Programming Assignment 2

1. (10 points) Use yacc or bison to implement a syntax analyzer for the miniC language.
   - See an attached file for the lexical rules in details.
   - You are requested to separate the C code and the yacc/bison specification into distinct files.

Guideline:

1. You have to demonstrate your program in person and have the report in paper with you.
2. You will get 30% bonus if you succeed in demonstrating your program in class, while the report in paper still need to be handed-in in the due week. And, 30% penalty will be given for lateness. More precisely, if you get X in demonstration, and Y for the report:
   - Your score = X * 70% + Y * 30%
   - In-class demonstration and on-time report = X * 70% * 1.3 + Y * 30%
   - In-class demonstration but late report = X * 70% * 1.3 + Y * 30% * 0.7
   - On-time demonstration but late report = X * 70% + Y * 30% * 0.7
   - Late demonstration and on-time report = X * 70% * 0.7 + Y * 30%
   - Both late = (X * 70% + Y * 30%) * 0.7
3. Your report have to include the following elements:
   I. A cover page.
   II. The problem description.
   III. Highlight of the way you write the program.
   IV. The program listing.
   V. Test run results.
   VI. Discussion.

SYNTAX RULES

```
Smallc_program ::= (Type_specifier id '(' (Param_decl_list)?
')' Compound_stmt)+

Type_specifier ::= int

Param_decl_list  ::= Param_decl (',' Param_decl )*

Param_decl ::= Type_specifier id

Compound_stmt  ::= '{' (Var_decl* Stmt*)? '}'

Var_decl ::= Type_specifier Var_decl_list ';'

Var_decl_list ::=  Variable_id ( ',' Variable_id)*

Variable_id  ::= id ( '=' Expr )?

Stmt ::= Compound_stmt | Cond_stmt | While_stmt | Assign_stmt
| break ';' | continue ';' | return expr ';' | printf '(' string
(',' Expr)? ')' ';'

Assign_stmt ::= id '=' Expr ';'

Cond_stmt ::=  if '('  Expr ')' Stmt (else Stmt)?

While_stmt ::= while '(' Expr ')' Stmt

Expr ::= id '=' Expr | Condition

Condition ::= Disjunction | Disjunction '?' Expr ':' Condition

Disjunction ::=  Conjunction | Disjunction '||' Conjunction

Conjunction ::= Comparison | Conjunction '&&' Comparison

Comparison ::= Relation | Relation '==' Relation

Relation ::= Sum | Sum ('<' | '>') Sum

Sum ::= Sum '+' Term | Sum '-' Term | Term

Term ::= Term '*' Factor | Term '/' Factor | Term '%' Factor
| Factor

Factor ::= '!' Factor | '-' Factor | Primary

Primary ::=  num | id | id '(' Expr_list ')' | '(' Expr ')'
```

```
Expr_list  ::= Expr (',' Expr )*
```

TEST PROGRAM

```
int ComputeFac(int num) {
    int num_aux;
        if (num < 1)
            num_aux = 1;
        else
            num_aux = num * ComputeFac(num-1);
        return num_aux;
}

int main() {
        printf("%d\n", ComputeFac(10));
}
```