The Interpreter

CSE 250 - Fall 2015

WARNING: This will not be easy. Please start early!

Note: The three deadlines are Monday, Wednesday, Friday for this assignment.

In this homework you will write an interpreter for the web programming language JavaScript. Writing a full interpreter for the language would be a monumental task, so we will implement a small subset of the languages features with many simplifying assumptions. For more information about the language here is an excellent JavaScript tutorial. It's not required to understand JavaScript (beside very basic syntax) to complete this assignment, though the tutorial contains editors where you can run code in a browser and compare the output with that of your interpreter which may prove useful.

The following language features will be interpreted in this homework:

- Tracking variables (numbers only)
- Printing to a document using “document.write”
- Computing infix expressions including parentheses
- Track and call user-defined functions
- Interpret if and else statements

The following is a list of simplifying assumptions for this assignment. These are not restrictions of JavaScript itself, but we will not violate these while testing during grading. The intent is to limit the amount of tedious string parsing involved in this assignment so you can spend more time on the data structures and logic of the project. This list might grow as questions arise:

- Functions only use local variables that are defined within the function or appear in the parameter list
- Variable names and function names cannot begin with a keyword
- All variables will be numbers which can be stored as C++ doubles
- document.write will always be called with a single variable or a single string
- No direct negative values (can have “0-5” not “-5”)
- Return is always a single variable (can have “return x” but no “return 1”)
- No function calls in infix or if statements. All function calls will be on a line of its own, with or without an assignment (“var y = myFunction(x)”).
- No immediately nested functions (can't have “function1(function2(five))”)

- Only number variables for function calls (can't have "function(5)")
- All JavaScript code can be assumed to be well-formed (no testing with syntax errors)
- The opening '{' is always on the same line as the function, if, or else that it opens
- Can assume } is on a line by itself except for if statements
- When a } closes an if statement it is either "}", "} else if {" or "} else {"
- Comparisons are only > and < (no ||, &&, ==)
- Comparisons always compare exactly 2 variables

We provide sample JavaScript code for each part to test the functionality of your interpreter, though these are not meant to comprehensively test all the functionality of the interpreter. You should also write your own programs to test for full functionality. You are encouraged to share sample JavaScript code for this assignment with others and can even post sample programs on Piazza, but do not share the C++ code for the interpreter.

Some string parsing code is provided in the `Parser` class. While you don't need to understand everything in `Parser.cpp`, you should read `Parser.h` to know how to use the provided functions. It's not required that you use any of these functions, though they can significantly reduce the amount of string parsing you'll need to implement. Beyond this, there is much freedom in how you implement the interpreter. This will be a fairly large program and you will need to make several design decisions while coding. There are a few suggestions in the code to help you get started, though you can go in different direction if you'd like.

The automatic feedback for this homework will not return a grade. We will be using this feedback for its original intended purpose and only give feedback on whether your code ran with or without errors. The idea is to let you know if it can run in the grading environment so you don't get a surprise 0 for mostly working code. To test for proper functionality of your code, run sample JavaScript programs that following the simplifying assumptions and verify the output.

The provided makefile compiles your code into an executable names `Interpreter`. The usage of this executable is "`./Interpreter <inputFile.js> <outputFile>`".

Relevant code can be found in the course repository:
https://bitbucket.org/hartloff/cse250-fall2015/src/1439472551c0/homework/Interpreter/?at=master

Part 1

Due: Wednesday, November 18 @ 11:59pm

Your interpreter must have the following functionality:

- Declare number variables using the “var” keyword
- Assign values to variables using infix notation to compute +,-,*,/ on numbers with or without parenthesis
- Output strings and numbers to an output file using “document.write”

Part 2

Due: Friday, November 20 @ 11:59pm

Your interpreter must have the functionality from part 1 as well as the following:

- Interpret and call user-defined functions using the “function” keyword
- Handle nested function calls

Part 3

Due: Monday, November 23 @ 11:59pm

Your interpreter must have the functionality from parts 1 and 2 as well as the following:

- The “if” statement with “else” and “else if”
- Handle nested if/else blocks

Submission

Submit `Interpreter.cpp`, `Interpreter.h`, `UserFunction.cpp`, and `UserFunction.h` using the `submit_cse250` command. You can define more classes and functions as long as they are contained in these 4 files. Do not edit `Makefile`, `Parser.cpp`, `Parser.h`, `run.cpp`, or `run.h`.