CS & IT Engineering

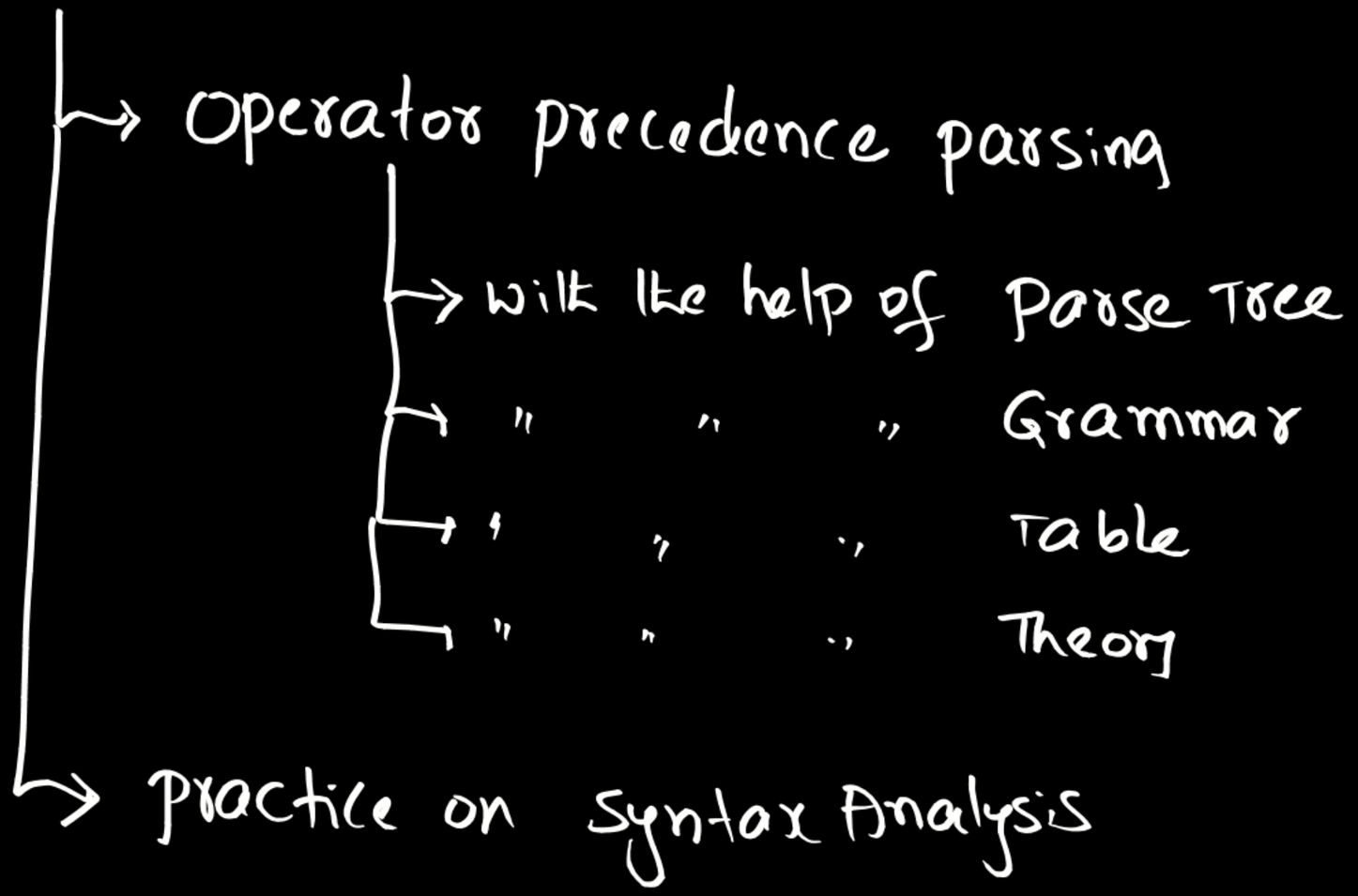
Compiler Design

Lexical Analysis and Syntax Analysis

Lecture : 11

Deva sir

Topics to be covered:



What is Precedence Relation ?

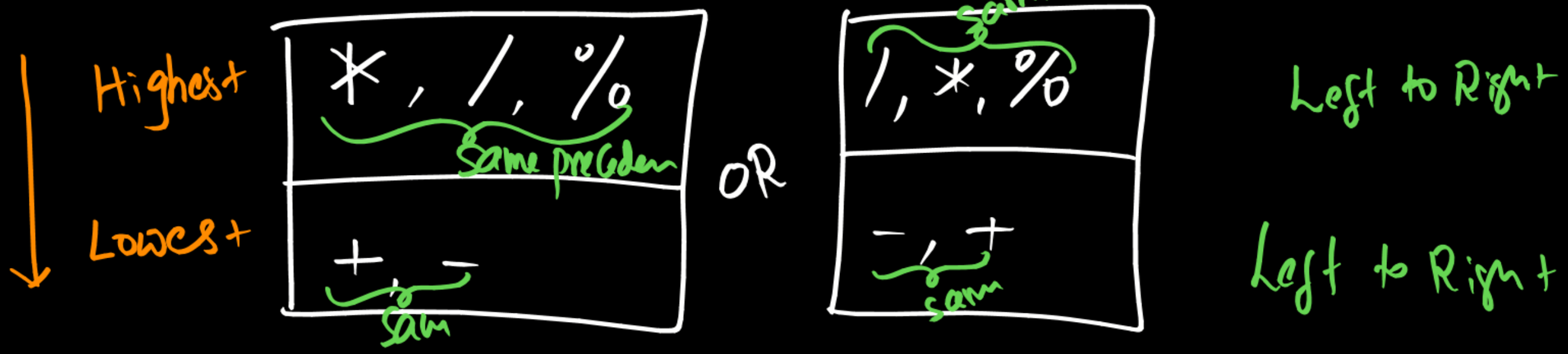
I) Highest

II) Lowest

III) Equal ⟹ L to R Associativity
R to L Associativity

a + b * c

a * b + c



Arithmetic operators precedence Table

a % b * c

a + b - c

a - b + c

Precedence Relations:

a + b + c

(I) a + b * c

Left Right

S → A + B

A → a

B → b * c

(II) a * b + c

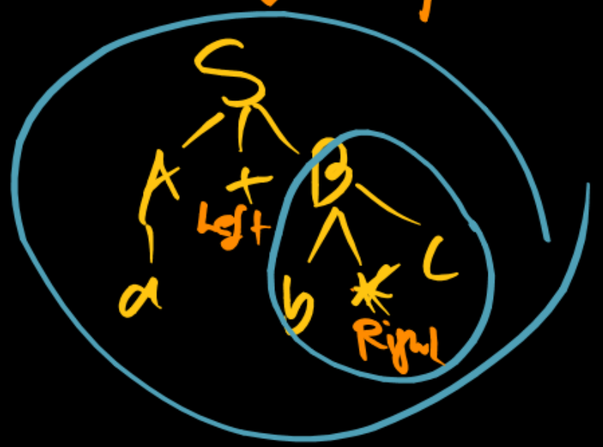
Left Right

(this is impossible using CFG)

Left (stack)

Right (Input)

