LL(1) parser🡪 left to right, leftmost derivation , 1 lookahed (Predictive Parser)

E🡪E+T|T

T->T\*F|F

F🡪(E) |id | s

Solution:

1. **Remove the left recursion**

E->TE’

E’->+TE’ | €

T->FT’

T’->\*FT’ | €

F-> (E) | id

1. **Compute first()**

First (F) = (first { ( } union first { id } ) = { (, id }

First (T’) = (first { \* } union first { € }) = { \*, € }

First(T) =(fisrt { F }} = { (, id }

First (E’) =( first ( + } union first { € } = { +, € }

First (E) =( first {T }) = { (, id }

**Compute follow()**

Follow(E)= ( $, first { ) }) = { $, ) }

Follow (E’)= ( follow { E}) union follow (E’)} ={ $, ) }

Follow (T)= ( first { E’ } union first {E’ })

=( + , € union follow (E) union + , €, follow( E’ ) }

={+, $, ) }

Follow(T’)=( follow (T) union follow (T’) )

={ + , $, ) }

Follow (F)=( first ( T’) union first ( T’) }

={ \*, € follow (T) union \* , €, follow( T’)

={ \*, +, $, )

**Step 3 design the parsing Table🡪**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nonterminal/Terminal** | **+** | **\*** | **(** | **)** | **Id** | **$** |
| **E** |  |  | E->TE’ |  | E->TE’ |  |
| **E’** | E’->+TE |  |  | **E’-**€ |  | **E’->**€ |
| **T** |  |  | T->FT’ |  | T->FT’ |  |
| **T’** | **T’->**€ | T’->\*FT’ |  | **T’->**€ |  | **T’->**€ |
| **F** |  |  | F-> (E) |  | F-> id |  |

**As the parsing table don’t have multiple entries therefore the Grammar is a LL(1).**