**REPORT**
**Assignment 2**

**Scanner:**

* Token is being returned which is used in Parser.
* For integer, character and string constant, the value is put into yylval.
  Integer and ASCII value of character value is put into yylval.value.
  String value is put into yylval.strval.
* Error message is being passed via yyerror() function to the Parser.

**nodetype.h:**

* enum nodetypes are kept in nodetype.h. It is used in both parser & tree.

**Parser:**

* Production rules are defined with node types and token.
* Arithmetic operators(+, -, *, /, %) are defined as left associative.
* To remove ambiguity for IF ELSE condition, a higher precedence is given to IF-ELSE than simple IF.
First IFX and KWD_ELSE made non-associative. Then %prec IFX is added in 'if' section of the 'condStmt'.

* AST is being created whose internal nodes are non-terminal & leaf nodes are terminal.
* For empty rule, $$ = NULL action. So for AST, when adding child to a parent, if child is NULL, it is discarded.
* For terminal nodes, value of the nodes are also saved in tree nodes.
  Value for type specifier nodes are int, char, void & string.
  Value for keyword nodes are if, else, while, return.
  Value for add & multiply operations are +, -, *, /, %.
  Value for relational operations are >, <, >=, <=, ==, !=.
  For identifier, yylval.strval.
  For Integer and ASCII value, yylval.value & for string, yylval.strval.

**AST tree:**

* AST tree is being traversed using in-order tree traversal algorithm.
* Non-terminal node names are being printed according to the example AST in the assignment.
* Terminal node names are being printed along with their value.

**Symbol Table:**

* For each ID, with their value and scope, a hash key is generated. IDs are saved in ‘strTable’ hash map.
* For IDs with same hash value, they are linked after one another in STACK fashin. Then the head node is saved in the ‘strTable’.

**Error:**

* Scanner error message are being passed via yyerror() function.
* Warning for ‘Use of undeclared symbol’ is passed from strtab.c

**Compile & Test:**

* Go to the assgn2 folder. Run 'make clean' & then 'make'.
* To run example test, give the name of the test file in place of 'input_file'.

    ./obj/scanner < input_file.mC
    Example: ./obj/scanner < test.mC

* Along with the given test cases, we have used two more test cases. test1.mC & test2.mC