CSE 3341, Assignment #1
Due: Sept. 13

Use only pure BNF, i.e., the version of BNF that we have been using in class, for solving these problems.

1. (6 points). Define a BNF grammar for generating the set of all those unsigned decimal integers in which each 0 is followed by a 1. Thus 5, 501, 55011016, 01, 011 are all legal; but 50, 0, 10, 502, 510 are all illegal. (Thus a valid number should consist of any number of decimal digits, and every 0 in the number should be immediately followed by a 1.)

2. (6 points). Consider the following grammar of identifiers:

\[
\begin{align*}
\langle id \rangle & ::= \langle let \rangle \ | \ \langle let \rangle \langle id \rangle \ | \ \langle let \rangle \_\langle id \rangle \\
\langle let \rangle & ::= A \ | \ B \ | \ C \ | \ D \ | \ldots \ | X \ | Y \ | Z
\end{align*}
\]

This allows A, A\_B, A\_B\_C etc. as legal identifiers, but not A\_, A\_B, \_B etc. Change the grammar so that a series of \_’s is allowed in the middle of an identifier, although the identifier must not start or end with a \_. Thus the new grammar should generate all identifiers that start and end with letters, the other characters in the identifier being letters or \_. (An identifier with a single letter is also legal.) Examples of invalid identifiers would \_A, \_A\_B, \_A\_B\_ etc.

3. (8 points). Consider the following grammar of expressions:

\[
\begin{align*}
\langle exp \rangle & ::= \langle id \rangle \ | \ \langle no \rangle \ | \ \langle exp \rangle + \langle exp \rangle \ | \ \langle exp \rangle * \langle exp \rangle \\
\langle id \rangle & ::= X \ | \ Y \ | Z
\end{align*}
\]

\langle no \rangle denotes unsigned integers as in the class notes/discussions. Rewrite this grammar such that the operations will be evaluated strictly left to right; i.e., \( X + Y \times Z \) will be evaluated as if it was \(( X + Y ) \times Z\); and \( X \times Y + Z \) will be evaluated as if it was \(( X \times Y ) + Z\). But do NOT introduce parentheses or other new terminal symbols into the language.

(Note: The language APL does something like this.)