	+	*	$
+	—	<	
*	—	—	—



a) + is Highest

b) * is Highest

c) + is lowest

d) * is lowest

Left < Right

Left > Right

Left = Right

a + b * c

Left Right

a + b + c

Left Right

a * b + c

Left Right

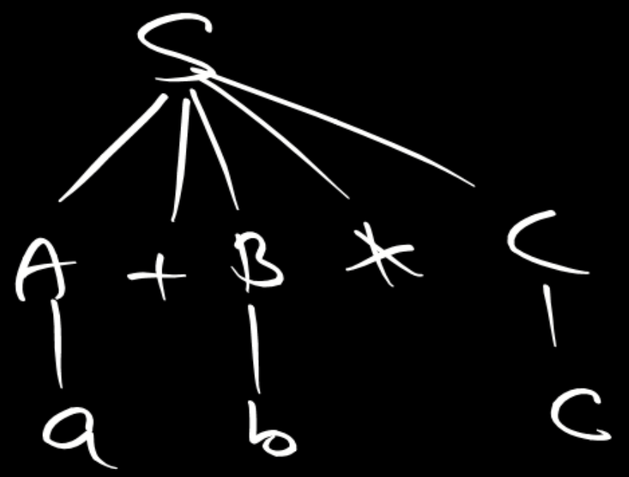
a * b * c

Left Right

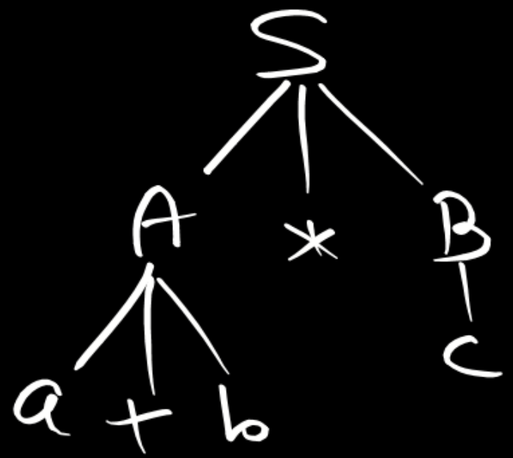
a + b * c

Left

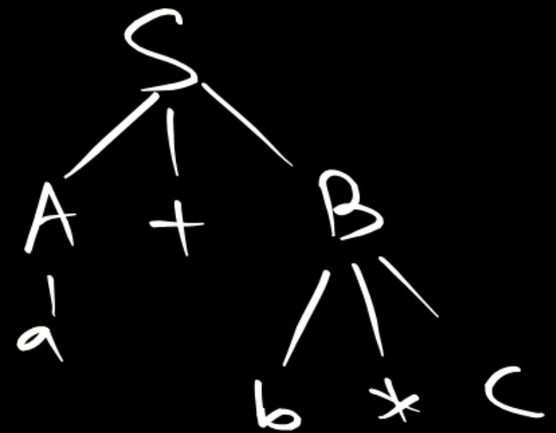
Right



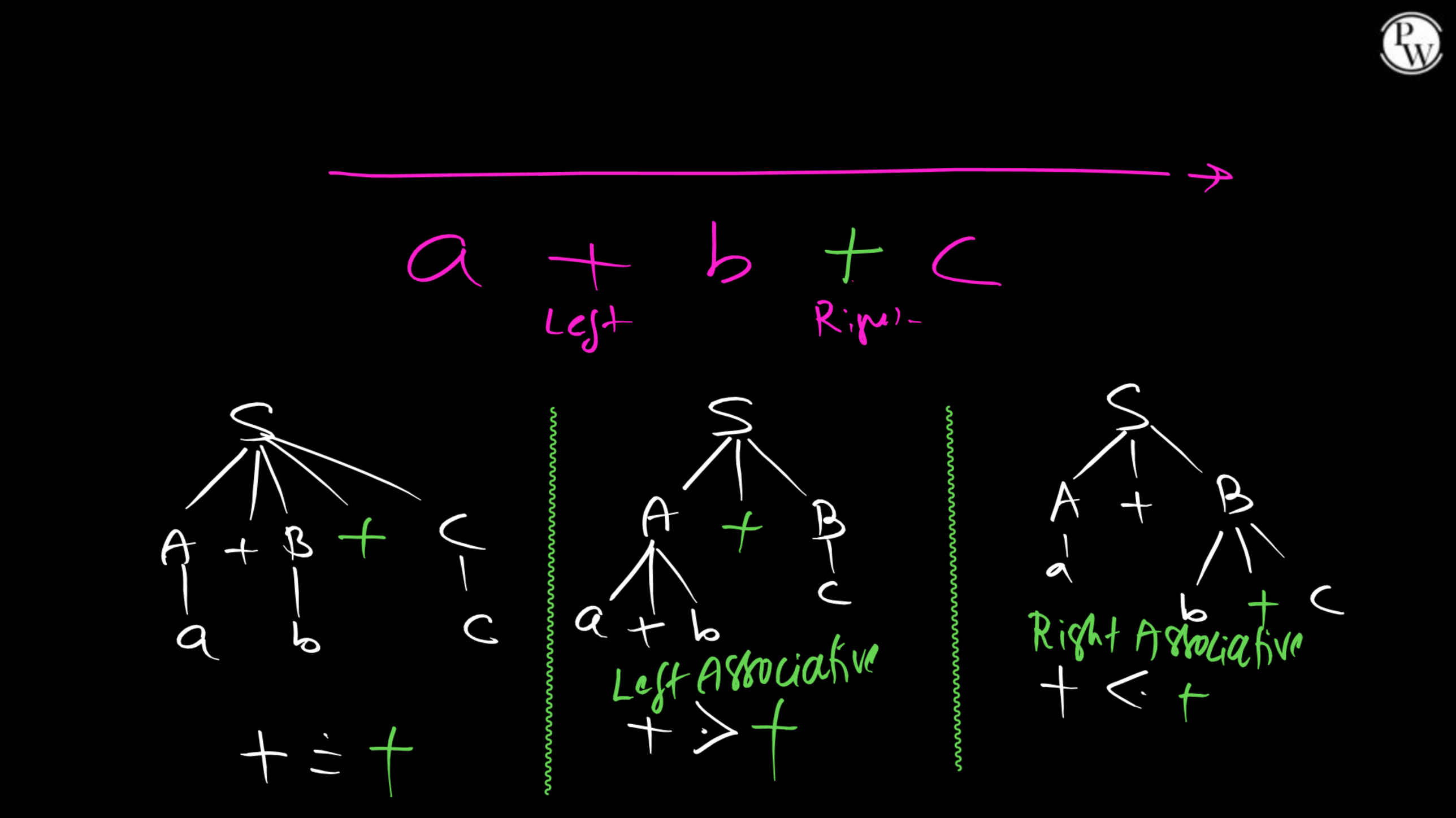
+ = *



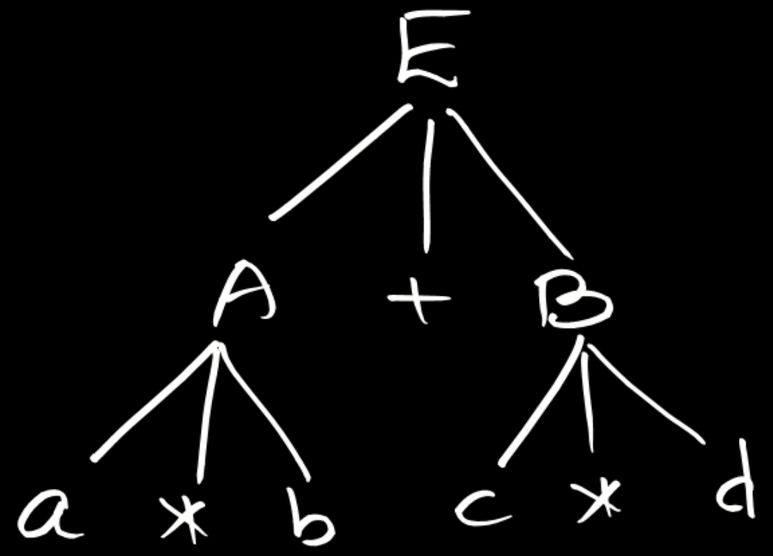
+ > *



+ < *


a + b + c
Left
Right
S
A + B + C
a
b
c
+ = +
S
A + B
a + b
c
Left Associative
+ > +
S
A + B
a
b + c
Right Associative
+ < +

① Find Precedence relations using Parse Tree.



