CS280
Programming Assignment 4
CS280 Spring 2019

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, here is the grammar and the semantic rules of the language:

```
Prog := Slist
Slist := Ssep { Slist } | Stmt Ssep { Slist }
Ssep := NL | SC
Stmt := IfStmt | PrintStmt | SetStmt | LoopStmt
IfStmt := IF Expr BEGIN Slist END
PrintStmt := PRINT Expr
SetStmt := SET IDENT Expr
LoopStmt := LOOP Expr BEGIN Slist END
Expr := Prod { (PLUS | MINUS) Prod }
Prod := Primary { (STAR | SLASH) Primary }
Primary := IDENT | ICONST | SCONST | LPAREN Expr RPAREN
```

Below is the list from assignment 3. Note that several of the items in the list are shown in boldface. These are items that must be checked at runtime.

1. The language contains two types: integer and string.
2. All operators are left associative.
3. An IfStmt evaluates the Expr. **The expr must evaluate to an integer**. If the integer is nonzero, then the Stmt is executed.
4. A PrintStmt evaluates the Expr and prints its value.
5. A SetExpr evaluates Expr and saves its value in memory associated with the IDENT. If the IDENT does not exist, it is created. If the IDENT already exists, its value is replaced.
6. The type of an IDENT is the type of the value assigned to it.
7. The PLUS and MINUS operators in Expr represent addition and subtraction.
8. The STAR and SLASH operators in Prod represents multiplication and division.
9. **It is an error if a variable is used before a value is assigned to it**.
10. Addition is defined between two integers (the result being the sum) or two strings (the result being the concatenation).

11. Subtraction is defined between two integers (the result being the difference).
12. 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).
13. Division is defined between two integers
14. **Performing an operation with incorrect types or type combinations is an error.**
15. **Multiplying a string by a negative integer is an error.**
16. **An IF or LOOP statement whose Expr is not integer typed is an error.**

For Programming Assignment 4, you must 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.

In the event that a runtime error occurs, your program should throw an exception. This exception should be caught in the main routine and should be printed after the message RUNTIME ERROR.

Here are the error messages that should be generated as RUNTIME ERROR conditions in the event of a failure. The messages should be self-explanatory; where they are not, some extra explanation is included

Type mismatch for arguments of +
Type mismatch for arguments of -
Type mismatch for arguments of *
Repetition count less than 0 *generated if you multiply a string by a negative integer*
Invalid type for arguments of /
Divide by zero error
Conditional is not an integer *used in if statement*
Loop conditional is not an integer *used in loop statement*
Symbol INDENTIFIER not defined *where IDENTIFIER is the variable used but not defined*