# Lucas Turpin (40029907) Assignment 3

COMP 442 - Presented to Prof. Joey Paquet on March 31, 2020

## Section 1: Rules Implemented

| Rule | Implemented |
|---|---:|
| 1 | Yes |
| 2 | Yes |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |
| 6 | Yes |
| 7 | Yes |
| 8 | Yes |
| 9 | Yes |
| 10 | Yes |
| 11 | Yes |
| 12 | Yes |
| 13 | Yes |
| 14 | Yes |
| 15 | Yes |

## Section 2: Design

I. I used the visitor pattern with an abstract Visitor superclass and 3 different concrete visitors. As Python is a dynamically typed language, dynamic dispatch is achieved by creating a dictionary (a mapping) from ASTNode types to handlers in the concrete Visitor implementation. Calling *Visitor.visit(node)* leads to a look-up on this mapping which then calls the appropriate handler for the type of the node at runtime.
Symbol tables records are stored in a dictionary which maps identifier names to a list of Record objects, holding information such as record type, function parameters, return type or variable type (including array dimensions). Finding a function in a

symbol table is based on both the function name and parameter types to allow for function overloading.

II. I used a 3-pass approach where each visitor accomplishes the following tasks on the abstract syntax tree:

    A. The first pass implemented in *sem/vis/table_builder.py* generates symbol tables and symbol table records. Upon visiting a Class Declaration node or a Function Definition node, a new symbol table is created and attached to its appropriate parent table. The purpose of this first pass is simply to create symbol table records and insert them into appropriate tables.

       The only checks performed in this pass occur when binding a member function definition to the class symbol table. To do so, the class must exist and the function must have been declared as a member.

    B. The second pass implemented in *sem/vis/table_check.py* uses the tables generated in the first pass to issue warnings and check for errors based on the records contained in each scope. Dependency cycles are detected by a depth-first search of the dependency graph. Cycles found are reported as errors, then one of the edges is removed to allow further semantic checks to succeed. Records in the symbol tables are validated to ensure various identifiers are not multiply declared and warnings are issued for shadowed inherited members. Undeclared types are also reported as errors.

    C. The third pass implemented in *sem/vis/type_check.py* deals with type checking in expressions and statements. Upon reaching statement nodes in the AST such as an *if* statement or an assignment, the type of the children nodes is determined using attribute migration. Errors are raised when types do not match expectations, when array dimensions are not used appropriately or when identifiers for functions and variables are misused.

## Section 3: Use of Tools

No additional tools were used in this assignment.