CS4713 Program #4 (130 pts) due 4/11/2017 Core
This is Part 4 of the Compiler/Interpreter Project. In this part this, you will implement the entire core. Larry provided (or will provide):

- p4Array.txt
- p4Expr.txt
- p4Func.txt
- p4String.txt

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.
        - **arrays as targets with scalars or arrays as sources**
        - **array elements**
        - **string elements**
    - **Complex expressions** involving
        - many operations
        - multiple data types
        - simple builtin functions
            - LENGTH(*string*)
            - SPACES(*string*)
            - ELEM(*array*)
            - MAXELEM(*array*)
        - arrrays
        - array element references with boundary check
        - string element references with boundary check, including support for negative subscripts
        - operators:
            - numeric: + - * / ^
            - string: #
            - comparison: < > <= >= == != (note that the type of comparison is dependent on the left operand)
            - logical: not and or
    - If doing infix, identify unary minus
    - Bool constants: T F
    - Simple **if** statements, if-then-else, nested if statements
    - **while** statements
        - includes nesting of while statements, if statements, and for statements
    - **print** function
    - **Declaration statements**
        - Int, Float, String, and Bool
        - Arrays of Int, Float, String or Bool
        - Initializations (note that this was moved from the flexible requirements)

- **for** statements
    - counting with an optional by
    - char in a string
    - element in an array

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. 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");
```

5. xxx

Software recommendations:

- Scanner, Token
- SymbolTable, STEntry, STFunction, STControl, STIdentifier, Storage Manager
    - initGlobal
    - putSymbol
    - getSymbol
    - creation of a scalar vs array
    - initialization of an array

    - access an array element
    - access string character
    - access scalar variable
    - assign scalar variable
    - assign array element
    - assign string character
- Parser
- Utility, Numeric
    - arithmetic
    - string comparisons
    - numeric comparisons
    - maybe builtin functions
    - coercions
    - toInteger

Output:

- show the output generated by the sample code
- detect errors

Notes:

1. Cross group sharing of code prohibited and violations will result in a zero for this and subsequent programs.
2. When you include arrays in the parser, remember to add "[]" to your scanner.
3. Add **for**, **in**, **to**, and **by** to the symbol table.
4. Add **ELEM**, **MAXELEM**, **LENGTH**, and **SPACES** to the symbol table.
5. **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.
6. **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

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