CS280 Programming Assignment 4
Fall 2017
Part 1 due Nov 27 before midnight, no later than Dec 2
Part 2 due Dec 4 before midnight, no later than Dec 9
Part 3 due Dec 11 before midnight, no later than Dec 14 (NOTE SHORTENED INTERVAL)

Now that we have a parsed input in our language, it is time to enhance the program to generate an evaluation of the parse.

To do this assignment, we will be extending assignment 3 in the following ways

- Eliminate the -t trace option and the statistics; they are not required in assignment 4

- Add static type checking so as to enforce semantic rules 5-17
  - The format of a type error message should be filename:line:Type error
  - If there are any type errors, you should NOT proceed with evaluating

- Add an evaluation function so that every class can be evaluated
  - The evaluation function for the different types of statements will execute the statements
  - The evaluation function for the various mathematical operators perform the calculations to determine the value of the expression
  - The evaluation function for constants will simply evaluate to the constant
  - The evaluation function for an identifier will return the value bound to the identifier

- Consider runtime errors such as uninitialized variables and divide by 0
  - There does not need to be any checks for uninitialized variables
  - Divide by 0 should cause your program to print filename:line:DIVIDE BY ZERO and stop

Some notes for you on doing the assignment:

1. Your symbol table is, or should be, a map of an identifier to some instance of a value class, which contains all the bindings for a variable (type and value). Let's refer to that class as Value.
2. You can use Value to hold values of constants, variables and expressions. In other words, the return type of your evaluation function is Value.
3. If you overload operator<< for Value, you can do the print command easily
4. You can overload the various operator functions for Value
5. Note that since you do static type checking before you evaluate expressions, you are guaranteed that you will only be running correct combinations of operands!

REMINDER: The following are the semantic rules associated with this language:

1. There are only two types: integer and string.
2. Variables must be declared before they are used. They are declared with Decl.
3. There is one single scope for variables.
4. Variable names must be unique; once a name has been declared, it may not be declared again.
5. An integer constant is of type integer.
6. A string constant is of type string.
7. The type of an identifier is the type assigned when it is declared.
8. The type of an Expr or a Term is the type of the result of the operation.
9. Adding two integers results in an integer representing the sum
10. Subtracting two integers results in an integer representing the difference
11. Multiplying two integers results in an integer representing the product
12. Dividing two integers results in an integer representing the quotient
13. Adding two strings results in a string representing the concatenation
14. Multiplying an integer by a string results in a string which is the string operand repeated integer times (i.e. 3 * “hi” is “hihihi”)
15. Dividing two strings results in an a string where the first instance of the denominator that is found in the numerator is removed.
16. ALL OTHER COMBINATIONS OF OPERATIONS ARE UNDEFINED and are an error
17. A variable is assigned a value using a Set. The type of the variable being set must match the type of the expression; if it does not this is an error.
18. An expression can be printed using either “print” or “println”. In both cases, the value of the Expr is printed to the standard output. In the case of “println”, a newline is printed to standard out at the end of the execution of the program.