CS4713 Program #3 (100 pts) due 3/23/2017 Parser

This is Part 3 of the Compiler/Interpreter Project. Larry provided (or will provide):

* p3Input.txt
* p3InputPrefix.txt (please let me know if you need this)
* Multiple additional input files to test some error handling

Note: Larry will only provide files which use infix expressions. If you are doing prefix, you must convert it and show it to Larry as soon as possible.

**If doing the interpreter:**

1. You must be able to execute:
   * **Assignment** operations to simple variables (Int, Float, Bool) and String variables.
   * Expressions involving one operator.
     + The data type of the result is based on the data type of the left operand.
   * Simple **if** statements, if-then-else, nested if statements
   * **while** statements
     + includes nesting of while statements and if statements
   * **print** function
   * **Declaration statements** for Int, Float, String, and Bool
2. You must be able to detect an error when referencing an **undeclared identifier**
3. You MUST decide whether you are doing **infix** or **prefix** for this (and all subsequent programs)
4. If doing infix, you must support **unary minus**.
5. Interpreter must support the following debugging aids (which can be turned on or off); the generated output should be indented two tabs and begin with "..."
   * bShowToken - print the token information for the current token returned by scan.getNext(); this is actually done in getNext()
   * bShowExpr - print the result of each expression evaluation for expressions involving at least one operator
   * bShowAssign - print the variable and value for an assignment
   * Provide a new HavaBol statement: debug *debugType onOff;*
     + Example:

// Simple if

i = 0;

print(">>> First If");

debug Expr on;

if loc == "TX": // check for TX

// It is TX

print("\tloc is", loc);

// it is ok to have this statement on multiple lines.

// That is why we have a semicolon

i

=

i

+ 1;

endif;

debug Expr off;

print("\tDone with first if");

1. Interpreter does not have to handle these for program #3
   * String # operator
   * String subscripts
   * **arrays**
   * complex expressions
   * **for** statement
   * array functions
   * **boundary checking** (e.g., subscript out of bounds)
   * LENGTH(*string*)
   * SPACES(*string*)
   * ELEM(*array*)
   * MAXELEM(*array*)

**If doing the compiler:**

1. Instead of executing, you will return a parse tree and print the tree.
2. Handle everything listed for the interpreter
3. Handle arrays.
   * declarations
   * referencing elements in expressions
   * scalar assignment
   * array to array assignment
   * **for** item in array
4. Compiler does not have to handle these for program #3

* String # operator
* String subscripts
* complex expressions
* **for** char in string
* for cv=sv to condition by incr
* array functions
* **boundary checking** (e.g., subscript out of bounds)
* LENGTH(*string*)
* SPACES(*string*)
* ELEM(*array*)
* MAXELEM(*array*

**Software recommendations:**

* Scanner, Token
* SymbolTable, STEntry, STFunction, STControl, STIdentifier
* Parser
* For the interpreter:
  + Numeric conversion class using Integer.parseInt(str) and Double.parseDouble(str)
    - returns int and double values
    - converts a ResultValue into a Numeric
  + New storage manager class which handles
    - storing variable values
    - accessing variable values
  + New utility software for executing +, -, U-, \*, /, ^, ==, <=, >=, <, >, !=
  + New utility software for other coercions

**Output:**

* Interpreter:
  + show the output generated by the sample code
  + detect errors
* Compiler:
  + show the parse tree
  + detect errors

**Notes:**

1. If you did not complete Program #1, you are not allowed to work in a group.
2. Cross group sharing of code prohibited and violations will result in a zero for this and subsequent programs.
3. When you include arrays in the parser, remember to add "[]" to your scanner.
4. **Everyone** on the team must upload a note stating:

- team name

- team leader (who will be turning in the assignment) as LastName, Firstname

- team members (max of 3) as LastName, FirstName; ...

The team leader must upload everything necessary for the assignment.

1. **Team Leader** must upload a zip file containing:

- each .java source file

- output files for each test file

- current decision file:

* doing interpreter or doing compiler
* doing prefix or doing infix

- a document specifying:

|  |  |  |
| --- | --- | --- |
| Source File Name | Primary Responsible Team Member | Reviewer |
|  |  |  |
|  |  |  |

1. It is assumed that the class HavaBol contains the main() method.
2. Your code must run correctly on a fox server.