Q1) Find correct.

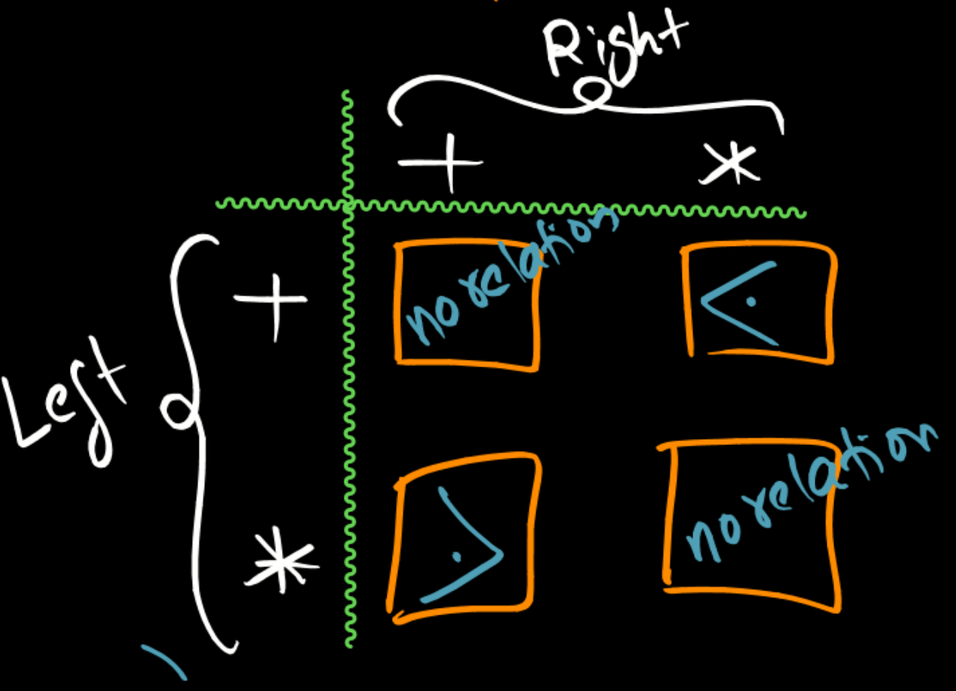
A. + is lowest

B. * is higher

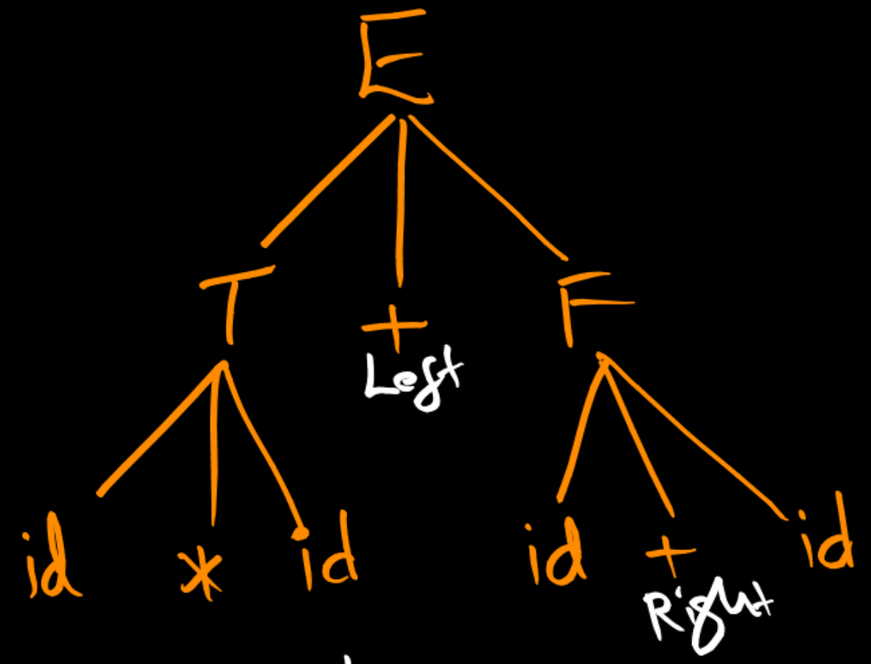
C. + is higher

D. None

Q2) Find correct operator precedence Table



(2) Find precedence relations using following Parse Tree.



Q1) Choose correct.

