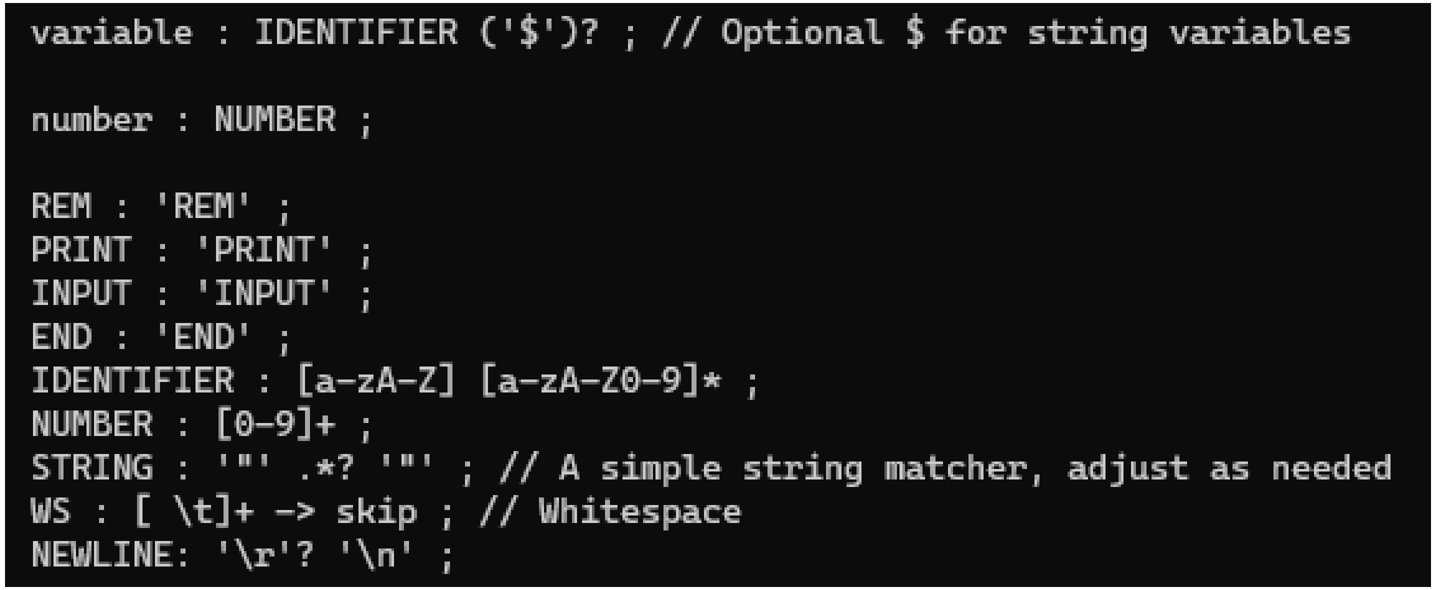
Example Code

Grammar



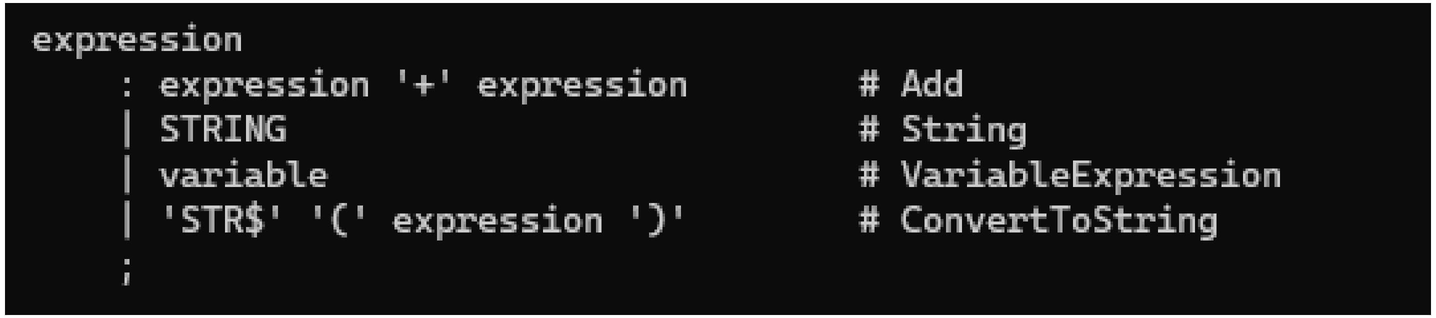
```
statement
    : remStatement
    | printStatement
    | inputStatement
    | assignmentStatement
    | endStatement
    ;

remStatement
    : REM .*? NEWLINE
    ;

printStatement
    : PRINT expression (';')?
    ;

inputStatement
    : INPUT variable
    ;

assignmentStatement
    : variable '=' expression
    ;

endStatement
    : END
    ;
```

Statement



```
variable : IDENTIFIER ('$')? ; // Optional $ for string variables

number : NUMBER ;

REM : 'REM' ;
PRINT : 'PRINT' ;
INPUT : 'INPUT' ;
END : 'END' ;
IDENTIFIER : [a-zA-Z] [a-zA-Z0-9]* ;
NUMBER : [0-9]+ ;
STRING : '"' .*? '"' ; // A simple string matcher, adjust as needed
WS : [ \t]+ -> skip ; // Whitespace
NEWLINE: '\r'? '\n' ;
```

