**Parser Report**

**Tanwir Lalani, Sai Kiran, Jithin Joyson**

**Symbol Table:**

Since the identifiers in this language are all lowercase and uppercase letters, the symbol table consists of 52 indices. The updateSymbolVal() function updates each change in the identifier onto the symbol table.

**Declarations:**

Standard and math libraries are included in the parser to write c code and conduct arithmetic operations. The union in our parser consists of float and char types. Since our calculator will only work with decimal values and identifiers are characters, it was decided that this would be the best course of action. The tokens and type are created from the union variables. Associativity and precedence is also considered in our parser. Basic arithmetic operations (+,-,/,\*,=) and complex operations (abs,sqrt,^) are given precedence from left to right and basic to complex. Structs of various nodes were declared to identify each entity of the expression.

**Grammar:**

Most of the grammar in the parse have been reduced from the set grammar/production rules due to errors regarding string formation. This idea will be visited later and fixed before turning in the final project. Due to strings and the comments being an output of string formations, these changes will be added as more insight is gained from the topic. All the calculation portion of the doubles are created and the parser correctly outputs the results. Some changes were made to the original grammar to create a working parser. In order to construct our abstract syntax tree, each non-terminal was assigned a specific node type to output AST of the input stream accordingly.

**C Code:**

Functions such as: computeSymbolIndex(), updateSymbolVal() and main() are created to create, initialize the symbol table and update the values in the table respectively.