1. Use the MyPL token types to give the token stream for the code snippet below. Give the type, lexeme, line, and column of each token. E.g., WHILE("while", 1, 1), ID("x", 1, 7), etc.

```plaintext
while x > 5 {
    x = f(x, y)
    if (x == nil) {
        return
    }
}
```

2. Using the simple “statement language” from the lecture, give a left-most derivation of \( A = B \). Show all derivation steps.

3. Using the simple “statement language” from the lecture, give a right-most derivation of \( A = B \). Show all derivation steps.

4. Using the simple “statement language” from the lecture, give a derivation of \( A = B \) that is neither left-most nor right-most. Show all derivation steps.
5. Develop a grammar for each of the following simple languages.

(a). The set of strings consisting of (only) zero or more a’s followed by a semicolon.

(b). The set of strings that consist of an odd number of a’s followed by the same number of b’s.

(c). The set of strings for performing zero or more sequences of addition operations over the values 0 and 1, where, 1 + 1, 1 + 0 + 1, 0 + 0 + 1 + 0, and so on are all in the language.

6. Develop a grammar to capture Boolean expressions over values true and false; single-letter lower-case variables (e.g., x, y, z, and so on); conjunction (and), disjunction (or), and logical negation (not); and optionally parenthesized subexpressions. The following are some examples of well-formed strings in the language (one per line):

   true
   x and y
   (x or y) and not true
   (not (x or y or z))
   not x or y and z or true
   not not true

Note your grammar should only allow well-formed Boolean expressions.