**Compiler Project**

1. **Methods and Discussion** 
   1. ANTLR Setup
   2. Scanner/Lexer
   3. Parser
   4. Symbol Table

A compiler creates and maintains a data structure called a symbol table so that information can be stored about occurrences of various items in the language “such as variable names, function names, objects, classes, interfaces” and the like (https://www. tutorialspoint.com/compiler\_design/compiler\_design\_symbol\_table.htm). A compiler will use this data structure in both synthesis and analysis. Depending on the language used, this table may serve several different purposes. These include variable declaration verification, type checking (semantic correctness of expressions and assignments), scope resolution of a name, and centralized and structured names of all entities. This symbol table can take the form of a linear list, linked list, binary search tree or a hash table. For small languages a linear or linked list is adequate since traversing this list wouldn’t take too much time or computing power. For larger languages a hash table or binary search tree is necessary for reducing time (O(1) and O(log2 n) respectively) and cutting down on the usage of the CPU (https://www. geeksforgeeks.org/symbol-table-compiler/). The information stored in the table about each symbol is its name, type, and attribute and can be formatted as follows:

<symbol name, type, attribute>

The first decision to be made was to use either the default setting for ANTLR or choose ... The decision was made based on …

Difficulties were numerous but not impossible to overcome.