CS280
Programming Assignment 4
CS280 Fall 2018

Now that we have a parser for our grammar, we will add an interpreter for the language.

In order to build the interpreter, we must build an evaluation function for each operation in the language. The evaluation functions are responsible for performing all type and value checks that were outlined in assignment 3. Any failures of these checks result in a RUNTIME ERROR.

All requirements for error detection and error message generation from Assignment 3 are included in assignment 4.

As a reminder from assignment 3, the following list of items describe the language. Note the small change in item 4, in red.

1. The language contains three types: integer, string, and boolean.
2. All operators are left associative except for ASSIGN and unary MINUS, which are both right associative.
3. An IfStmt evaluates the Expr. The expr must evaluate to a boolean. If the boolean is true, then the Stmt is executed.
4. A PrintStmt evaluates the Expr and prints its value. ***A boolean valued Expr should print as the string True or False.*** A newline ***should be appended after the print.***
5. In an Expr, the ASSIGN operator copies the value from the operand on the right side of the operator to the operand on the left side of the operator. The left side of the operator must be an IDENT. If the IDENT does not exist, it is created. If the ident already exists, its value is replaced. The value of the Expr is the value of the identifier, and the type of the Expr is the type associated with the value.
6. The type of an IDENT is the type of the value assigned to it.
7. The LogicExpr performs short-circuited evaluation of logical and and or.
8. The CompareExpr performs a comparison between its operands.
9. The PLUS and MINUS operators in AddExpr represent addition and subtraction.
10. The STAR and SLASH operators in MulExpr represents multiplication and division.
11. The MINUS operator in Factor is a unary minus.
12. The type of TRUE is boolean true. The type of FALSE is boolean false.
13. It is an error if a variable is used before a value is assigned to it.
14. Addition is defined between two integers (the result being the sum) or two strings (the result being the concatenation).
15. Subtraction is defined between two integers (the result being the difference).
16. Multiplication is defined between two integers (the result being the product) or for an integer and a string (the result being the string repeated integer times).
17. Division is defined between two integers

18. Logical and and logical or are defined between two booleans. The result is the boolean value of the logical operation.
19. Comparisons for equality and inequality is defined between pairs of integers, pairs of strings, and pairs of booleans. The result is the boolean result of the test.
20. Comparisons for other than equality and inequality are defined between pairs of integers and pairs of strings.
21. Unary minus is defined for an integer (which is defined as -1 * the integer) and for boolean (which is defined as a logical not).
22. Performing an operation with incorrect types or type combinations is an error.
23. Multiplying a string by a negative integer is an error.
24. Assigning to anything other than an identifier is an error.
25. An IF statement whose Expr is not boolean typed is an error.

For Programming Assignment 4, you mult implement a mechanism for evaluating each of the parse tree nodes that you created for Assignment 3.

The evaluation function for each parse tree node must evaluate any arguments, check for any errors, and generate either a Value representing the evaluation, or an error.

You must create a main for your interpreter, with the same requirements as the main from assignment 3.

The result of an unsuccessful parse is a set of error messages printed by the parse functions. If the parse fails, the program should stop after the parse function returns. The requirements for parse error messages are the same for assignment 3.

On a successful parse, the tree should be evaluated using some evaluation function that you write.

The assignment does not specify the exact runtime error messages that should be printed out by the parse; however the format of the message should be the line number, followed by a colon and a space, followed by RUNTIME ERROR followed by some descriptive text.

PART 1 will test simple expressions and statements.

PART 2 will have more complex inputs