Written Report for CSE 4714/6714 Programming Languages Project Part 3

Name: Tichafa Rinomhota
NetID: tar351
Date: April 17, 2025

1. The development Process

   I was able to develop my Part 3 project by extending my Part 2 recursive descent parser to construct a parse tree. I took my Part 2 relevant files and worked around the example parse_tree_nodes.h and parse_tree_nodes.cpp files.My approache was the so called top-dow that Mr Willis used, starting with the ProramNode for the <program> production and moving onto to BlockNode, StatementNode and derived statement nodes. I did CompoundStmtNode, AssignmentStmtNode, IfStmtNode, WhileStmtNode. I was able to implement 12 parsing functions in my parse.cpp file. For each production I wrote the node class, parsing function and tested them incrementally using the -p flag to trace parsing steps. I tested with provided files like making sure my parser to could handle nested statements and expressions correctly.

2. Most & Least Difficult Nodes

   The most challenging nodes were CompoundStmtNode and ExprsessionNode. CompoundStmtNode took long for me and required for me to to do a lot of debugging. ExpressionNode had SimpleExpNode, TermNode, and FactorNode subclasses, with vectors for relational operators (TOK_EQUALTO, TOK_LESSTHAN), which also took me a long time. The easier one for me was StringNode as all I had to do was store a string literal and print it.

3. Parse Tree Printouts
   Printing the tree twice- once during parsing and once using operator<<—provided valuable insight. Because the printout during parsing helping trace production rules were initiated and the structure of the tree. The destructors printing out the deletion order assured me of debug memory issues.

4. Sources

   The sources I looked at https://en.cppreference.com/w/cpp/container/vector, https://en.cppreference.com/w/cpp, https://www.ibm.com/docs/en/developer-for-zos/15.0.x?topic=files-cc-parser,