Ex: Top-down parsing

- **Write a recursive descent recognizer for this grammar:**

A → **int** | '(' **B** ')'
B → ~~[illegible]~~ '+' A B | int

```
type token = INT of int | LPAREN | RPAREN | PLUS | EOF

let rec parseA toklis = match (hd toklis) with
    INT i -> tl toklis
  | LPAREN -> let r = parseB (tl toklis)
              in if hd r = RPAREN then tl r else raise SyntaxErr
  | _ -> raise SyntaxError
and parseB toklis = match hd toklis with
    | INT i -> tl toklis
    | PLUS -> let r = parseA (tl toklis)
              in parseB r
and parse toklis = (parseA toklis = [EOF])
```

Top-down recognizer exercise

L $\rightarrow$ **E R**
E $\rightarrow$ **id**
R $\rightarrow$ **; L** | ϵ

```
type token = IDENT of string | SEMIC | EOF
let rec parseL toklis = parseR (parseE toklis)

and parseE toklis = match hd toklis with
                      IDENT _ -> tl toklis
                    | _       -> raise SyntaxError

and parseR toklis =
          if hd toklis = SEMIC
          then parseL (tl toklis)
          else toklis

and parse toklis = (parseL toklis = [EOF])
```

FIRST set examples

A $\rightarrow$ **id** | **'(' B ')'**
B $\rightarrow$ **int** | **A**

FIRST(A) = { id, (}

FIRST(B) = { int, id, (}

A $\rightarrow$ **int** | **'(' B ')'**
B $\rightarrow$ **A '+' B**

FIRST(A) = { int, (}

FIRST(B) = { int, (}

Ex: FIRST sets calculation

$E \rightarrow T \mid E + T$
$T \rightarrow P \mid T * P$
$P \rightarrow \textbf{id} \mid (\ E\ )$

$fsts_0$:

E	
T	
P	

$fsts_1$:

E	
T	
P	id, (

$fsts_2$:

E	
T	id, (
P	id, (

$fsts_3$:

E	id, (
T	id, (
P	id, (

$fsts_4$:

E	id, (
T	id, (
P	id, (

Ex: FIRST sets calculation

$$
\begin{array}{lcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & \epsilon \mid \mathbf{+}\ E \\
T & \rightarrow & P\ T' \\
T' & \rightarrow & \epsilon \mid \mathbf{*}\ T \\
P & \rightarrow & \mathbf{id} \mid \mathbf{(}\ E\ \mathbf{)}
\end{array}
$$

$fsts_0$:

E	
E'	
T	
T'	
P	

$fsts_1$:

E	
E'	•, +
T	
T'	•, *
P	id, (

$fsts_2$:

E	
E'	•, +
T	id, (
T'	•, *
P	id, (

$fsts_3$:

E	id, (
E'	•, +
T	id, (
T'	•, *
P	id, (

$fsts_4$:

E	id, (
E'	•, +
T	id, (
T'	•, *
P	id, (

Ex: FIRST sets calculation

$$
\begin{array}{lcl}
S & \rightarrow & A\ \mathbf{b} \\
A & \rightarrow & \mathbf{a} \mid B \mid \epsilon \\
B & \rightarrow & \mathbf{b} \mid \epsilon
\end{array}
$$

$fsts_0$:

S	
A	
B	

$fsts_1$:

s	
A	a, •
B	b, •

$fsts_2$:

S	a, b
A	a, b, •
B	b, •

$fsts_3$:

S	a, b
A	a, b, •
B	b, •

Ex: FOLLOW sets calculation

$$
\begin{array}{lcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & \epsilon \mid +\ E \\
T & \rightarrow & P\ T' \\
T' & \rightarrow & \epsilon \mid *\ T \\
P & \rightarrow & \textbf{id} \mid (\ E\ )
\end{array}
$$

$FIRST$:

E	**id, (**
E'	•, +
T	**id, (**
T'	•, +
P	**id, (**

$flws_0$:

E	eof
E'	
T	
T'	
P	

$flws_1$:

E	eof ,)
E'	eof
T	+ , eof
T'	
P	+ , eof

$flws_2$:

E	eof ,)
E'	eof ,)
T	+ , eof ,)
T'	+ , eof
P	+ , eof

$flws_3$:

E	eof ,)
E'	eof ,)
T	+ , eof ,)
T'	+ , eof ,)
P	+ , eof ,)

Ex: FOLLOW sets calculation

$$
\begin{array}{lcl}
S & \rightarrow & A\ B\ A \mid \mathbf{c}\ C \\
A & \rightarrow & \mathbf{a} \mid \epsilon \\
B & \rightarrow & \mathbf{b}\ D \mid \epsilon \\
C & \rightarrow & A\ D \mid \mathbf{b} \\
D & \rightarrow & \mathbf{a}\ A \mid \mathbf{c}
\end{array}
$$

$FIRST$:

S	**a, b, c, •**
A	**a, •**
B	**b, •**
C	**a, b, c**
D	**a, c**

$flws_0$:

S	eof
A	
B	
C	
D	

$flws_1$:

S	eof
A	b, a, c, eof
B	a, eof
C	eof
D	

$flws_2$:

S	eof
A	b, a, c, eof
B	a, eof
C	eof
D	a, eof

$flws_3$:

S	eof
A	b, a, c, eof
B	a, eof
C	eof
D	a, eof