CS117 (Fall 2017) Programming Assignment 3 30 pts. Due: Sept. 21 (Th)

We design a mini-language named FresnoF17 that supports variable declaration with type and two primitive statements; i.e., assignment statement and print statement.
So far, we have designed a simple language named Simplified-Infix-Expression, which is a portion of the FresnoF17, and implemented its interpreter using recursive-descent-parsing technique. This assignment extends the interpreter to accept language FresnoF17. The syntax of FresnoF17 in BNF is:

```
        <Prog> ::=  program  <Declarations> begin <Statements> end
<Declarations> ::=  <Declaration> | <Declaration> <Declarations> |  ε
 <Declaration> ::=  <Type> <Id-list> ;
       <Type> ::=  int | double
     <Id-list> ::=  <Id> | <Id> , <Id-list>
 <Statements> ::=  <Statement> <Statements>  |  ε
  <Statement> ::=  <Assign-St>  |  <Print-St>
  <Assign-St> ::=  <Id> = <Exp> ;  | <Id> = <Id> ;
    <Print-St> ::=  print <Id> ;  |  print <Exp> ;
          <Id> ::=  a|b|c| … |z|A|B|C| … |Z
```

we have completed this part

```
       <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, int, double, print, =, ; , , , +, -, *, /, ^, (,), 0..9, a..z, A..Z }

A sample program in FresnoF17 is:

```
program
  int a, b, c;
  double d;
  begin
     a = 3*(5+2);
     b = (3 + 4) * 5;
     c = a;
     print a;
     print b;
     print c;
     print (2+3)*7 + 2^3;
  end
```

Expected output from the interpreter is:

```
21
35
21
43
```

- Build an interpreter for FresnoF17, and submit the hardcopies of your source code and input/output.
  Input : a mini-language FresnoF17 programming (above code stored in a data file);
  Output : execution result (screen snapshot);
- Your interpreter should check at least two errors for each of the following three error classes:
  Lexical error, Syntax error, Semantic error

Please make your own programs having errors and show your outputs displaying error messages.