Programming Assignment 1

1. Use lex (or flex) and yacc (or bison) to implement a front end (including a lexical analyzer and a syntax recognizer) of the compiler for the T language.
   - See an attached file for the lexical rules in details.
   - You are requested to separate the C code, the Lex specification, the Yacc 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 10% bonus if you succeed in demonstrating your program on April 13th or 15th. No bonus or penalty will be given if you demonstrate your program AND upload the report on the official due (June 1st or 3rd), but after that 10% penalty will be given for lateness. More precisely, if you get X in demonstration, and Y for the report:
   - (4/13, 15) In-class demonstration = X * 70% * 110% + Y * 30%
   - (6/1, 3) Your score = X * 70% + Y * 30%
   - Late = (X * 70% + Y * 30%) * 90%
3. Your report has 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.

The *T* Programming Language

The T Lexicons

Keywords (All keywords are reserved. Each keyword can be a terminal.):

```
WRITE READ IF ELSE RETURN BEGIN END MAIN INT REAL
```

Single-character separators (Each operator can be a terminal.):

```
; , ( )
```

Single-character operators (Each operator can be a terminal.):

```
+ - * / > <
```

Multi-character operators (Each operator can be a terminal.):

```
:= == != >= <=
```

Identifiers:

An *identifier* consists of a letter followed by any number of latters or digits.

Integer numbers:

An *integer number* is a sequence of digits, where a *digit* has the following definition:

```
Digit -> ‘0’ | ‘1’ |‘2’ |‘3’ |‘4’ |‘5’ |‘6’ |‘7’ |‘8’ |‘9’
```

Real numbers:

A *real number* is a sequence of digits followed by a dot, and followed by digits.

Comments:

A *comment* is a string between `/*` and `*/`. Comments can be longer than one line.

QStrings:

A *QString* is any sequence of characters except double quote itself, enclosed in double quotes.

The T Grammar

The *T* grammar is given by EBNF rules as follows.

High-level program structures:

```
Program -> MethodDecl MethodDecl*
Type -> INT | REAL
MethodDecl -> Type [MAIN] Id '(' FormalParams ')' Block
FormalParams -> [FormalParam (',' FormalParam)*]
FormalParam -> Type Id
```

Statements:

```
Block -> BEGIN Statement+ End

Statement -> Block
           | LocalVarDecl
           | AssignStmt
           | ReturnStmt
           | IfStmt
           | WriteStmt
           | ReadStmt

LocalVarDecl -> Type Id ';' | Type AssignStmt

AssignStmt -> Id := Expression ';'

ReturnStmt -> RETURN Expression ';'

IfStmt -> IF '(' BoolExpression ')' Statement
        | IF '(' BoolExpression ')' Statement ELSE Statement

WriteStmt -> WRITE '(' Expression ',' QString ')' ';'

ReadStmt -> READ '(' Id ',' QString ')' ';'
```

Expressions:

```
Expression -> MultiplicativeExpr ( ('+' | '-') MultiplicativeExpr )*

MultiplicativeExpr -> PrimaryExpr ( ('*' | '/') PrimaryExpr )*

PrimaryExpr -> Num // Integer or Real numbers
             | Id
             | '(' Expression ')'
             | Id '(' ActualParams ')'

BoolExpr -> Expression '==' Expression
          | Expression '!=' Expression
          | Expression '>' Expression
          | Expression '>=' Expression
          | Expression '<' Expression
          | Expression '<=' Expression

ActualParams -> [Expression (',' Expression)*]
```

A Sample Program

```
/* This is a comment line in the sample program. */
INT f2 ( INT x, INT y )
BEGIN
   INT z;
   z := x*x - y*y;
   RETURN z;
END

INT MAIN f1 ()
BEGIN
   INT x;
   READ(x, "Please input an integer number x: ");
   INT y;
   READ(y, "Please input another integer number y: ");
   INT z;
   z := f2(x, y) + f2(y, x);
   WRITE(z, "f2(x, y) + f2(y, x) = ");
END
```