Variable



```
expression
    : expression '+' expression         # Add
    | STRING                            # String
    | variable                          # VariableExpression
    | 'STR$' '(' expression ')'         # ConvertToString
    ;
```

Expression

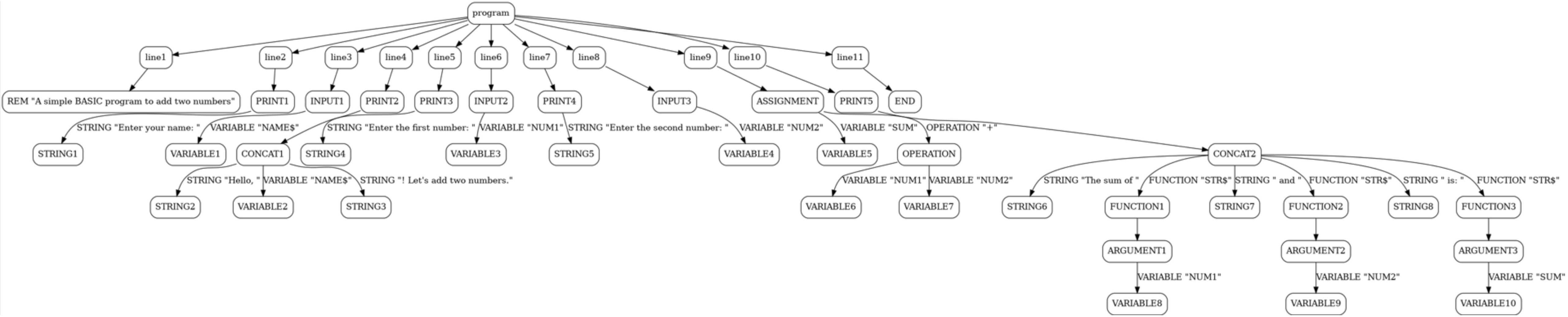
Tokenized AST

- As a result of ANTLR4 pipeline, we are able to get a tokenized Abstract Syntax Tree.

```
parsed_output = '''(program (line (number 1) (statement
(remStatement REM A simple BASIC program to add two numbers \n 2 PRINT
"Enter your name: " ; \n 3 INPUT NAME $ \n 4 PRINT "Hello, " + NAME $ +
"! Let's add two numbers." \n 5 PRINT "Enter the first number: " ; \n 6
INPUT NUM1 \n 7 PRINT "Enter the second number: " ; \n 8 INPUT NUM2 \n 9
SUM = NUM1 + NUM2 \n 10 PRINT "The sum of " + STR$ ( NUM1 ) + " and " +
STR$ ( NUM2 ) + " is: " + STR$ ( SUM ) \n 11 END \n)) \n) <EOF>)'''
```

