Problem 1: Perform the pairwise disjointness test for the following grammar rules.
   a. \[ A \rightarrow aB \mid b \mid cBB \]
   b. \[ B \rightarrow aB \mid bA \mid aBb \]
   c. \[ C \rightarrow aaA \mid b \mid caB \]

Problem 2: Show a trace of the recursive descent parser given in Section 4.4.1 for the string
   \[ a + b * c \].

Problem 3: Given the following grammar and the right sentential form, draw a parse tree and
   show the phrases and simple phrases, as well as the handle.
   \[ S \rightarrow aAb \mid bBA \]
   \[ A \rightarrow ab \mid aAB \]
   \[ B \rightarrow aB \mid b \]
   a. \[ aaAbb \]
   b. \[ bBab \]
   c. \[ aaAbBb \]

Programming Assignments

1) Design a state diagram to recognize one form of the comments of the C-based
   programming languages, those that begin with /* and end with */.
   Write and test the code to implement the state diagram of part a.

2) Write and test a lexical analyzer (lex) similar to the one given in section 4.2. To test lex, write
   a test application that receives a simple program as its input (as a string) and calls lex to analyze
   all the lexemes of the input program (this application is similar to a syntax analyzer—but do not
   develop syntax analyzer’s functionalities!). You have to define a small set of “build in
   keywords” to be able to develop “lookup (lexeme)” function.