A) + is lowest than *

B) * is highest than +

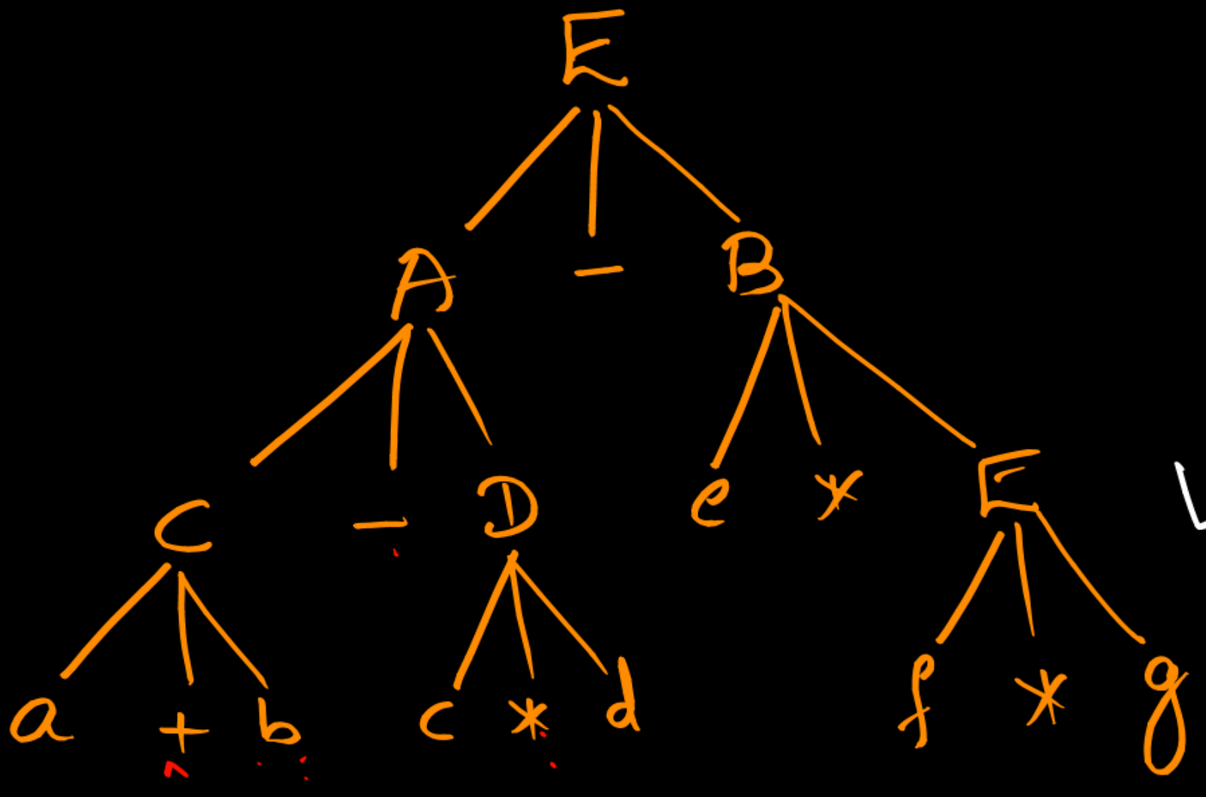
C) + is Left Associative

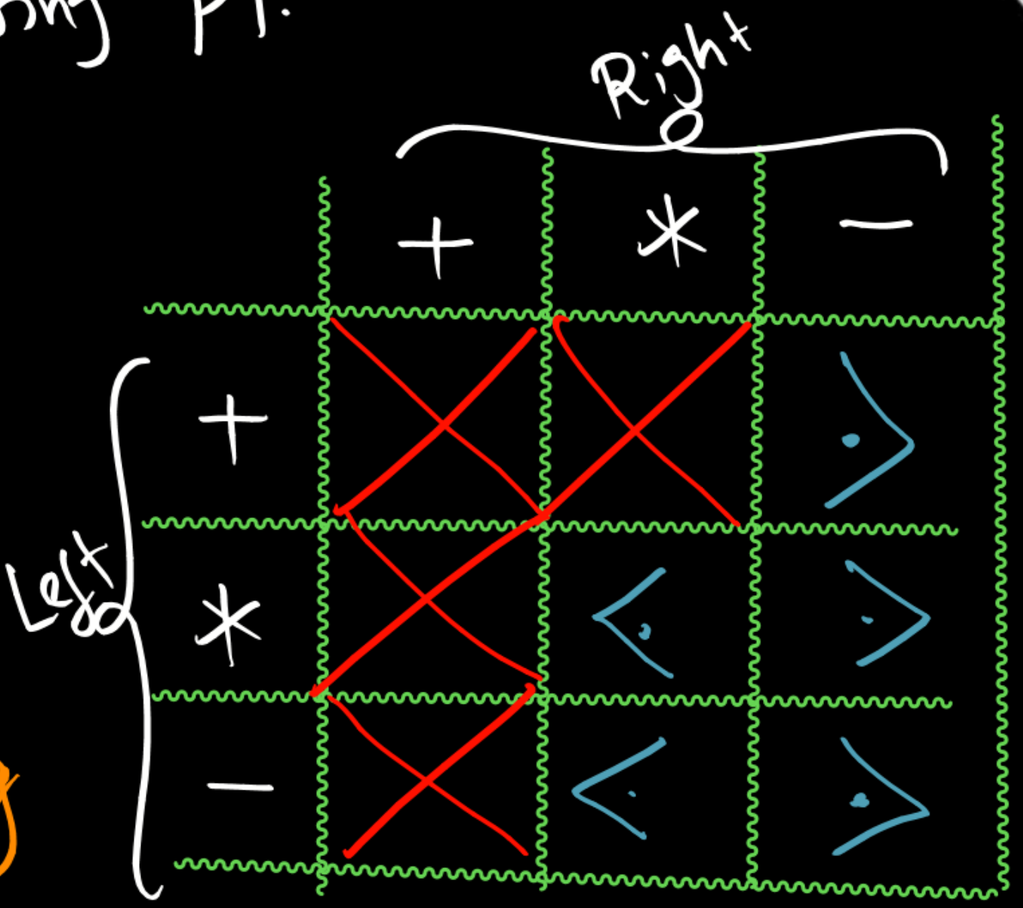
D) + is Right Associative

Q2) Fill all entries of Table.

	Right: +	Right: *
Left: +	<	no relation
Left: *	>	no relation

③ Find precedence Relations using PT.

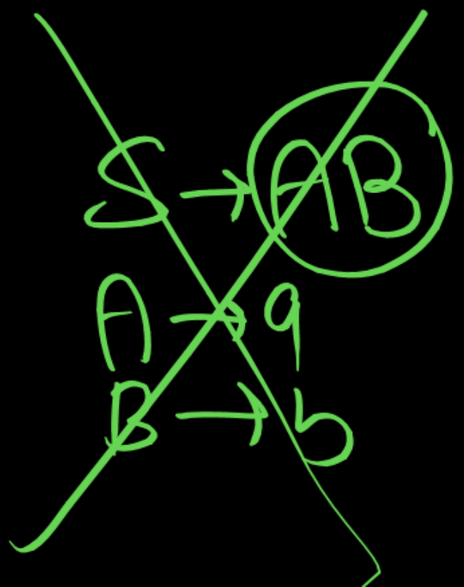




Operator Grammar

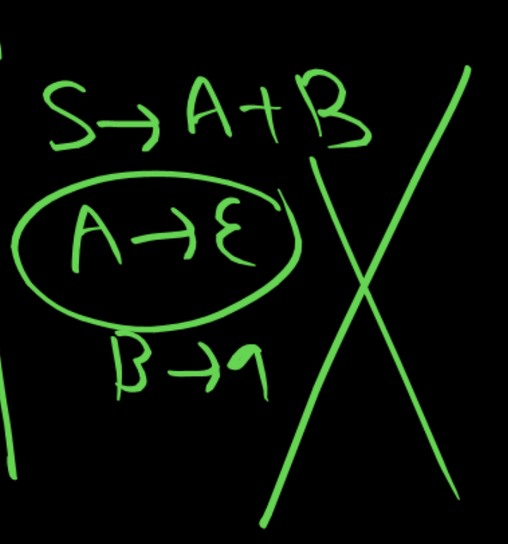
- It may be ambiguous or unambiguous
- It is CFG which do not contain

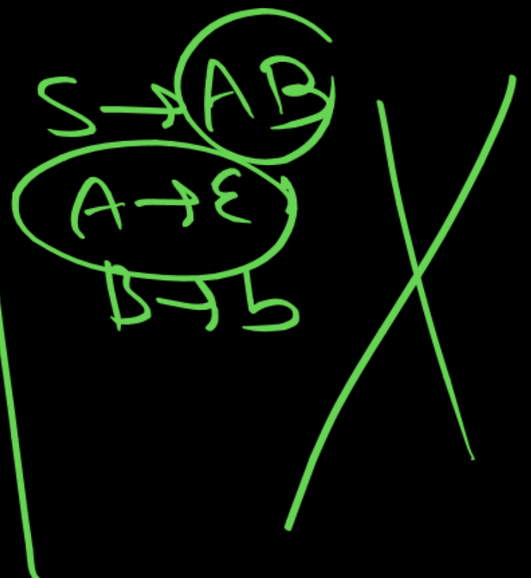
Consecutive non-terminals and also should not have **null rules**.



It is operator Grammar

S → A + B
A → a
B → b ✓





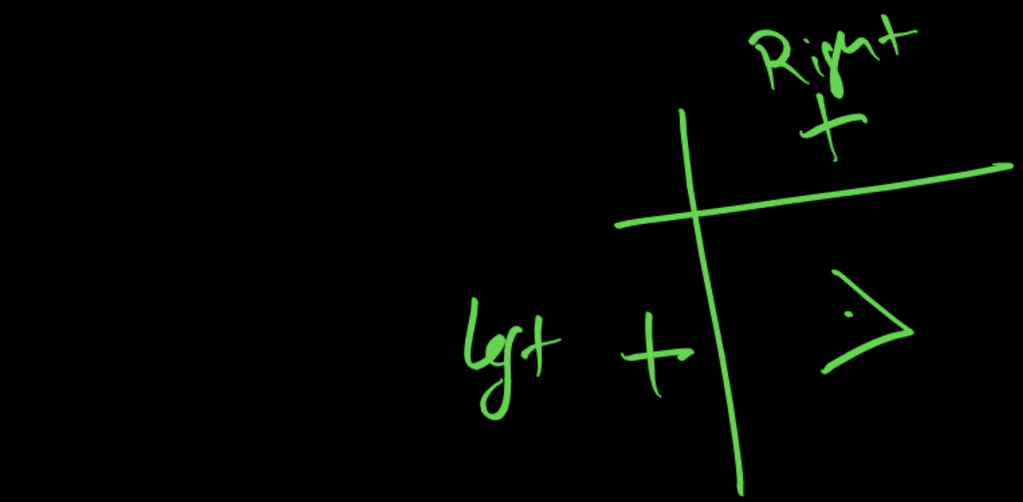
④ Find Precedence Relations using CFG.