Tokenized AST

AST Contsruction

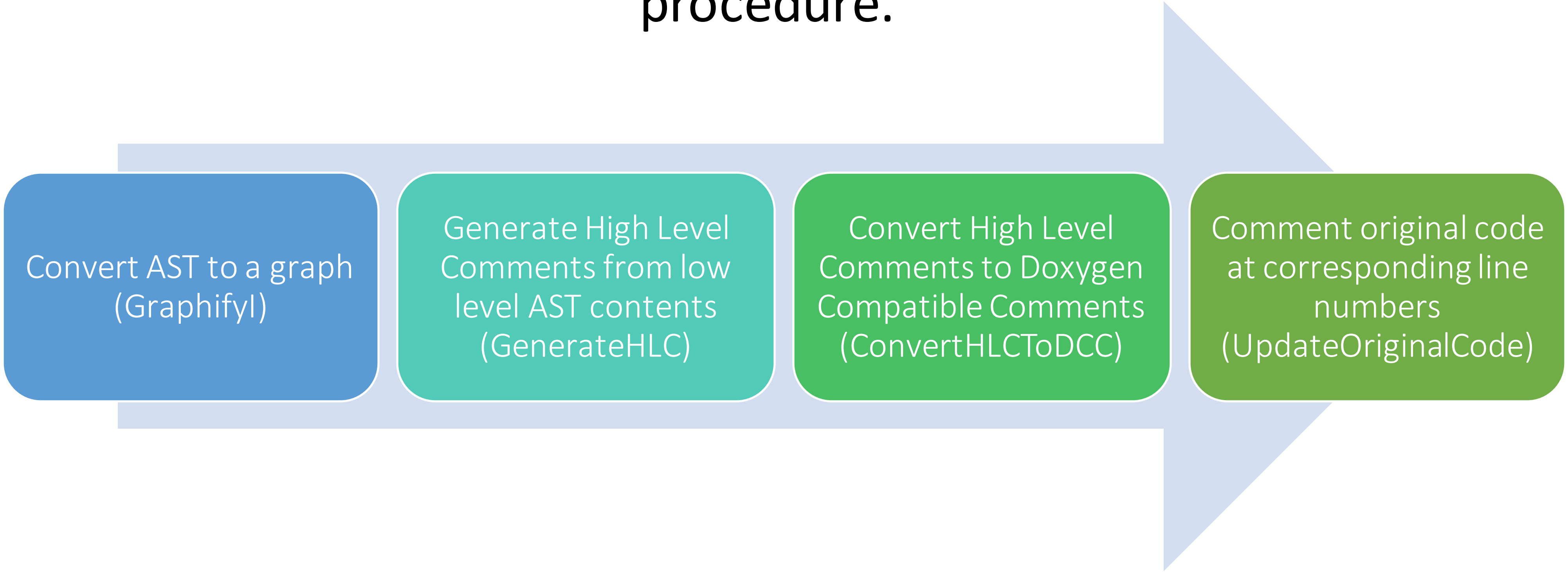


Challenges

- Standardize the Python code to construct the AST based on how Pranav & Youssif formatted (Slide 17).
- The bullet point above will be our task next week.

Comment Generation (Pranav & Youssif)

Assuming the existence of an AST in a desired format (with proper indentation and line numbers), we seek to adopt the following procedure.



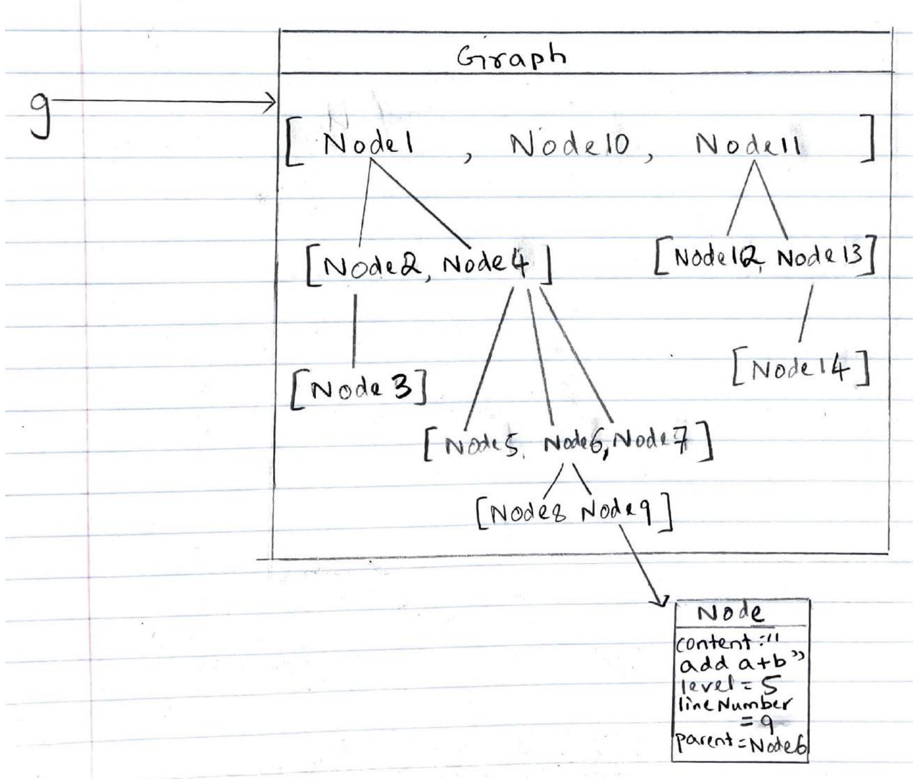
Convert AST to a graph (GraphifyI)

Generate High Level Comments from low level AST contents (GenerateHLC)

Convert High Level Comments to Doxygen Compatible Comments (ConvertHLCToDCC)

Comment original code at corresponding line numbers (UpdateOriginalCode)

Step 1: Converting AST to a Graph (GraphifyI)

