SHIFT - REDUCE PARSER

* Bottom-up Parser where the **stack** Data structure holds the **'grammar' symbols** and **input buffer** holds the **rest of string** to be parsed.

* ACTIONS OF SHIFT-REDUCE PARSER:
  (1) **Shift**: Shift next input symbol to top of stack
  (2) **Reduce**: Reduce if possible using appropriate production Rule.
  (3) **Accept**: Succesful completion of Parsing
  (4) **Error**: Syntax Error.

$E \rightarrow E+T \mid T$ input → id*id
$T \rightarrow T*F \mid F$
$F \rightarrow (E) \mid id$

STACK	INPUT BUFFER	ACTION
$	id * id $	Shift
$id	*id $	Reduce F→id
$ F	*id $	Reduce T→F
$ T	*id $	Shift → If Reduced of (E→T) is performed, String cannot be Parsed.
$T*	id $	Shift
$T*id	$	Reduce F→id
$T*F	$	Reduce T→T*F
$ T	$	Reduce E→T
$E (Input)	$	ACCEPTED

$S \rightarrow (L) \mid a$ input = (a,(a,a))
$L \rightarrow L,S \mid S$

STACK	INPUT BUFFER	ACTION
$	(a,(a,a)) $	Shift
$(	a,(a,a)) $	Shift
$(a	,(a,a)) $	Reduce S→a
$(S	,(a,a)) $	Reduce L→S
$(L	,(a,a)) $	Shift
$(L,	(a,a)) $	Shift
$(L,(	a,a)) $	Shift
$(L,(a	,a)) $	Reduce S→a
$(L,(S	,a)) $	Reduce L→S
$(L,(L	,a)) $	Shift
$(L,(L,	a)) $	Shift
$(L,(L,a	)) $	Reduce S→a
$(L,(L,S	)) $	Reduce L,S→L
$(L,(L)		

Operator Precedence Parser.

★ It is a **Bottom-Up Parser** which parses the input string based on the Operator precedence Relation table.

★ Based on Operator Grammar:
- To define mathematical operations with some restrictions.
- **No 2 Variables** can be adjacent on R.H.S.
- **No 'ε' on RHS** of production.

#1- $E \to E+E \mid E*E \mid id$ ✓ Operator Grammar

#2- $E \to EAE \mid id$ ✗ → Can be Converted to Operator Grammar

$A \to +\mid *$

#3- $S \to SAS \mid a$ ✗ → Can be Converted to Operator Grammar

$A \to bSb \mid b$

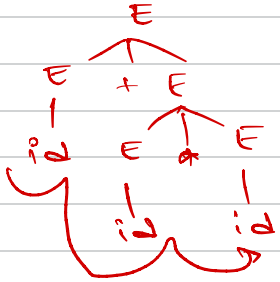
#1- $E \to E+E \mid E*E \mid id$ id+id*id

	+	*	id	$
+	>	<	<	>
*	>	>	<	>
id	>	>	—	>
$	<	<	<	Accept

Take all terminal Symbols.

id ⟩ a,b
$ <
stack ~ + > + → input
+ < *
* > *

STACK	RELATION	INPUT STRING	OPERATION
$	<	id+id*id $	Shift id
$id	>	+id*id $	Reduce
$E	<	+id*id $	Shift +
$E+	<	id*id $	Shift id
$E+id	>	*id $	Reduce
$E+E	<	*id $	Shift *
$E+E*	<	id $	Shift id
$E+E*id	>	$	Reduce id
$E+E*E	>	$	Reduce
$E+E	>	$	Reduce
$E	Accept	$	—



Leading & Trailing:

Leading (A):

* $A \rightarrow \alpha a \beta$ [$\alpha \rightarrow$ Single Variable or ε]
  Leading (A) = {a}

* $A \rightarrow B a \beta$
  Leading (A) = Leading (B)

Trailing (A)

* $A \rightarrow \alpha a \beta$
  Trailing (A) = {a}

* $A \rightarrow \alpha a B$
  Trailing (A) = Trailing (B)

LR- Parsing.

→ Left to Right Scanning

→ Rightmost Derivation

* Bottom-up Parsing Technique.

ADVANTAGES:

- Efficient non-backtracking shift-reduce parser method.
- Detects Syntactic error as soon as possible
- Recognises almost all programming languages.

DISADVANTAGES:

- Too much work to construct LR Parser by hand.

① SLR – Simple LR
② CLR – Canonical LR
③ LALR – LookAhead LR

} Works the same, only their parsing table is different.