Assignment 1

CSE 3341

Due Wednesday, September 5, by 2:45 pm

1. (5 pts) The course web page has a link to the specification of the Java language (under "Resources"). Section 3.10.1 of the spec describes the lexical structure of *integer literals*. Such literals define one category of *tokens* produced by the scanner. The description uses a grammar notation slightly different from what was defined in class; see Section 2.4 of the spec for details.

   Using only the notation described on slide 8 of the lecture notes, write a regular expression that represents all and only valid integer literals in Java. Note that the spec does not allow literals that are "too big" - e.g., literals with `int` values that do not fit in 32 bits. Your solution does **not** have to account for such restrictions: as long as the literal conforms to the lexical structure, it is acceptable regardless of its value.

2. (5 pts) Consider the following grammar for hexadecimal integers:

   $\langle \textit{hex\_literal} \rangle$ ::= `0x`$\langle \textit{number} \rangle$
   $\langle \textit{number} \rangle$ ::= $\langle \textit{number} \rangle\langle \textit{number} \rangle$ | $\langle \textit{digit} \rangle$
   $\langle \textit{digit} \rangle$ ::= `0` | `1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9` | `A` | `B` | `C` | `D` | `E` | `F`

   String `0xF3` is an element of the language generated by this grammar. How many *distinct derivation sequences* exist for this string? How many *distinct parse trees* exist for this string? Show all parse trees.

3. (5 pts) Consider the following grammar of identifiers:

   $\langle \textit{id} \rangle$ ::= $\langle \textit{letter} \rangle\langle \textit{id} \rangle$ | $\langle \textit{letter} \rangle$`_`$\langle \textit{id} \rangle$ | $\langle \textit{letter} \rangle$
   $\langle \textit{letter} \rangle$ ::= `A` | `B` | … | `Z`

   This grammar allows `A`, `A_B`, and `A_B_C` as legal identifiers, but not `A_`, `A___B`, and `_B`. Change the grammar so that sequences of `_` are allowed in the middle of an identifier (as in `A___B__CD`), although the identifier must not start or end with a `_`.

4. (5 pts) Consider the following grammar of expressions:

   $\langle \textit{exp} \rangle$ ::= `id` | `hex_literal` | $\langle \textit{exp} \rangle$ `+` $\langle \textit{exp} \rangle$ | $\langle \textit{exp} \rangle$ `*` $\langle \textit{exp} \rangle$

   In this grammar, `id` and `hex_literal` are terminal symbols—that is, tokens that will be produced by the scanner and consumed by the parser.

   Change this grammar to achieve the following goals. First, make the precedence of operator `+` lower than the precedence of operator `*`. Second, make operators `+` and `*` left-associative. Third, add a *pre-increment* operator `++` to represent expressions of the form `++A`, similarly to languages such as C, C++, Java, and C#. The pre-increment operator applies *only to identifiers*, and has higher precedence than `+` and `*`.

   Show a modified grammar that achieves these goals. Note that `++` is represented by its own terminal symbol, not by two terminal symbols for the `+` operator. Show a parse tree for expression `0x3FA + ++ A + ++ B * ++ C`. Your new grammar should be non-ambiguous.

(Please turn over)

- Assignments should be done independently. General high-level discussion of assignments with others in the class is allowed, but **all of the actual work** should be your own. Assignments that show excessive similarities will be taken as evidence of cheating and dealt with accordingly.
- Assignments should be turned in **by the end of class** on the due day. Late assignments turned in by the end of the next class will be graded with 30% reduction. Assignments turned in later than that will not be accepted.
- Make the assignments readable and understandable. They should be handed in on regular paper, legibly written or typed. If you have more than one sheet, **staple the sheets together**. If the grader has trouble reading or understanding what you have done, points will be deducted even if it can finally be determined that you have the correct answer.
- Your solutions have to be **precise and detailed**: you have to work out **all** details that are necessary to solve the problem using the approaches discussed in class. You also have to write your solutions in a way that convinces the grader that you understand all these details. Be careful, precise, and thorough.