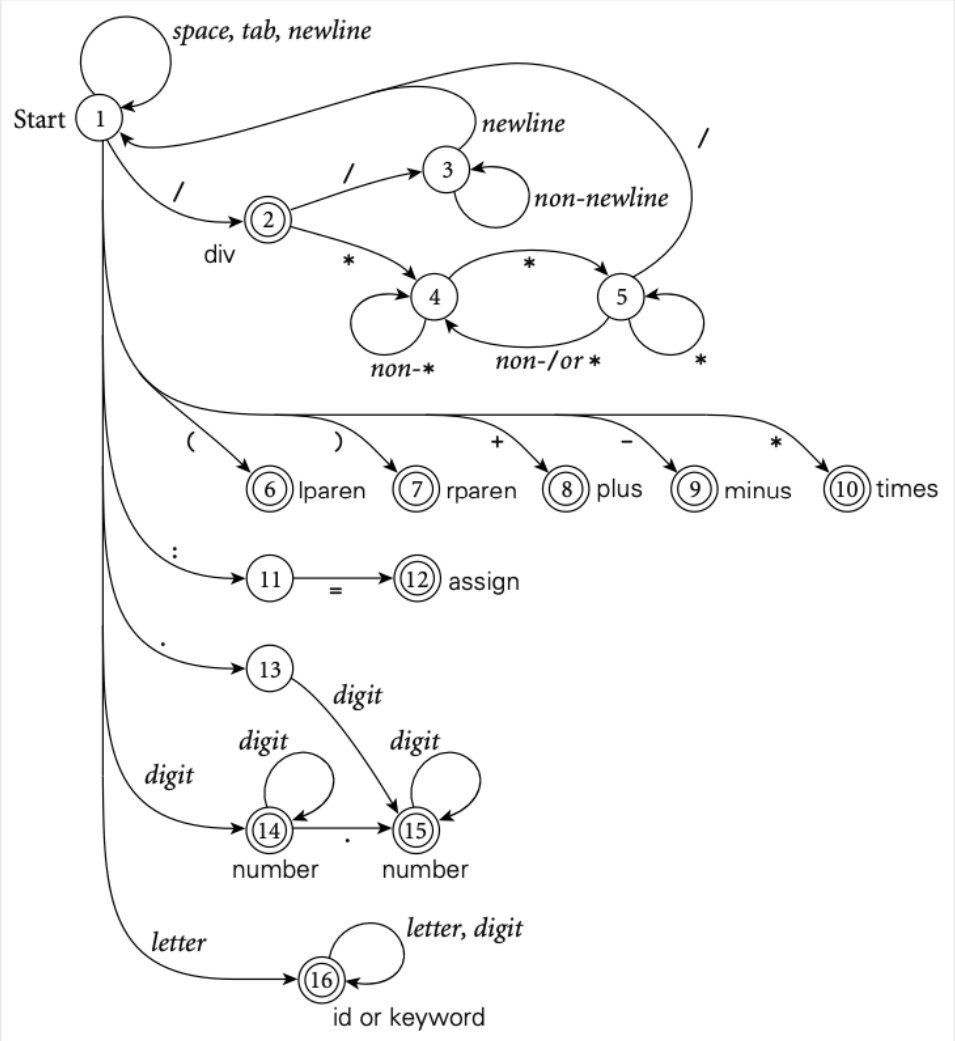


double circle make a token so 4b would be two tokens 4 and b

1. $E \longrightarrow T\ TT$
   - $\triangleright$ TT.st := T.val
   - $\triangleright$ E.val := TT.val
2. $TT_1 \longrightarrow$ `+` $T\ TT_2$
   - $\triangleright$ TT_2.st := TT_1.st + T.val
   - $\triangleright$ TT_1.val := TT_2.val
3. $TT_1 \longrightarrow$ `-` $T\ TT_2$
   - $\triangleright$ TT_2.st := TT_1.st − T.val
   - $\triangleright$ TT_1.val := TT_2.val
4. $TT \longrightarrow \epsilon$
   - $\triangleright$ TT.val := TT.st
5. $T \longrightarrow F\ FT$
   - $\triangleright$ FT.st := F.val
   - $\triangleright$ T.val := FT.val
6. $FT_1 \longrightarrow$ `*` $F\ FT_2$
   - $\triangleright$ FT_2.st := FT_1.st × F.val
   - $\triangleright$ FT_1.val := FT_2.val
7. $FT_1 \longrightarrow$ `/` $F\ FT_2$
   - $\triangleright$ FT_2.st := FT_1.st ÷ F.val
   - $\triangleright$ FT_1.val := FT_2.val
8. $FT \longrightarrow \epsilon$
   - $\triangleright$ FT.val := FT.st
9. $F_1 \longrightarrow$ `-` F_2
   - $\triangleright$ F_1.val := − F_2.val
10. $F \longrightarrow$ `(` E `)`
    - $\triangleright$ F.val := E.val
11. $F \longrightarrow$ `const`
    - $\triangleright$ F.val := const.val

EPS(a): if it products eps. then true
First(a): the first terminal of the productions
Follow(a): the first terminal comes after a
(derive non-terminals into terminals)
and if a is produced by non-terminal b then add Follow(b) to Follow(a)
Predict(a -> b): First(b)
if EPS(b) true then add Follow(a)

$$
\begin{array}{rcl}
Program & \rightarrow & Stmts\ \$\$ \\
Stmts & \rightarrow & Stmt\ Stmts\_Tail \\
Stmt & \rightarrow & \text{until}\ (Stmts, \text{id}) \mid \text{id} \\
Stmts\_Tail & \rightarrow & ;\ Stmt\ Stmts\_Tail \mid \epsilon
\end{array}
$$

short circuit evaluation check one thing that matters
non-short check everything

EPS(Program) false
EPS(Stmts) false
EPS(Stmt) false
EPS(Stmts_Tail) true

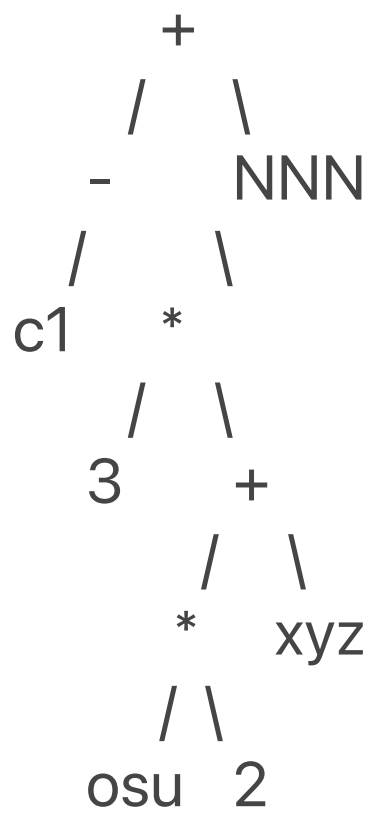
First(Program) until id
First(Stmts) until id
First(Stmt) until id
First(Stmts_Tail) ;

Follow(Program) empty set
Follow(Stmts) $$,
Follow(Stmt) ;
Follow(Stmts_Tail) $$,

Predict(Stmts_Tail -> eps) $$,
EPS(eps) is true

Scoping
keep the track of globally and locally declared variables
    static scoping
        no locally declared var, looks for global var
    dynamic scoping
        no locally declared var, looks for parent function var

draw a tree for
    c1-3 * (osu * 2 + xyz) + NNN



2. Suppose that we use the CFG and the attribute grammar in Figure [illegible] its decoration for the expression

```
10 * (8 - 4)
```