CMPSC471 Programming Project: Mini Interactive Calculator
(due = October 29th, 2018; 11:55PM)

Description:
Implement an interactive calculator using lex and yacc.

Details:
- The calculator *accepts* one expression at a time, *evaluates* the expression, and *prints* the result value in the next line. The input line starts with "->" and the result line starts with "=>".
- The input expression is represented in infix notation. An expression (`expression`) consists of variables (`variable`), expressions, numbers and ops (supported *ops* listed as below).
- A user can define variables. A declared variable is set to 0 if not initialized. A variable begins with a letter followed by any number of letters (lower-case).
- Numbers (`number`) are represented in decimals using `digit`.
- If an expression has syntax error, it simply prints "error", and waits for the next input.
- Precedence and associativity are defined as below.

Submission Guideline:
You need to submit a lex and a yacc file, with your .c files. You need to provide a Makefile to build the calculator executable. A manual of usage is needed for describing the usage.

Grammar Description:
```
expression :  variable            | number              | '(' expression ')'
            | '-' expression      | expression '+' expression
            | expression '-' expression | expression MUL_OP expression
            | expression '^' expression | variable ASSIGN_OP expression
            | expression REL_OP expression
variable     : letter[letter]*
letter       : [a-z]
number       : integer | '.' integer  | integer '.' integer
integer      : digit   | integer digit
digit        : [0-9]
op           : MUL_OP | ASSIGN_OP | REL_OP
MUL_OP       : * /
ASSIGN_OP    : =
REL_OP       : == <= >= != < >
```

Precedence and Associativity (lowest to highest):
```
REL_OP, left associative
ASSIGN_OP, right associative
+ and - operators, left associative
MUL_OP, left associative
^ operator, right associative
unary - operator, nonassociative
```

Examples:
```
-> a = 5
=> 5
-> b = a + 7 * 3.1
=> 26.7
-> (b - a)* 0.2
=> 4.34
```

References:

lex and yacc. http://dinosaur.compilertools.net/
bc - arbitrary-precision arithmetic language

1. bc Command Manual, https://www.gnu.org/software/bc/manual/html_mono/bc.html
2. bc, http://pubs.opengroup.org/onlinepubs/000095399/utilities/bc.html