CS117 (Fall 2015) Programming Assignment 3 30 pts. Due: Oct. 1 (Th)

We design a mini-language named FresnoF15 that supports variable declarations and three statements; i.e., assignment statement, input statement, and output statement.
So far, we have designed a simple language named Simplified-Infix-Expression that is a portion of the FresnoF15 and implemented its interpreter using recursive-descent-parsing technique. This assignment extends the interpreter to accept language FresnoF15. The syntax of FresnoF15 in BNF is:

```
          <Prog> ::=  program  <Declarations> begin <Statements> end
  <Declarations> ::=  <Declaration> | <Declaration> <Declarations>
   <Declaration> ::=  var <Id-list> ;  |  ε
       <Id-list> ::=  <Id> | <Id> , <Id-list>
    <Statements> ::=  <Statement> <Statements>  |  ε
     <Statement> ::=  <Assign-St>  |  <Input-St> | <Output-St>
     <Assign-St> ::=  <Id> = <Exp> ;
      <Input-St> ::=  input <Id> ;                                   we have completed
     <Output-St> ::=  output <Id> ;  |  output <Exp> ;                this part
            <Id> ::=  a|b|c| … |z|A|B|C| … |Z

           <Exp> ::=  <Term> <Exp2>
          <Exp2> ::=  + <Term> <Exp2>  |  - <Term> <Exp2>  |  ε
          <Term> ::=  <Factor> <Term2>
         <Term2> ::=  * <Factor> <Term2>  |  / <Factor> <Term2>  |  ε
        <Factor> ::=  <Num> | <Num> ^ <Factor>
           <Num> ::=  0|1|2|3|… |9  | (<Exp>) |
```

The above grammar is already in the right-recursive form.
Items in the left-hand side are all non-terminals, and terminals include { program, begin, end, var, input, output, =, ; , , , +, -, *, /, ^, (,), 0..9, a..z, A..Z }
A possible sample program in FresnoF15 is:

```
program
  var a, b;
  var c, d;
  begin
     input a;
     b = 3*(5+2);
     output b;
     c = (3+4)*5;
     output c;
     output a;
     output (2+3)*7+2^(1+2);
  end
```

The expected output from the interpreter is:

```
    21
    35
    the input value typed for a
    43
```

- Build an interpreter for FresnoF15, and submit the hardcopies of your source code and output.
  Also, prepare for the demonstration.
  Input : a mini-language FresnoF15 programming (stored in a data file)
  Output : execution result
- Your interpreter should check at least two errors for each of the following three error classes:
  Lexical error, Syntax error, Semantic error