Homework 3 - LL Grammars

Question 1

The grammar does not satisfy the LL(1) condition, because it contains an indirect left recursion, where the grammar recursively alternates between B and C.

Solution:
Substitue B into C:
1. C ::= C `ri` | `ti` | `t`

To resolve the left recursion, we create recursive tail from suffix of recursive production:
2. RTail ::= `ri` RTail

Append Tail to non-recursive productions:
1. C ::= `ti` RTail | `t` RTail
2. RTail ::= `ri` RTail

Add empty string (ϵ) as a rhs for the tail production:
1. C ::= `ti` RTail | `t` RTail
2. RTail ::= `ri` RTail | ϵ

To resolve the FIRST conflict in C, we factor out the prefix `t`:
1. C ::= `t` FTail
2. RTail ::= `ri` RTail | ϵ
3. FTail ::= `i` RTail | RTail

Reorder the productions to make the grammar more readable:
1. C ::= `t` FTail
2. FTail ::= `i` RTail | RTail
3. RTail ::= `ri` RTail | ϵ

The new grammar is:
0. A ::= `s` C `ng` | ϵ
1. C ::= `t` FTail
2. FTail ::= `i` RTail | RTail
3. RTail ::= `ri` RTail | ϵ

Question 2

0. S ::= S;S | `id:=` E | `print(`L`)`
1. E ::= `id` | `num` | E + E | `(`S`,`E`)`
2. L ::= E | L`,`E

Resolve the left recursion in S, E, and L:
Create recursive tail from suffix of recursive production:
3. STail ::= ;S STail
4. ETail ::= +E ETail
5. LTail ::= ,E LTail

Append Tail to non-recursive productions and add empty string (ϵ) as a rhs for the tail production:
0. S ::= `id:=` E STail | `print(`L`)` STail
1. E ::= `id` ETail | `num` ETail | `(`S`,`E`)` ETail
2. L ::= E LTail
3. STail ::= ;S STail | ϵ
4. ETail ::= +E ETail | ϵ
5. LTail ::= ,E LTail | ϵ

The new grammar is:
0. S ::= `id:=` E STail | `print(`L`)` STail
1. E ::= `id` ETail | `num` ETail | `(`S`,`E`)` ETail
2. L ::= E LTail
3. STail ::= ;S STail | ϵ
4. ETail ::= +E ETail | ϵ
5. LTail ::= ,E LTail | ϵ