CS280 Programming Assignment 3+4
Spring 2017

The final programming assignment will create a recursive descent parser for a little language for polynomials, and will implement an interpreter for the language,

There will be three weeks of deliverables for the assignment: Monday April 17, April 24, and May 1.

You will be given code segments in class that you can incorporate into your programs.

The grammar for our language is as follows:

```
Prog := Stmt | Stmt Prog
Stmt := Set ID Expr SC | PRINT Expr SC
Expr := Term { (+|-) Expr }
Term := Primary { * Term }
Primary :=  ICONST | FCONST | STRING | ( Expr ) | Poly
Poly := LBR Coeffs RBR { EvalAt } | ID { EvalAt }
Coeffs := Coeff { , Coeff }
Coeff := ICONST | FCONST
EvalAt := LSQ Expr RSQ
```

Note that there are 9 rules so there may be 9 functions, one per rule; you may decide that you want to combine some rules together for ease of implementation.

Some examples:

```
set f { 1, 2, 4 };  # f is x^2 + 2x + 4
print f[0] ; # should print 4

set f { 2,0,0,1 };  # 2x^3 + 1
set g { 1,0, 0 }; # x^2
set x 2;
set gofx g[x];   ## should be 4  (2^2)
set fofgofx f[ gofx ];  ### should be  129


print {1,1,1,1}[2];   ## should print 2^3 + 2^2 + 2 + 1  or 15

print {1,1,1}*2;  ## should print {2,2,2}

print 3 + 4*5;  ## should print 23
```

Every error detected in the parse phase must be printed on its own line with the first words as PARSE ERROR:

A runtime error should be printed on its own line with the first words RUNTIME ERROR:

There are four operations: addition, subtraction, multiplication, and polynomial evaluation.

The language has the following semantic rules:

1. An identifier must be set before it can be used
2. Addition, subtraction and multiplication associate left to right
3. All binary operators are commutative; if we specify the behavior of integer op float, then that behavior applies to float op integer
4. Addition is defined between the following like types: two integers, two floating point numbers, two strings, or two polynomials.
5. String addition is concatenation
6. A mixed mode expression of integer + float causes the integer to be converted to a float
7. A mixed mode expression of either integer or float + poly causes the integer or float to be converted to a poly
8. Subtraction is defined between the following like types: two integers, two floating point numbers, or two polynomials. Note: string subtraction is not defined.
9. A mixed mode expression of integer - float causes the integer to be converted to a float
10. A mixed mode expression of either integer or float - poly causes the integer or float to be converted to a poly
11. Multiplication is defined between the following like types: two integers, two floating point numbers, or two polynomials. Note: multiplying a string by another string is not defined.
12. A mixed mode expression of integer * float causes the integer to be converted to a float
13. A mixed mode expression of either integer or float * poly causes the integer or float to be converted to a poly
14. A mixed mode expression of string * int results in a string that is repeated int times
15. Polynomials can only be evaluated at integer or float values
16. ALL OTHER COMBINATIONS OF OPERATORS AND TYPES ARE ERRORS
17. EvalAt means calculating the value of a polynomial for a particular value
18. Set evaluates the expression and binds the value to the variable named in the set statement
19. Print evaluates the expression and prints its value
20. When asked to print a polynomial, the Print statement prints a curly-brace enclosed, comma-separated list of coefficients
21. Strings are printed without their quotation marks

Week 1 Test Cases (due April 17)

- Detect parse errors
- There are 10 test cases:
    - Compile
    - 4 cases of files with errors; you must detect those errors
    - 5 test cases with no errors; your program should generate no output in these cases
- You can see the test files by running “ls $LIB/public/p3”
- The test case names are err1, err2, err3, err4 and good1, good2, good3, good4 and good5.

Week 2 Test Cases (due April 24)

- The first 5 test cases from part 1 (compile, and error cases) are included
- Detect case of using a variable before it is declared
- Set identifiers to iconst, fconst, string
- Print iconsts, fconsts, strings and identifiers
- Addition, subtraction with Set and Print

Week 3 Test Cases (due May 1)

- Evaluations of mixed-mode expressions
- Evaluations of polynomials