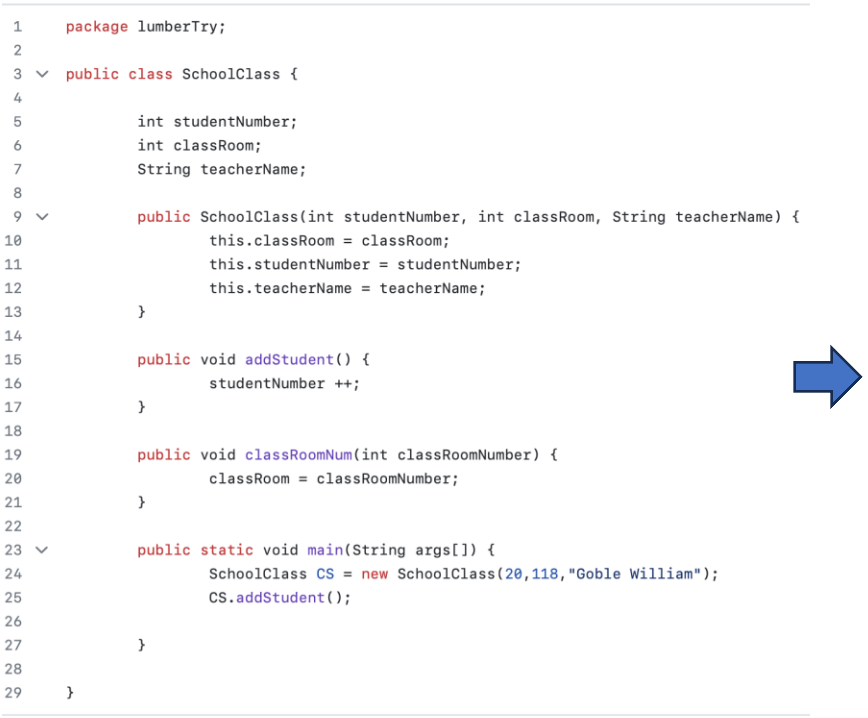


- Method named GraphifyI operates on an AST file in txt format, converting every line to a custom Node in the format

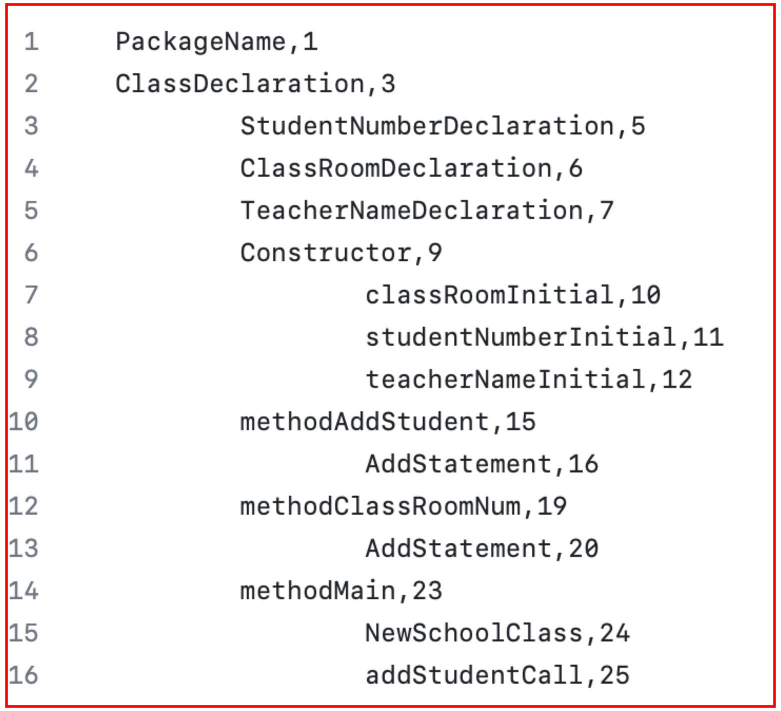
"__content,lineNumber" to node object Node(content, level, lineNumber, parent) (here, ___ denotes number of indentations done, which determines the level of the node).

- Subsequently, GraphifyI adds these nodes to a Graph object 'g', which is shown in the object diagram on the left

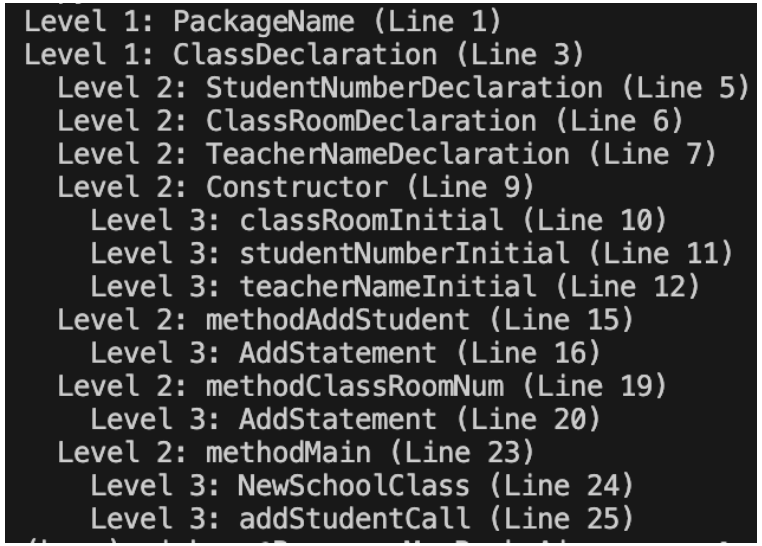
Code can be found at https://github.com/JuheonChu/lumber_84/tree/main/pranav



```java
package lumberTry;

public class SchoolClass {

        int studentNumber;
        int classRoom;
        String teacherName;

        public SchoolClass(int studentNumber, int classRoom, String teacherName) {
                this.classRoom = classRoom;
                this.studentNumber = studentNumber;
                this.teacherName = teacherName;
        }

        public void addStudent() {
                studentNumber ++;
        }

        public void classRoomNum(int classRoomNumber) {
                classRoom = classRoomNumber;
        }

        public static void main(String args[]) {
                SchoolClass CS = new SchoolClass(20,118,"Goble William");
                CS.addStudent();

        }

}
```



```
PackageName,1
ClassDeclaration,3
        StudentNumberDeclaration,5
        ClassRoomDeclaration,6
        TeacherNameDeclaration,7
        Constructor,9
                classRoomInitial,10
                studentNumberInitial,11
                teacherNameInitial,12
        methodAddStudent,15
                AddStatement,16
        methodClassRoomNum,19
                AddStatement,20
        methodMain,23
                NewSchoolClass,24
                addStudentCall,25
```



```
Level 1: PackageName (Line 1)
Level 1: ClassDeclaration (Line 3)
  Level 2: StudentNumberDeclaration (Line 5)
  Level 2: ClassRoomDeclaration (Line 6)
  Level 2: TeacherNameDeclaration (Line 7)
  Level 2: Constructor (Line 9)
    Level 3: classRoomInitial (Line 10)
    Level 3: studentNumberInitial (Line 11)
    Level 3: teacherNameInitial (Line 12)
  Level 2: methodAddStudent (Line 15)
    Level 3: AddStatement (Line 16)
  Level 2: methodClassRoomNum (Line 19)
    Level 3: AddStatement (Line 20)
  Level 2: methodMain (Line 23)
    Level 3: NewSchoolClass (Line 24)
    Level 3: addStudentCall (Line 25)
```

