1. **First and Follow Sets and CFSM**

   A. Construct the first and follow sets for the following grammar.
   
   \[ \text{Lexp} \rightarrow \text{Atom} \mid \text{List} \]
   
   \[ \text{Atom} \rightarrow \text{number} \mid \text{identifier} \]
   
   \[ \text{List} \rightarrow ( \text{Lexpseq} ) \]
   
   \[ \text{Lexpseq} \rightarrow \text{Lexp Lexpseq} \mid \varepsilon \]

   where the terminals are \{number, identifier, (, )\}, and the starting symbol is `Lexp`.

   B. Construct the CFSM for the following grammar.

   \[ \text{S} \rightarrow ( \text{L} ) \mid a \]
   
   \[ \text{L} \rightarrow \text{S L} \mid \varepsilon \]

   The alphabet is \{(, a\}.\

2. **Grammers for LL Parsing**

   A. Eliminating left-recursion. LL parsers cannot deal with left-recursion, they go into an infinite loop. For each of the following grammers, remove its left-recursion so that it’s LL parsable.

   i. \[ \text{S} \rightarrow \text{S} + \text{S} | \text{S} - \text{S} | \text{T} \]
      
      \[ \text{T} \rightarrow \text{T} \times \text{T} | \text{T} / \text{T} | \text{I} \]
      
      \[ \text{I} \rightarrow \text{I0} | 1 | \text{I1} | 1 \]

   ii. \[ \text{S} \rightarrow \text{Sa} | \text{Gb} | \text{c} \]
       
       \[ \text{T} \rightarrow \text{Tx} | \text{Sy} | \z \]

   B. Eliminating common prefixes (left factoring). LL parsers also cannot handle common prefixes. Use left factoring to remove the common prefixes in the following grammar and make it LL parsable.
\[ S \rightarrow xyA|ByB|v \]
\[ A \rightarrow zB|zx|w \]
\[ B \rightarrow y|x \]

C. For each of the following grammars, indicate whether it is LL. If it is not LL, fix it so that it is.

i. \[ S \rightarrow Sxy \]
   \[ S \rightarrow SxB \]
   \[ S \rightarrow q \]
   \[ B \rightarrow Sz \]

ii. \[ S \rightarrow Aw \]
    \[ S \rightarrow Bx \]
    \[ A \rightarrow yB \]
    \[ B \rightarrow zA \]

iii. \[ S \rightarrow Ax \]
     \[ S \rightarrow By \]
     \[ A \rightarrow zB \]
     \[ B \rightarrow z \]

3. Writing Recursive-Descent Parsers

Write pseudocode to parse the following grammar by recursive-descent.
\[ S \rightarrow (S)S|\varepsilon \]