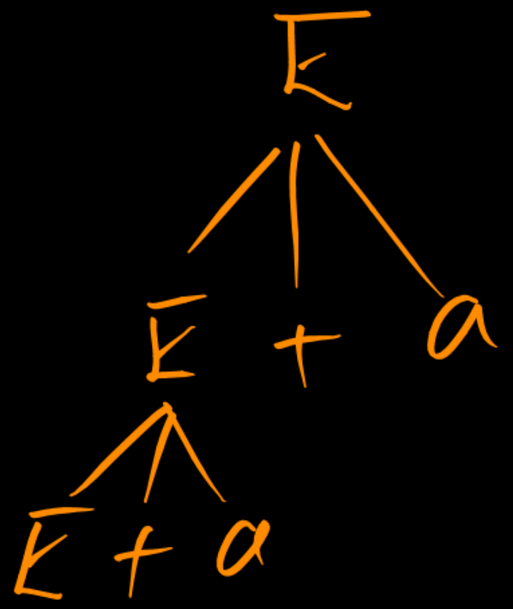
$E \rightarrow E + a \mid a$

Left Recursion

$a + a + a$

+ is Left Associative



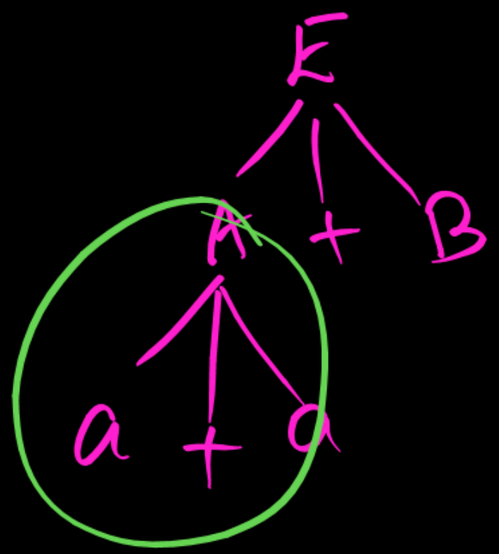


(5)

$E \rightarrow A + B$

$A \rightarrow a + a$

$B \rightarrow a$



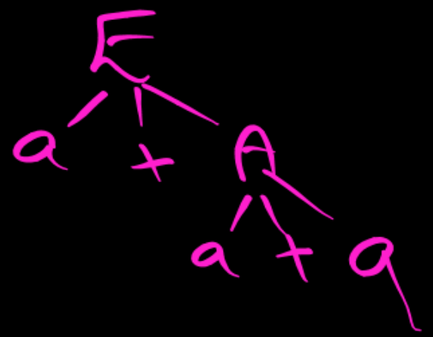
+ is left Associative

⑥ $E \rightarrow a+E \mid a$

+ is Right Associative

⑦ $E \rightarrow a+A$

$A \rightarrow a+a$



+ is Right Associative

(8)

$$E \rightarrow E + T \mid a$$

$$T \rightarrow F * T \mid b$$

$$F \rightarrow c$$

A) * is highest

B) + is lowest

C) + is Left Associative

D) * is Right Associative

(9) $E \rightarrow E+E \mid E-E \mid id$

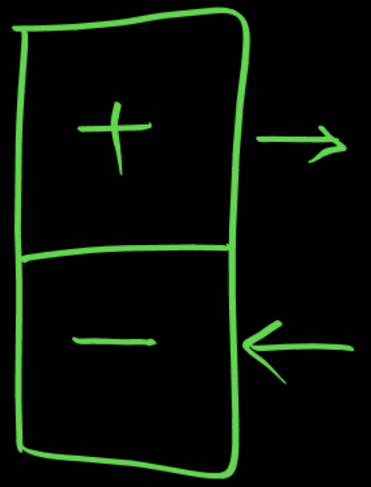
and

+ is highest

− is lowest

+ is left to right associative

− is right to left associative



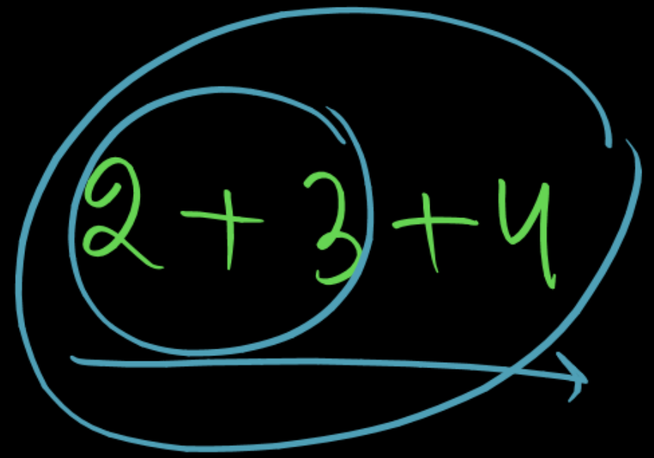
Find output for $2+3-1-5+6-1$

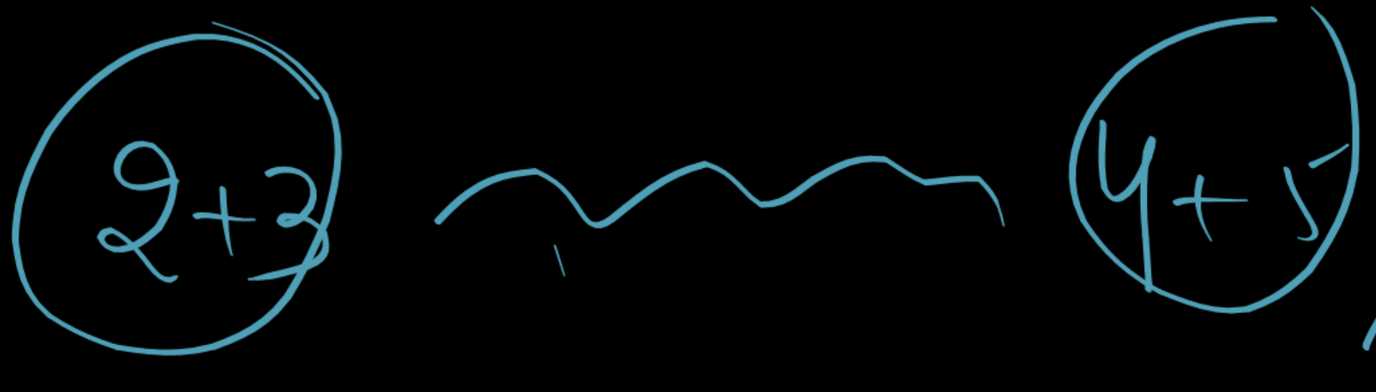
$5-1-11-1$

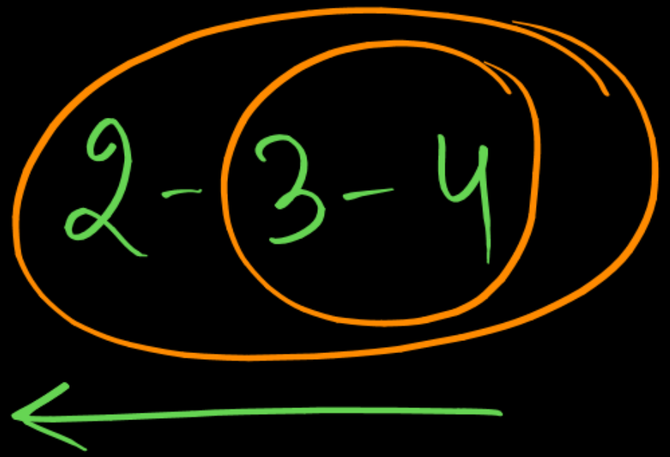
$11-1 = 10$

$1-10 = -9$

$5-(-9) = 14$


2+3+4


2+3
4+5
Independent



$2 - (-3) - 4$

MCQ

Which one of the following statements is TRUE?