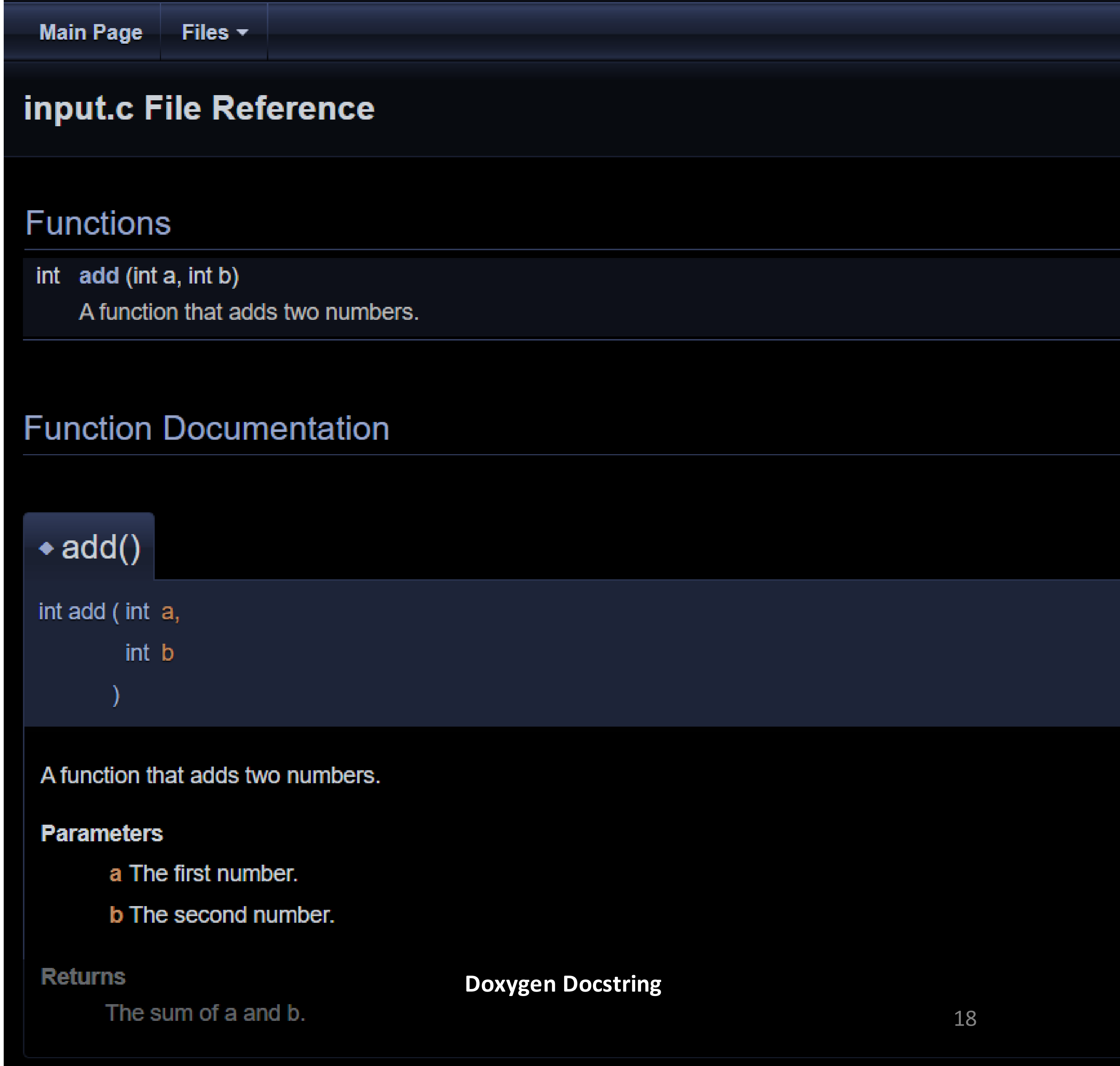
Currently, we have obtained successful results by testing GraphifyI on a dummy AST

SWOT Analysis of GraphifyI

- Strengths: Captures both the hierarchy and sequence of program instructions via a custom data structure tailor-made for ASTs!
- Weaknesses: Requires AST to be formatted in a particular format
- Opportunities: Code for GraphifyI can be generalized for a wide range of files where both sequence and hierarchy are important
- Threats: Coming up with an efficient traversal method may be difficult

Document Generation

- When DocString is generated at Step 2, we will be able to generate a documentation that outlines the code.



input.c File Reference

Functions

int add (int a, int b)
A function that adds two numbers.

