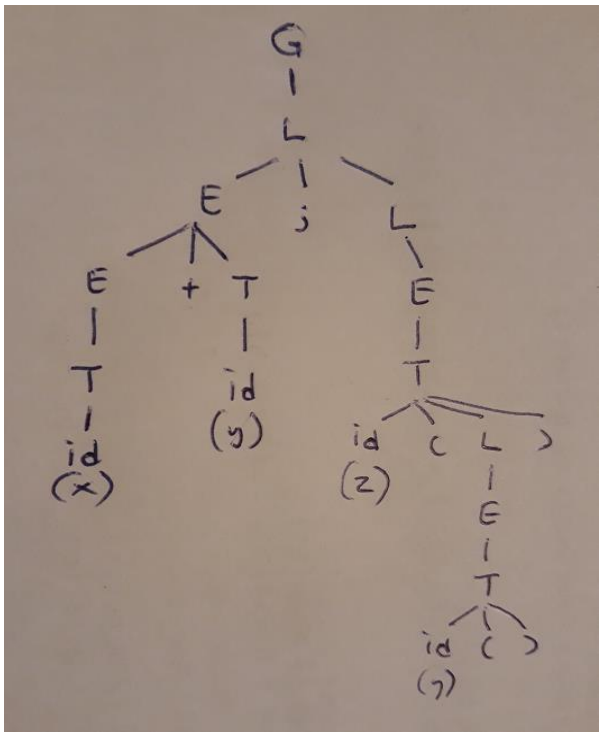
FORMATIVE EXERCISES (2)

i) Leftmost derivation:
G → L → E ; L → E+T ; L →
➔ T + T ; L →
➔ id + T ; L →
➔ id + id ; L →
➔ id + id; E →
➔ id + id; T →
➔ id + id; id (L) →
➔ id + id; id (E; L)→
➔ id + id; id (T; L) →
➔ id + id; id (id; L) →
➔ id+id; id (id; E) →
➔ id+id; id(id; T) →
➔ id+id; id (id; id( ))

Parse tree:



ii) to transform this grammar so that it can be used to construct a top-down predictive parser with one symbol of lookahead we first need to eliminate left recursion as a result of the rules E → E + T | T, which become: E → T E' and E' → + T E' | ε
Then, we have to eliminate common prefixes to make sure that the LL(1) property holds. We do this by factoring and rules: L → E ; L, L → E become: L→ E L' and L' → ;L | ε. Same also for rules T → id, T → id ( ), T → id (L), which become: T → id T' and T' → (T'' | ε and T'' →) | L).

b) The relevant steps taken are:

Stack	Input	Action taken
$ 0	(( )) ( )	Shift 3
$ 0 (3	( )) ( )	Shift 6
$ 0 (3 (6	)) ( )	Shift 10
$ 0 (3 (6) 10	) ( )	Reduce 5
$ 0 (3 P 5	) ( )	Shift 8
$ 0 (3 P 5) 8	( )	Reduce 4
$ 0 P 2	( )	Reduce 3
$ 0 L 1	( )	Shift 3
$ 0 L 1 (3	)	Shift 7
$ 0 L 1 (3) 7	eof	Reduce 5
$ 0 L 1 P 4	eof	Reduce 2
$ 0 L 1	eof	accept

c) Clearly the grammar doesn't have the LL(1) property as there are two rules for the same left-hand side symbol whose right-hand side can start with the same terminal symbol. These are rules 1 and 2: in both these rules, the right-hand side can start with b.

d) It is sufficient to show that there are two different leftmost (or rightmost) derivations for the same output string (or two different parse trees). The easiest string to consider is abab