

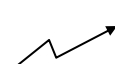
We design a mini-language named ‘FresnoF24’ that supports variable declaration with type and two basic statements; i.e., assignment statement and print statement.

So far, we have designed a simple language named Simplified-Infix-Expression, which is a portion of the FresnoF24, and implemented its interpreter using recursive-descent-parsing technique. This assignment extends the interpreter to accept language FresnoF24. The syntax of FresnoF24 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
    <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>)

```

we have completed this part 

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 }

An additional feature you should implement is multiple digit numbers, which the above grammar does not cover. You should upgrade the function for <Num> to handle this.

A sample program in FresnoF24 is:

```

program
  int a, b, c;
  double d;
  begin
    a = 2*(55+200);
    b = (31 + 4) * 50;
    c = a;
    print a;
    print b;
    print c;
    print (2+300/ 2)*4 + 2^3;
  end

```

Expected output from the interpreter is:

```

510
1750
510
616

```

- Build an interpreter for FresnoF24, and submit your source code and input/output.
Input : a mini-language FresnoF24 programming (above code stored in a 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 sample programs having errors and show your outputs displaying appropriate error messages.