Function Documentation

◆ add()

```
int add ( int  a,
          int  b
        )
```

A function that adds two numbers.

Parameters
a The first number.
b The second number.

Returns
The sum of a and b.

Doxygen Docstring

Progress

John & Ahnaf are researching regular expressions to define a grammar that tokenizes corporate BASIC codebases & constructs AST.

Pranav & Youssif are nailing down to construct the GraphifyI function and further investigating to apply Spacy library to generate Doxygen-formatted Docstrings.

Subgroup 3 (William and Boosung)

Subgroup 3 (William and Boosung)

Approach

- Propose a translation pipeline using Copilot or GPT4 specified to the use-case per codebase.

Objective

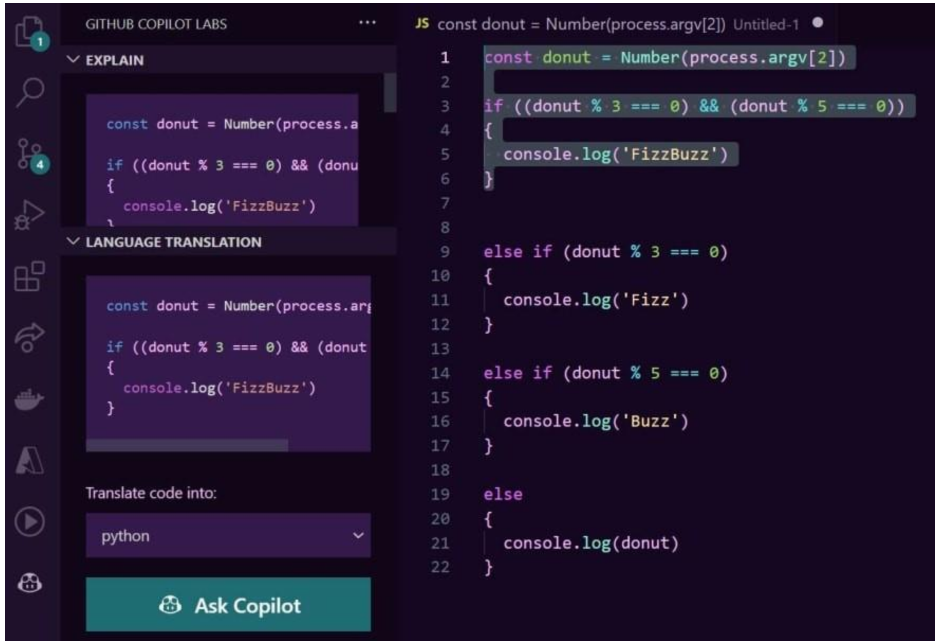
- Address Project 1 as denoted in Outline.

Motivation

- Microsoft Copilot is the standard used in the industry, used by over 1M developers and 20,000 organizations

Copilot Labs and Power Platform Extension

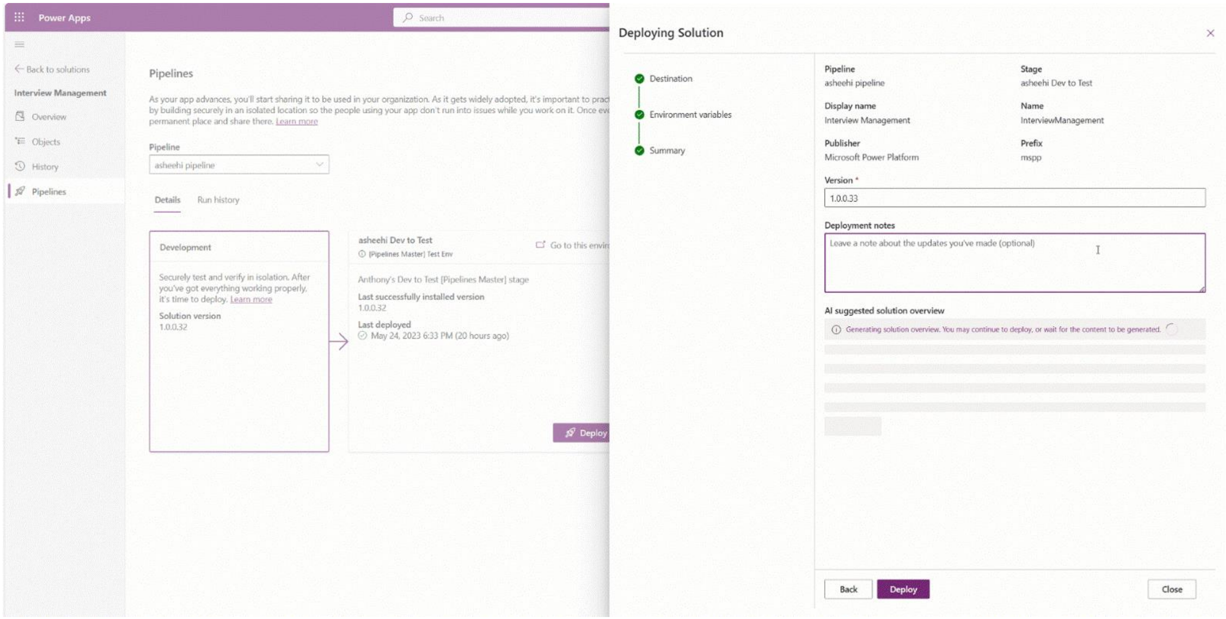
- Copilot labs is used by enterprises like 3M, Prada Group, Kraft Heinz.
- Friend in Microsoft confirmed Copilot can operate on C and Basic code.



```
const donut = Number(process.a

if ((donut % 3 === 0) && (donu
{
  console.log('FizzBuzz')
```

```
const donut = Number(process.ar

if ((donut % 3 === 0) && (donut
{
  console.log('FizzBuzz')
}
```

```
const donut = Number(process.argv[2])

if ((donut % 3 === 0) && (donut % 5 === 0))
{
  console.log('FizzBuzz')
}


else if (donut % 3 === 0)
{
  console.log('Fizz')
}

else if (donut % 5 === 0)
{
  console.log('Buzz')
}

else
{
  console.log(donut)
}
```



