(Problems from *Modern Compiler Implementation in Java* by Andrew Appel)

1. Write an unambiguous grammar for each of the following languages.
   a. Palindromes over the alphabet {a, b} (strings that are the same backward and forward)
   b. Strings that match the regular expression a*b* and have more a’s than b’s.
   c. Balanced parenthesis and square brackets. Example [[[(()][()]]]].
   d. Statement blocks in C where the semicolons terminate the statements: { expression; expression; } expression

2. Problem 2.13 from the textbook by M.L Scott.

3. Write a grammar that accepts the same language as the grammar below, but that is suitable for LL(1) parsing. That is, eliminate the left recursion, and (if necessary) left-factor.

   \[ S \rightarrow S \, ; \, S \\
   S \rightarrow \text{id} \, := \, E \\
   S \rightarrow \text{print} \, ( \, L \, ) \\
   E \rightarrow \text{id} \\
   E \rightarrow \text{num} \\
   E \rightarrow E \, + \, E \\
   E \rightarrow ( \, S \, , \, E \, ) \\
   L \rightarrow E \\
   L \rightarrow L \, , \, E \]

4. a. Calculate FIRST and FOLLOW set for this grammar

   \[ S \rightarrow u \, B \, D \, z \\
   B \rightarrow B \, v \\
   B \rightarrow w \\
   D \rightarrow E \, F \\
   E \rightarrow y \\
   E \rightarrow \epsilon \\
   F \rightarrow x \\
   F \rightarrow \epsilon \]

   b. Construct the LL(1) parsing table

   c. Give evidence that the grammar is not LL(1)

   d. Modify the grammar as little as possible to make an LL(1) grammar that accepts the same language.

5. Problem 2.19 and 2.20 from the textbook.

6. Problem 2.21 from the textbook.