[GATE-2022-CS: 1M]

A. The LALR(1) parser for a grammar G cannot have reduce – reduce conflict if the LR(1) parser for G does not have reduce – reduce conflict.

B. Symbol table is accessed only during the lexical analysis phase.

C. Data flow analysis is necessary for run-time memory management.

D. LR(1) parsing is sufficient for deterministic context – free languages.

DCFL $\cong$ LR(1) CFG $\cong$ LR(1) language

I) For every DCFL, LR(1) CFG exist

II) Every LR(1) generates DCFL.

Set of all DCFLs

=

Set of all languages generated by LR(K) CFGs
$K \geq 1$

NAT

Consider the augmented grammar with {+,* , (,), id} as the set of terminals.

$S' \rightarrow S$

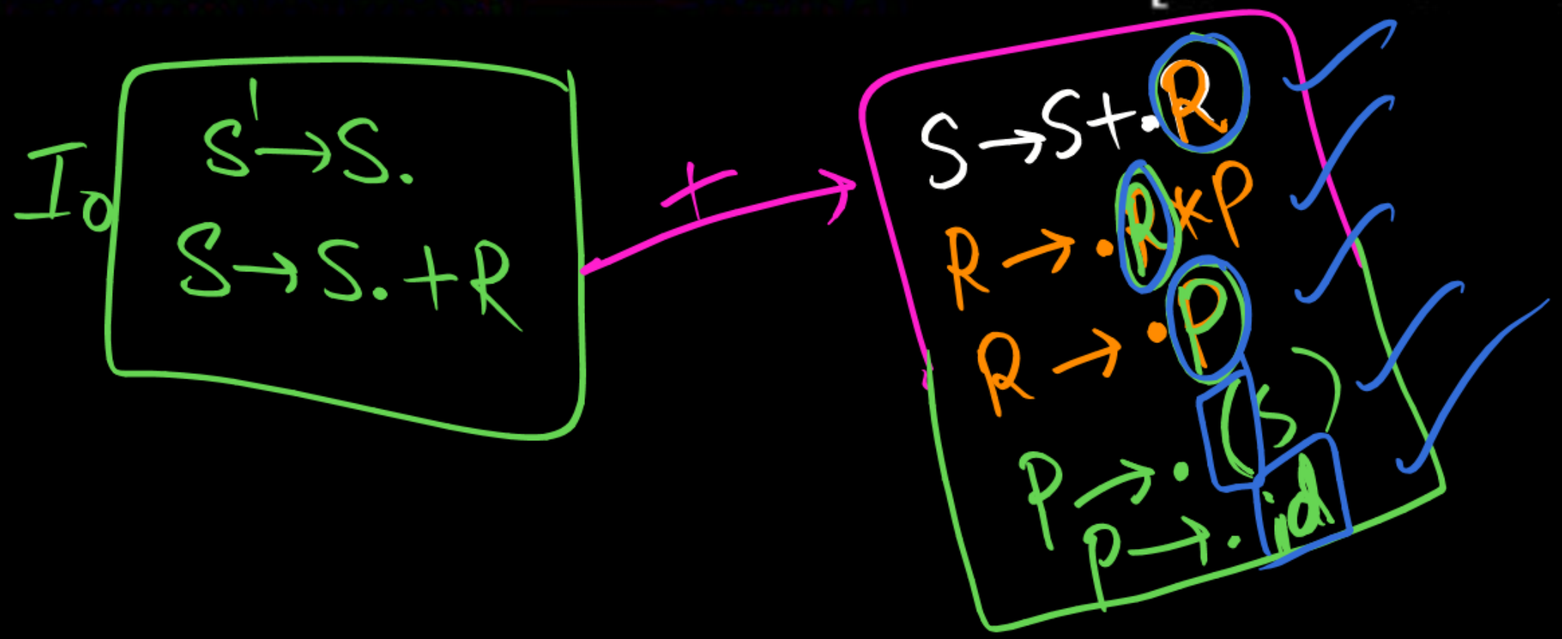
$S \rightarrow S + R \mid R$

$R \rightarrow R * P \mid P$

$P \rightarrow (S) \mid id$

If I_0 is the set of two $LR(0)$ items $\{[S' \rightarrow S.\,], [S \rightarrow S.\,+R]\}$, then $goto(closure(I_0), +)$ contains exactly $\underline{\quad 5 \quad}$ items.

[GATE-2022-CS: 1M]



MCQ

Consider the following statements

S_1: Every SLR (1) grammar is unambiguous but there are certain unambiguous grammars that are not SLR(1).

S_2: For any context-free grammar, there is a parser that takes at most $O(n^3)$ time to parse a string of length n.

Which one of the following options is correct? [GATE-2021-CS: 1M]

A. S_1 is true and S_2 is false

B. S_1 is false and S_2 is true

C. S_1 is true and S_2 is true

D. S_1 is false and S_2 is false

CYK Algorithm

- Bottom up Parsing
- Dynamic programming
- $O(n^3)$ Algorithm
- Membership Algo for CFG.
- It verifies given string generated given CFG or not.

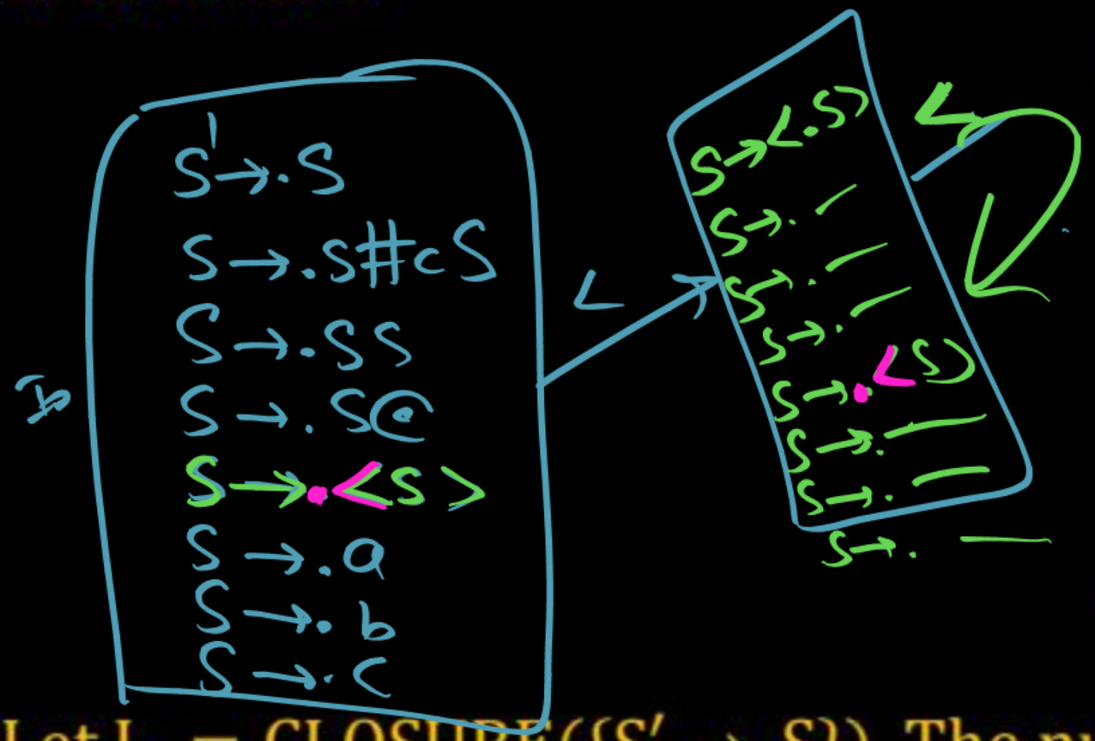
CFG:

String = abc


all Unamb CFGs
all CLR CFGs
all LALR CFGs
all SLRs
all LR(0) CFGs

NAT

Consider the following augmented grammar with {#, @, <, >, a, b, c} as the set of terminals



$S' \rightarrow S$
$S \rightarrow S \# cS$
$S \rightarrow SS$
$S \rightarrow S@$
$S \rightarrow < S >$
$S \rightarrow a$
$S \rightarrow b$
$S \rightarrow c$

Let I_0 = CLOSURE({$S' \rightarrow .S$}). The number of items in the set GOTO(GOTO(I_0, <), <) is _____

[GATE-2021-CS: 2M]

NAT

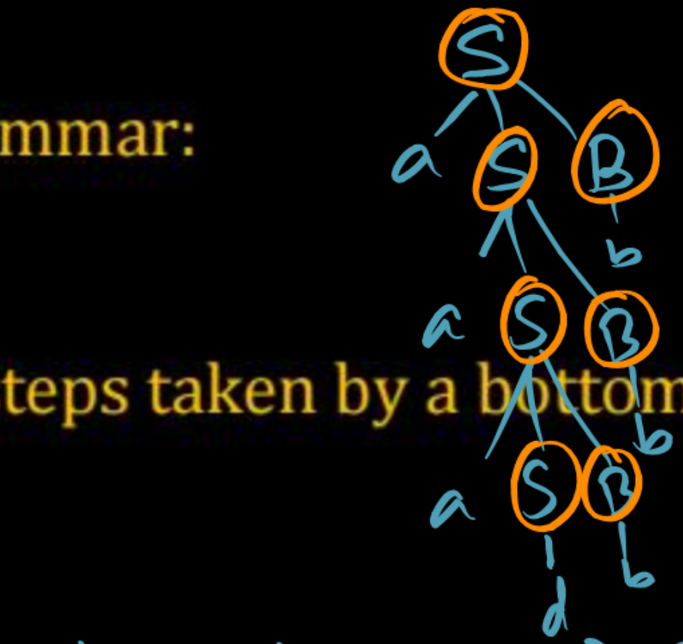
Consider the following grammar:

$S \rightarrow aSB|d$

$B \rightarrow b$

The number of reduction steps taken by a bottom-up parser while accepting the string *aaadbbb* is _____.

[GATE-2021-CS: 2M]

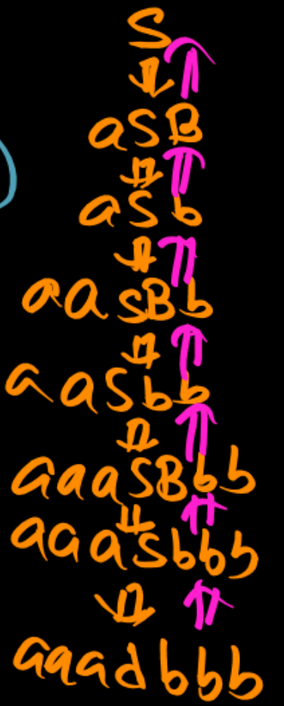


= 7 //

No. of reduction steps in BUP (reverse of RMD)

=

No. of steps in RMD



MCQ

Which one of the following kinds of derivation is used by LR parsers?

[GATE-2019-CS: 1M]

A. Leftmost

B. Leftmost in reverse

C. Rightmost

D. Rightmost in reverse

MCQ

Which of the following statements about parser is/are CORRECT?

I. Canonical LR is more powerful than SLR.

II. SLR is more powerful than LALR.

III. SLR is more powerful than Canonical LR.

[GATE-2017-CS: 1M]

A

B I only

C II only

D III only

II and III only

MCQ

Consider the following grammar G.

$S \rightarrow F \mid H$

$F \rightarrow p \mid c$

$H \rightarrow d \mid c$

Where S, F and H are non-terminal symbols, p, d and c are terminal symbols.

Which of the following statements(s) is/are correct?

S1: LL(1) can parse all strings that are generated using grammar G.

S2: LR(1) can parse all strings that are generated using grammar G.

[GATE-2015-CS: 2M]

A. Only S1

B. Only S2

C. Both S1 and S2

D. Neither S1 and S2

MCQ

Among simple LR (SLR), canonical LR, and look-ahead LR (LALR), which of the following pairs identify the method that is very easy to implement and the method that is the most powerful, in that order?

[GATE-2015-CS: 1M]

A. SLR, LALR

B. Canonical LR, LALR

C. SLR, canonical LR

D. LALR, canonical LR

MCQ

Which one of the following is TRUE at any valid state in shift-reduce parsing?

[GATE-2015-CS: 1M]

A. Viable prefixes appear only at the bottom of the stack and not inside

B. Viable prefixes appear only at the top of the stack and not inside

C. The stack contains only a set of viable prefixes

D. The stack never contains viable prefixes

MCQ

Consider the grammar defined by the following production rules, with two operators * and +

$S \rightarrow T * P$

$T \rightarrow U \mid T * U$

$P \rightarrow Q + P \mid Q$

$Q \rightarrow Id$

$U \rightarrow Id$

Which one of the following is TRUE? [GATE-2014-CS: 1M]

A. + is left associative, while * is right associative

B. + is right associative, while * is left associative

C. Both + and * are right associative

D. Both + and * are left associative

MCQ

A canonical set of items is given below

$S \rightarrow L. > R$

$Q \rightarrow R.$

On input symbol > the set has

[GATE-2014-CS: 1M]

(A) a shift-reduce conflict and a reduce-reduce conflict.

(B) a shift-reduce conflict but not a reduce-reduce conflict.

(C) a reduce-reduce conflict but not a shift-reduce conflict.

(D) neither a shift-reduce nor a reduce-reduce conflict.

MCQ

Consider the following two sets of LR(1) items of LR(1) grammar.

X → c.X,c/d	X→c.X,$
X → .cX,c/d	X→.cX,$
X → .d,c/d	X→.d,$

Which of the following statement related to merging of the two sets in the corresponding LALR parser is/are FALSE?

1. Cannot be merged since look aheads are different.
2. Can be merged but will result in S-R conflict.
3. Can be merged but will result in R-R conflict.
4. Cannot be merged since goto on c will lead to two different sets.

[GATE-2015-CS: 2M]

(A) 1 only
(B) 2 only
(C) 1 and 4 only
(D) 1, 2, 3 and 4

MCQ

What is the maximum number of reduces moves that can be taken by a bottom up parser for a grammar with no epsilon and unit productions (i.e. of type $A \rightarrow \varepsilon$ and $A \rightarrow a$) to parse a string with n tokens?

[GATE-2013-CS: 1M]

A) n/2

B) n-1

C) 2n-1

D) 2^n

MCQ

The grammar $S \rightarrow aSa \mid bS \mid c$ is

[GATE-2010-CS: 2M]

A. LL(1) but not LR(1)

B. LR(1) but not LL(1)

C. Both LL(1) and LR(1)

D. Neither LL(1) nor LR(1)

MCQ

An LALR(1) parser for a grammar G can have shift-reduce (S-R) conflicts if and only if

[GATE-2008-CS: 2M]

A. The SLR(1) parser for G has S-R conflicts

B. The LR(1) parser for G has S-R conflicts

C. The LR(0) parser for G has S-R conflicts

D. The LALR(1) parser for G has reduce-reduce conflicts.

MCQ

Which of the following describes a handle (as applicable to LR-parsing) appropriately?