Pipelines

As your app advances, you'll start sharing it to be used in your organization. As it gets widely adopted, it's important to prac by building securely in an isolated location so the people using your app don't run into issues while you work on it. Once ev permanent place and share there. Learn more

Pipeline: asheehi pipeline

Development

Securely test and verify in isolation. After you've got everything working properly, it's time to deploy. Learn more

Solution version: 1.0.0.32

asheehi Dev to Test

[Pipelines Master] Test Env

Anthony's Dev to Test [Pipelines Master] stage

Last successfully installed version: 1.0.0.32

Last deployed: May 24, 2023 6:33 PM (20 hours ago)

Deploying Solution

Pipeline: asheehi pipeline

Stage: asheehi Dev to Test

Display name: Interview Management

Name: InterviewManagement

Publisher: Microsoft Power Platform

Prefix: mspp

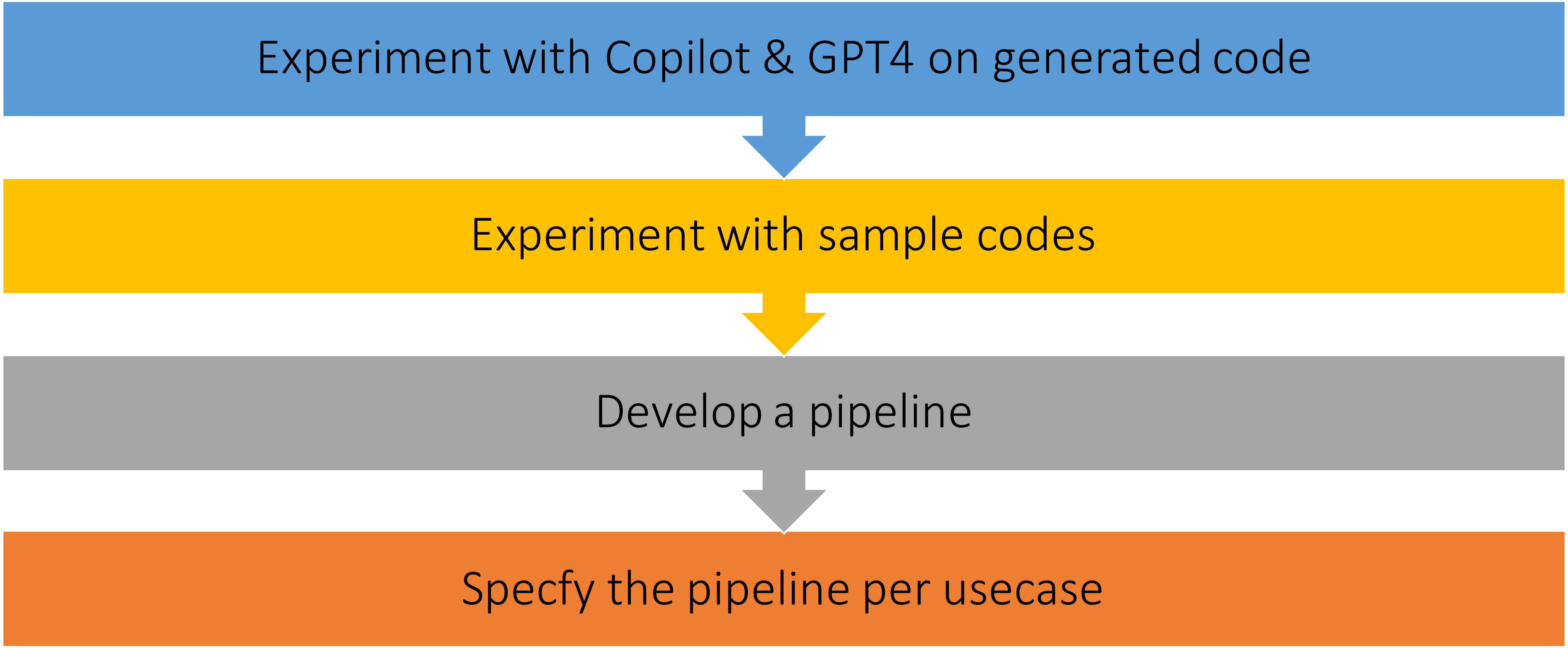
Version *: 1.0.0.33

Deployment notes: Leave a note about the updates you've made (optional)

AI suggested solution overview

Generating solution overview. You may continue to deploy, or wait for the content to be generated.

Project Timeline



Experiment with Copilot & GPT4 on generated code

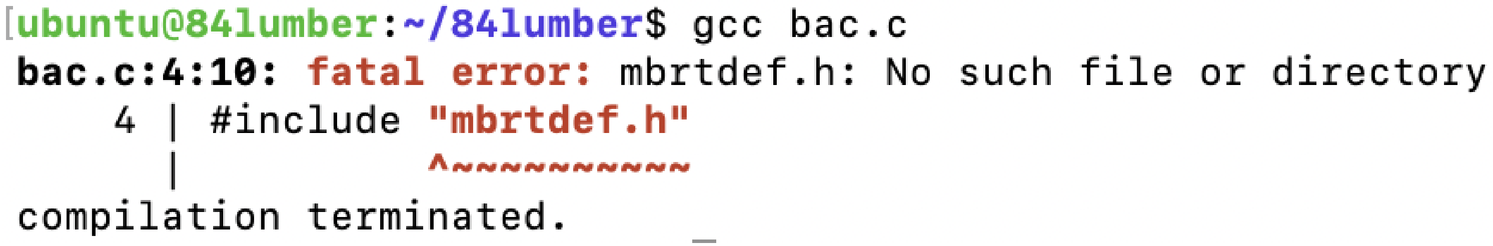
Experiment with sample codes

