Classtest-1

M. Sanjay,
1904119.

',' is a terminal.

(1)

FIRST(T) = {id, ε}

FIRST(ArgL) = {id, ε}

FIRST(Args) = {ε, id}

FIRST(Func) = {id}

---

(2) We know; a Grammar is Left associative

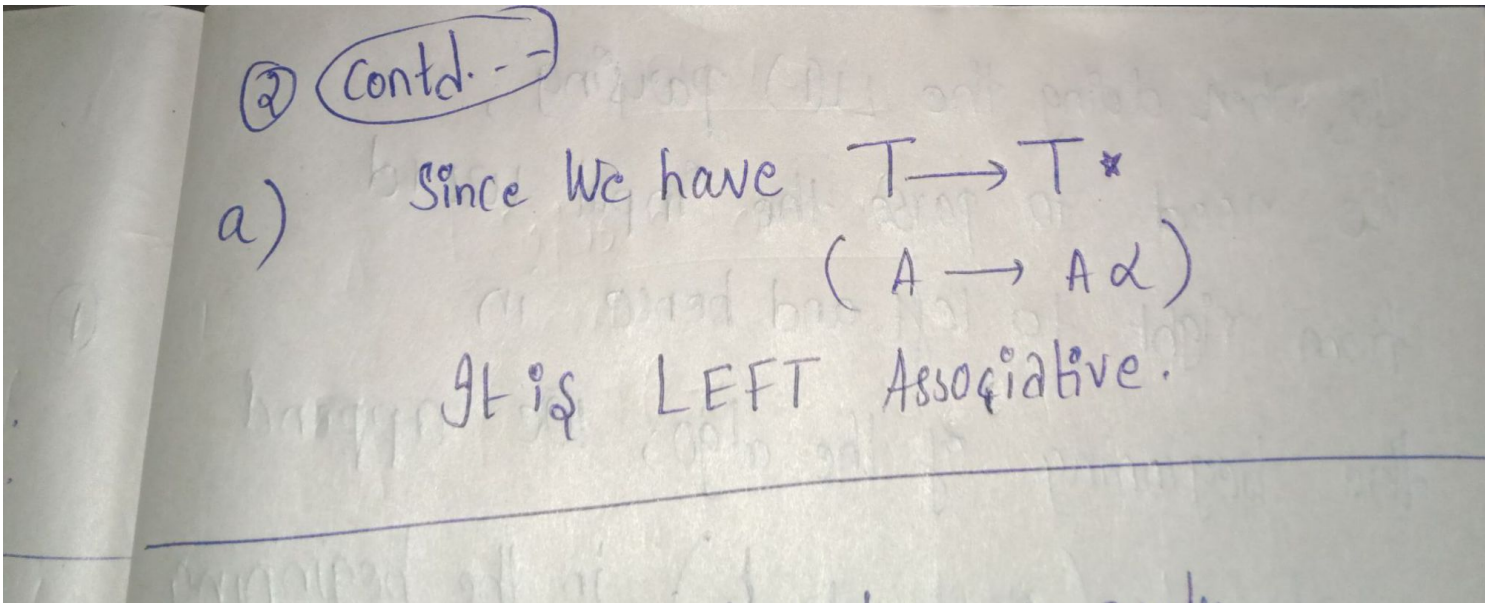
a) if the production is LEFT RECURSIVE

Left Recursive has production of form:-

$$A \longrightarrow A\alpha \mid \beta$$

~~Since, NOT found~~

It is ~~NOT LEFT Associative~~


② Contd. - -
a) Since We have T ⟶ T*
(A ⟶ Aα)
It is LEFT Associative.

