# CS3005D Compiler Design

Assignment 1

February, 2022

## Instructions

- This assignment is to implement a small compiler for the given grammar. There are two grammars, viz., *Grammar 1* and *Grammar 2*. You may implement the compiler for any one of them. If you choose *Grammar 1*, you can score a maximum of **12** out of 20 marks. On the other hand, if you choose *Grammar 2*, you can score a maximum of **20** out of 20 marks.
- Your submission has to be prepared in `C` programming language using Lex/Flex and Yacc/Bison. Alternatively, you may write the whole source code in `C` without the use of any other tools. However other language implementations are not supported by the submission verification and testing tools available to us. (You can use the tutorials available at the site silcnitc.github.io for learning Lex and Yacc programming in `C`).
- Your compiler must generate XSM target executable file that can be run on the XSM simulator supplied at silcnitc.github.io. (Tutorial for generating XSM target code is also available there in silcnitc.github.io).

## Grammar 1 [12 Marks]

**Keywords** : *FUN, if, else, do, while, read, write*

```
Program     ::=  FUN() { Stmts }  /* There is only one function that returns an integer */

Stmts       ::=  Stmts Stmt
                 | Stmt

Stmt        ::=  AsgStmt
                 | IfStmt
                 | IfElseStmt
                 | WhileStmt
                 | ReadStmt
                 | WriteStmt

AsgStmt     ::=  VAR = E;

IfStmt      ::=  If (E) { Stmts }

IfElseStmt  ::=  If (E) { Stmts } else { Stmts }

WhileStmt   ::=  do { Stmts } while (E);

ReadStmt    ::=  read(VAR);                    /* Reads an integer into a variable */

WriteStmt   ::=  write(E);                     /* Prints an integer expression */

E           ::=  E + E
                 | E - E
                 | E * E
                 | E <= E
                 | E >= E
                 | E == E
                 | E != E
                 | (E)
                 | NUM
                 | VAR              /* All expressions evaluate to integer values, as in C */

NUM         :==  digit(digit)*

VAR         :==  letter(letter|digit)*        /* Variables can store only integers */
```

### Example Program 1.1:

```
FUN() {     /*  Max of three numbers **/
       read (abc);
       read (pqr);
       read (stu);
       If (abc > pqr)  {
              If (abc >  stu) {  write(abc); }  else { write(stu); }
       } else {
              If (pqr >  stu) {  write(pqr); }  else { write(stu); }
       }
}
```

### Example Program 1.2:

```
FUN() {     /*  Sum of numbers till zero is entered **/
       sum = 0;
       do {
             read(x);
             sum = sum + x;
       }  while (x != 0 );
       write (sum);
}
```

## Grammar 2 [20 Marks]

**Keywords** : *FUN, if, else, read, write, return, argc*

```
Program     ::=  read(argc); argc=FUN(argc) { Stmts } write(argc);
                 /* FUN: int → int function; in the initial
                  call the value read must be passed to FUN */

Stmts       ::=  Stmts Stmt
                 | Stmt

Stmt        ::=  AsgStmt
                 | IfStmt
                 | IfElseStmt
                 | RetStmt

AsgStmt     ::=  VAR = E;        /** Variables have local scope inside FUN **/

IfStmt      ::=  If (E) { Stmts }

IfElseStmt  ::=  If (E) { Stmts } else { Stmts }

RetStmt     ::=  return E;

E           ::=  E + E
                 | E - E
                 | E * E
                 | E <= E
                 | E >= E
                 | E == E
                 | E != E
                 | (E)
                 | NUM
                 | VAR
                 | FUN(E)

NUM         :==  digit(digit)*

VAR         :==  letter(letter|digit)*/* Variables can store only integers */
```

### Example Program 2.1:

/** Calculates the factorial of *argc* **/

```
read (argc);
argc = FUN(argc) {
       If ( argc == 1 )  {
             x = argc;
       }  else {
             x = FUN(argc-1) * argc;
       }
      return x;
}
write(argc);
```

### Example Program 2.2:

/** Calculates the $argc^{th}$ Fibonacci number **/

```
read(argc);
argc = FUN(argc) {
        If (argc == 1) {  return 1;  }
        If (argc == 2) {  return 1;  }

        return ( FUN(argc-1) +  FUN(argc-2) );
}
write(argc);
```