Develop a pipeline

Specfy the pipeline per usecase

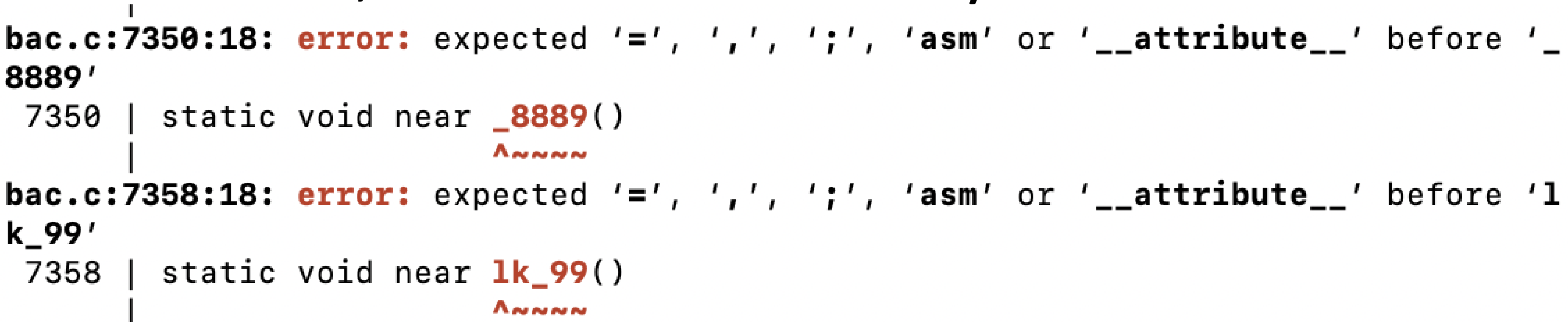
Current Stage and Challenges

- Waiting on Copilot and ChatGPT4.0
- The intention was to run the bac.c file to see what the file's functionality is so that we can make sure our translation achieves the same thing
- However, the mbrtdef.h file is needed for the bac.c file to execute



```
[ubuntu@84lumber:~/84lumber$ gcc bac.c
bac.c:4:10: fatal error: mbrtdef.h: No such file or directory
    4 | #include "mbrtdef.h"
      |          ^~~~~~~~~~~
compilation terminated.
```

- If commented out, errors occurs as they are not defined



```
bac.c:7350:18: error: expected '=', ',', ';', 'asm' or '__attribute__' before '_
8889'
 7350 | static void near _8889()
      |                  ^~~~~
bac.c:7358:18: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'l
k_99'
 7358 | static void near lk_99()
      |                  ^~~~~
```

- It is an obstacle for us to develop test cases and make sure the translation performs.
- Proposal
  - If the mbrtdef.h file cannot be provided, then maybe a detailed description OR a simulated run on your computer of what the file will do can be helpful
  - Maybe access to a virtual machine with only executable permission and no read and write permission on the mbrtdef.h file