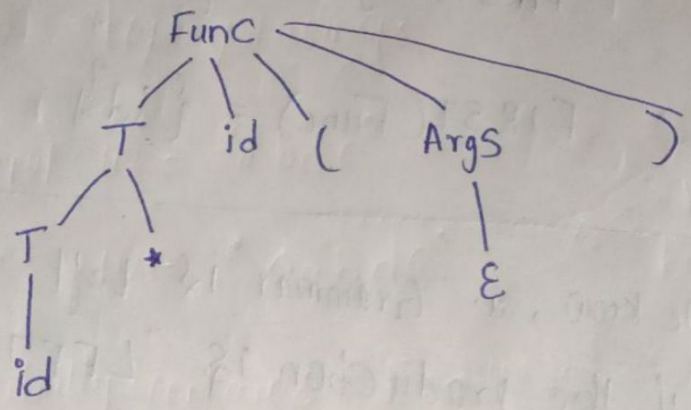
② b) $A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \varepsilon$

It is found in 1(c) rule.

So It is RIGHT Associative. also.

② c) id*id( )



② d) From the above Parse tree, It looks like LEFT ASSOCIATIVE. But we can't infer anything about associativity of grammar, just from the parse tree bcoz different word might have different Parse Tree. We need to look for Production rules.

(3.) In Right most derivation, we apply Production to the right most variable in each step in LL(1) parsing.

If we have for example:- the following grammar.

$E \rightarrow E+E$

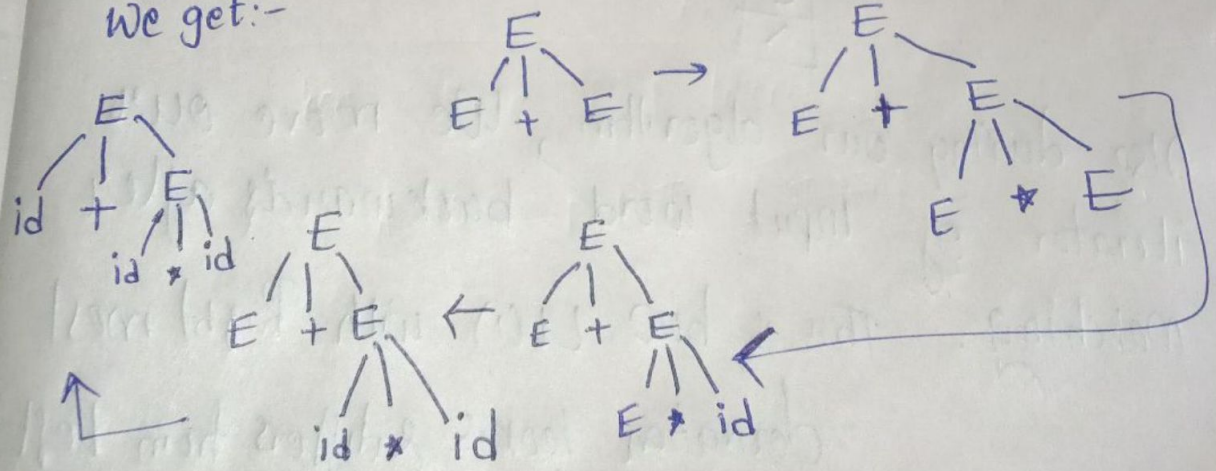
$E \rightarrow E*E$

$E \rightarrow -E$

$E \rightarrow (E)$

$E \rightarrow id$

To generate **id + id*id** with RIGHTMOST, we get:-

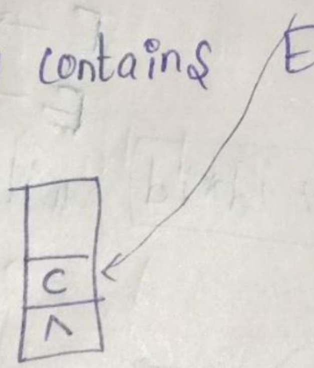


So; when doing the LL(1) parsing;
We need to parse the input word from right to left and hence in the beginning of the algo; we append a character (similar to $) in the beginning of input word so that we know when to stop parsing.

Ex:- if input w = 'abdc'

w = '^abdc'   ^ → Marks beginning of word.

stack initially contains End word & ^



Also during our algorithm, we move our iterator of input word backwards after matching. This is how LL(1) with Right most derivation looks & differs from left.