Programming Languages and Paradigms

COMP 302, Fall 2016

Assignment 2

Due date: Monday, October 24, 2016
6pm

In this assignment you will implement a simple, top-down parser in JavaScript for the "WML" template language described in class. A(n optional) framework you can use to test your code is provided through the files `wml.html` and `wml.js`, or you may adapt the evaluator code written by some of your TAs and linked in *MyCourses*.

Your code must run without error or modification in current versions of Firefox or Chrome: restrict yourself to basic JavaScript idioms defined in the ECMA standard 5.1. Be sure to respect precise naming, input, and output requirements.

All code should be well-commented, in a professional style, with appropriate variables names, indenting (uses spaces and avoid tabs), etc. **The onus is on you to ensure your code is clear and readable. Marks will be very generously deducted for bad style or lack of clarity.**

1. Define appropriate regular expressions for recognizing the tokens for the WML grammar developed in class. You should define `RegExp`-type variables for each of the all-capital named tokens. **6**

   Note that "anything" in some of the token descriptions includes newlines!

2. Implement a basic scanner to recognize your tokens within a `scan(s,tokenset)` function, which receives a string and an allowed set of tokens. The tokenset is an object with keys being the token names, and the value being `true`. For example, a token set containing just `DSTART` and `DEND` would look like **11**

   ```
   { DSTART: true,
     DEND: true }
   ```

   Your `scan` function should return an object describing the token at the start of the string. This object should include (at least) 2 fields,

   ```
   { token:  tokenname,
     value:  tokenvalue }
   ```

   Where `tokenname` is the string name of a token (e.g., `"DSTART"`), and `tokenvalue` is a string representing the literal characters that were used to match the token.

   Your scanner should be able to recognize all components of template invocations, definitions, and parameters.

   Verify your scanner works. Ensure that given the following driver function, and a string `s` composed of characters that would match tokens in `TOKENSET`, it should be the case that `scanit(s)==s`.

   ```
   function scanit(s) {
      var sout = ""; {
      while(s) {
         var t = scan(s,TOKENSET);
         sout += t.tokenvalue;
         s = s.substr(t.tokenvalue.length);
      }
      return sout;
   }
   ```

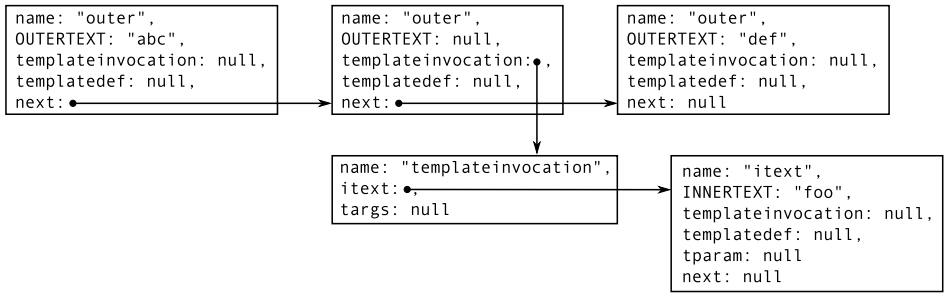
3. Implement functions to parse WML input. You should (at least) include functions, **17**

   - `parseOuter(s)`
   - `parseTemplateInvocation(s)`
   - `parseTemplateDef(s)`
   - `parseTParam(s)`

   each of which receives a string and returns a parse tree (AST) of the result. Note that the input is a string, so you will need your `scan` function from the previous question. You may assume the input is syntactically correct WML text.

   The AST should be represented by a hierarchy of objects, one object per grammar rule applied in the parse. Each object should include a `name` field a string representation of the rule name (non-terminal), and have fields for each (non-terminal) child object, including tokens where the content depends on the input string (`OUTERTEXT`, `INNERTEXT`, `INNERDTEXT`, `DNAME`, `PNAME`). Where choice is possible, unused fields should have the value `null`.

   Some grammar rules have iterated structure, specified by $\star$. In these cases form a linked list, by including a `next` field that points to the next instance, terminating in `null`. For example, the input string `abc{{foo}}def` would be parsed into an AST like:



   Note that template names, arguments, and parameters in the AST should not have leading or trailing whitespace (but template bodies may).

4. To verify your parse, define a function, `printAST(a)` which receives an AST node (as returned from any of your functions in the previous question), and returns a string representation of the input. You may format the string output however you like, but it should be the case that **6**

   ```
   printAST(parse(s)) == printAST(parse(printAST(parse(s))))
   ```

What to hand in

Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**. Assignments must be submitted on the due date **before 6pm**.

For each question n, include a file `qn.js` with the source code of your answer. You can assume your code from prior answers will be included in evaluating each answer. Do not include the provided files (`wml.html` and `wml.js`).

This assignment is worth 10% of your final grade. **40**