# ICSI 311 Assignment 9 – Continue on with the Interpreter

**You must submit .java files. Any other file type will be ignored. Especially “.class” files.**

**You must not zip or otherwise compress your assignment. Blackboard will allow you to submit multiple files.**

***You must submit buildable .java files for credit.***

Create an “interpret” method that takes a StatementNode as a parameter. We will pass it the first statement from the StatementsNode.

Loop over the statements in the StatementsNode. Create a series of if statements using “is” to determine which type of statement the current statement is.

Consider some of the statement types:

ReadNode – for each variable, get the variable from the map, get the next value from the collection of data values. if the types don’t match, throw an exception. Set the variable to the type. Set the currentNode to the next node in the statements list. Set done if there are no more statements.

AssignNode – process the expression (more on this later). Lookup/create the variable and set its value in the map.

InputNode – use Java to print the string. Use Java appropriate input processing for input and assign the variable the values from Java.

PrintNode – for each child of Print, use Java to print the values.

Function Call – evaluate each of the parameters, then run whatever Java code to implement the function

The other type of Nodes that we have to consider are the MathOp nodes. Evaluate left and right, and then do the operation. Remember that the left and/or right sides could need evaluation. Much like the parsing, we will need to recursively consider each side. We don’t have to do multiple levels of reconsideration, like we did with parsing (Factor, Expression, Term), though.

Wrote an “EvaluateIntMathOp” that takes a Node returns an int. If the Node is an Integer, just return the value. If the Node is an integer variable, return the value of the variable. If the Node is a MathOp, call EvaluateIntMathOp on the Left and Right, then do the op (add, subtract, multiply or divide) on the results of each side. Return the result. Make a matching version for float.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rubric | Poor | OK | Good | Great |
| Comments | None/Excessive (0) | “What” not “Why”, few (5) | Some “what” comments or missing some (7) | Anything not obvious has reasoning (10) |
| Variable/Function naming | Single letters everywhere (0) | Lots of abbreviations (5) | Full words most of the time (8) | Full words, descriptive (10) |
| Expression | None work (0) | A few work (10) | Most work (18) | All work (30) |
| Functions | None work (0) | A few work (7) | Most work (13) | All work (25) |
| Input/Print/Read/Assign | Don’t exist (0) |  | Attempted (5) | Correct(25) |