# Assignment-5

Due : $9^{th}$ March, 2021

Augment your syntax analyzer using Bison which will accept an input code written in a toy programming language. You have already designed a lexical analyzer and a parser in your previous assignments. Now add the following rules to your grammar.

**Production Rules :**

$prog : funcDef;$

$funcDef : type\ ID\ '('\ argList\ ')'\ '\{'\ declList\ stmtList\ '\}'$

$argList : arg\ ','\ arg\ |\ \epsilon;$

$arg : type\ ID;$

$type : INT\ |\ FLOAT;$

$declList : declList\ decl\ |\ decl;$

$decl : type\ varList\ SEMICOLON;$

$varList : ID\ COMMA\ varList\ |\ ID;$

$stmtList : stmtList\ stmt\ |\ stmt;$

$stmt : assignStmt\ |\ ifStmt\ |\ whileStmt;$

$assignStmt : ID\ '='\ EXP\ SEMICOLON;$

$EXP : EXP\ '+'\ TERM\ |\ EXP\ '-'\ TERM\ |\ TERM;$

$TERM : TERM\ '*'\ FACTOR\ |\ TERM\ '/'\ FACTOR\ |\ FACTOR;$

$FACTOR : ID;$

$ifStmt : IF'('bExp')''\{'stmtList\ '\}';$

$bExp : EXP\ RELOP\ EXP;$

$whileStmt : WHILE'('bExp')''\{'stmtList'\}';$

To add the above rules, you have to add *RELOP* as a token for all the symbols $>, <, \geq, \leq, ==, !=, ==$. Also add grammar rules to accept boolean expression *bExp* containing logical expression logical and, or and not. Add grammar rules for accepting else and else if statements.

A sample accepted input is given below.

```
int main()
{
    int a, b, c;
    a = b + c;
    if(a>b){ a = b + c;}else{ a = b - c; }
    while(a<b){ a = a+c;}
}
```