Lab Assignment 3

Syntax Analysis/Parsing

NOTE: Refer lecture notes, Chapter 4.

Q1. Write a program to remove left-recursion from any grammar G given as input.

Example Input:

E -> E + T | T
T -> T * F |F
F -> (E) | id

Example Output:

E -> TE'
E' -> +TE' | ϵ
T -> FT'
T' -> *FT' | ϵ
F -> (E) | id

Q2. Write a program that takes a grammar as input and produces an equivalent left-factored grammar as output.

Example Input:

A → aAB | aBc | aAc

Example Output:

A → aA'

A' → AD / Bc

D → B / c

Q3. We discussed about a basic top-down parsing approach (**Recursive-descent parsing**) that may require backtracking. Implement a recursive descent parser for the following expression grammar with non-terminals {**E, E', T, T', F**}, and start symbol **E**:

E -> TE'
E' -> +TE' | ϵ
T -> FT'
T' -> *FT' | ϵ
F -> (E) | id

Given any string of terminal symbols, the parser should answer whether it is accepted or not accepted. Display the execution of the parser for any given input. Display "(Rule applied, current sentential form, and the remaining input)".