1. Calculate the first sets for S, A, and B in the following grammar:

   S $\rightarrow$ Adg | Bh

   A $\rightarrow$ aBc | ϵ

   B $\rightarrow$ f | g

2. Rewrite the following grammars so they can be parsed by a top–down recursive descent parser by eliminating left recursion and applying left factoring where necessary. (Keep in mind that left recurion **only** needs to be elmininated in a grammar if you want to write a top–down recursive descent parser for it.)

   a. S $\rightarrow$ S + a | b

   b. S $\rightarrow$ S + a | S + b | c

   c. S $\rightarrow$ abc | ac

   d. S $\rightarrow$ aa | ab | a

   e. S $\rightarrow$ aSc | aSb | b

3. Consider the following grammar: S $\rightarrow$ S and S | S or S | (S) | true | false

   a. Give the first sets for each production and nonterminal in the grammar.

   b. Show why the grammar cannot be parsed by a top–down recursive descent parser.

4. Consider the following grammar: S $\rightarrow$ abS | acS | c

   a. Give the first sets for each production and nonterminal in the grammar.

   b. Show why the grammar cannot be parsed by a top–down recursive descent parser.

   c. Rewrite the grammar so it can be parsed by a top–down recursive descent parser.

   d. Write a top–down recursive descent parser for the modified grammar. You may use the pseudocode notation from the examples in class, and may assume the existencee of functions `match()` and `error()`, whose effects are as explained in class.