CS 331 – Assignment 4

Due: 11:59pm Wednesday, March 19 (two days before your next exam)

There are five problems on this assignment. You will submit your answers to the first four in a PDF file named *assign4.pdf*. The first four problems do not require you to "code" as you have on previous assignments. Instead you will just providing grammars to parse languages according to the specifications I have given. You should use BNF and EBNF notation as we have developed it in class notes. The PDF file must have been prepared through some document processing tool – TitanApps, Latex, OpenOffice, even MSWord. Whatever tool you use, the submission must be as a PDF file, and it cannot be a scanned document of hand-written answers.

The fifth problem requires you to develop a JISON grammar to parse assignment expressions in a Java-like language. Your solution to this problem should be submitted as a JISON grammar file called *assign4.jison*. Here is precisely the way we will test it on the campus Linux workstations:

```
$ jison assign4.jison
$ node
> .load assign4.js
.
.
.
```

lots of node stuff produced here as parser loads

```
> require('util').puts(require('util').inspect(assign4.parse("a = b = x + (2 ^ z) ^ y ^ d % 3"), false ,10,true))
[ '=',
  'a',
  [ '=',
    'b',
    [ '+',
      'x',
      [ '%',
        [ '^',
          [ '^', 2, 'z' ],
          [ '^', 'y', 'd' ] ],
        3 ] ] ] ]
undefined
>
.
.
.
```

Many more test cases follow

The first four problems will each be graded on a 10-point scale. The fifth problem will be evaluated on a 40-point scale determined by how many test cases are passed by your solution. If JISON does not successfully parse your BNF grammar to produce a parser, you will receive no credit for Problem 5.

1. Give a BNF grammar for the set of all strings consisting of the keyword `begin`, followed by one or more statements with a semi-colon after each statement, followed by the keyword `end`. Use the non-terminal `<statement>` for a statement and do not give a production for it.
2. Repeat Problem 1 except now use EBNF extensions wherever possible to simplify the grammar.
3. Write a BNF grammar that can generate all strings in the language $a^n c^k b^m$ where $n > m \geq 1$ and $k = 0, 1$, or 2. In other words, there are more a's than b's. There is at least 1 b. The a's and b's may be separated by nothing, 1 c or 2 c's. Hint: In the review problems you've seen a grammar for a language whose "sentences" are a string of a's followed by an equal number of b's. Think about how to extend that grammar to satisfy the specifications of this problem.
4. Repeat Problem 3, but now use EBNF extensions to make your grammar as concise as possible.
5. Consider a subset of Java assignment statements that allow expressions using the operators `= + - * / %`. Java establishes the precedence and associativity of these operators, and in your work on this assignment you should use Java's conventions in this regard. To these six operators, add a right associative exponentiation operator `^` whose precedence is higher than any other operator. Your new language thus has seven operators – we'll call it "Double-O-7" . Starting from the *demo4.jison* grammar we discussed in class on March 12 (available in the code snippets on D2L), extend that grammar to correctly produce abstract syntax trees for expressions in "Double-O-7". These abstract syntax trees should be patterned after those already existing in *demo4.jison*, that is a sub-tree for an operation should be returned as a list of three items – first the operator, then the result from the first operand, followed by the result of the second operand.

   Be sure that your grammar for "Double-O-7" doesn't allow assignment statements of the following form:

   ```
   a = b + 10 = c % 13
   ```

   Important ground rule for your work on this problem: If you google for help on this assignment, you'll no doubt learn that JISON has "add-ons" in the form of `%left` and `%right` directives that enable you to control operator precedence and associativity. You are not allowed to use them. I want you control "Double-O-7" operator precedence and associativity completely through the BNF specification you give to JISON.