[GATE-2008-CS: 1M]

A. It is the position in a sentential form where the next shift or reduce operation will occur.

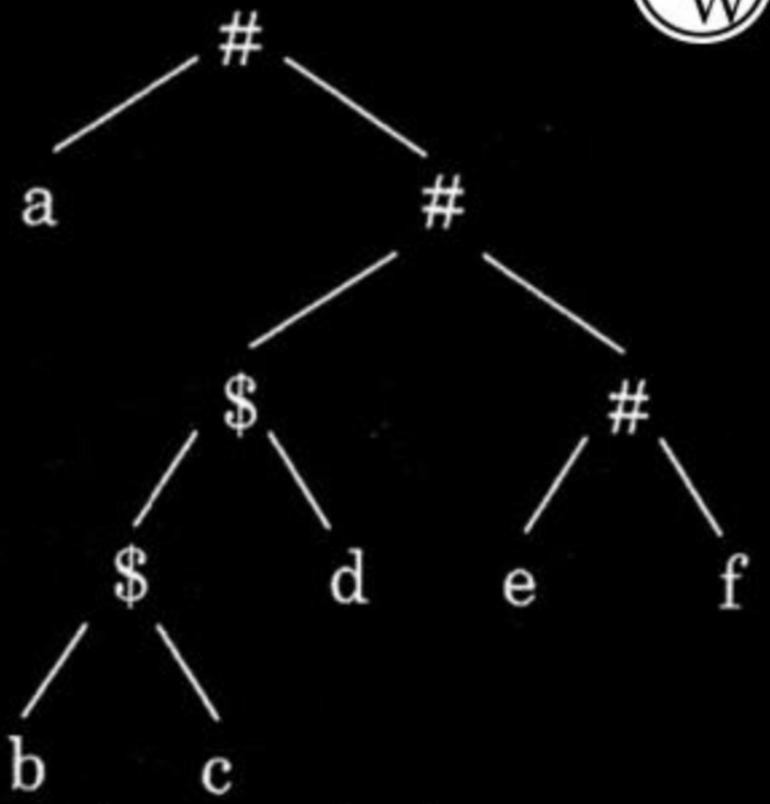
B. It is non-terminal whose production will be used for reduction in the next step

C. It is a production that may be used for reduction in a future step along with a position in the sentential form where the next shift or reduce operation will occur.

D. It is the production p that will be used for reduction in the next step along with a position in the sentential form where the right-hand side of the production may be found.

MCQ

Consider the following parse tree for the expression
$a \# b \$ c \$ d \# e \# f$,
involving two binary operators $ and #.
Which one of the following is correct for
the given parse tree? [GATE-2018-CS: 2M]



(A) *$ has higher precedence and is left associative; # is right associative*

(B) *# has higher precedence and is left associative; $ is right associative*

(C) *$ has higher precedence and is left associative; # is left associative*

(D) *# has higher precedence and is right associative; $ is left associative.*

MCQ

Which one of the following statements is FALSE? [GATE-2018-CS: 1M]

A. Context-free grammar can be used to specify both lexical and syntax rules.

B. Type checking is done before parsing.

C. High-level language programs can be translated to different Intermediate Representations.

D. Arguments to a function can be passed using the program stack

NAT

The attribute of three arithmetic operators in some programming language are given below.

OPERATOR	PRECEDENCE	ASSOCIATIVITY	ARITY
+	High	Left	Binary
-	Medium	Right	Binary
*	Low	Left	Binary

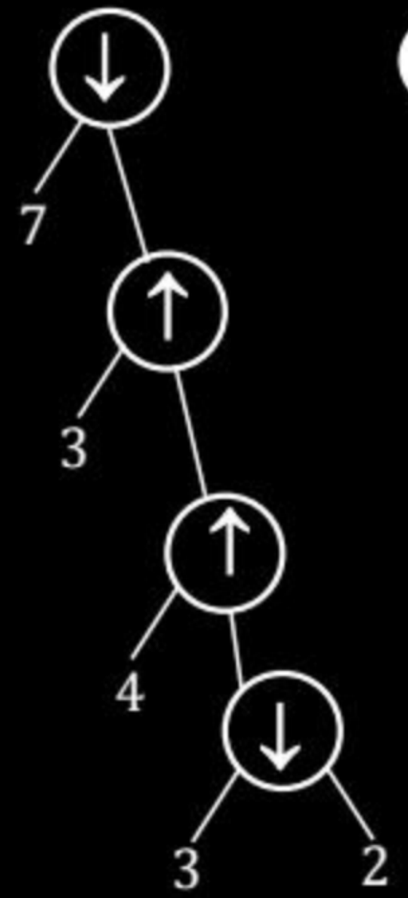
The value of the expression *2 – 5 + 1 – 7 * 3* in this language is _____.

[GATE-2016-CS: 2M]

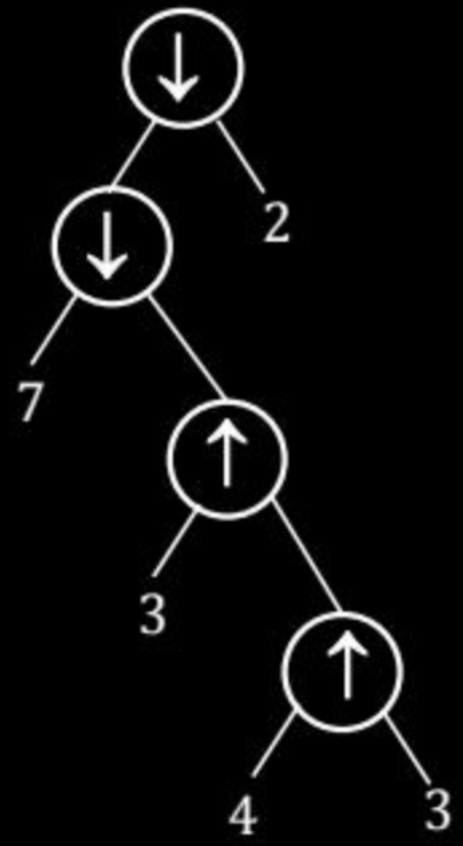
MCQ

Consider two binary operators '↑' and '↓' with the precedence of operator ↓ being lower than that of the operator ↑. Operator ↑ is right associative while operator ↓ is left associative. Which one of the following represents the parse tree for expression (*7 ↓ 3 ↑ 4 ↑ 3 ↓ 2*) [GATE-2011-CS: 2M]

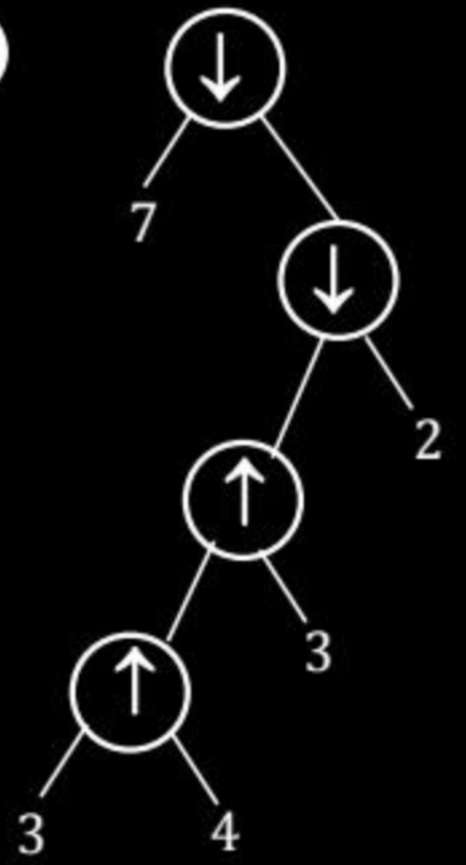
A



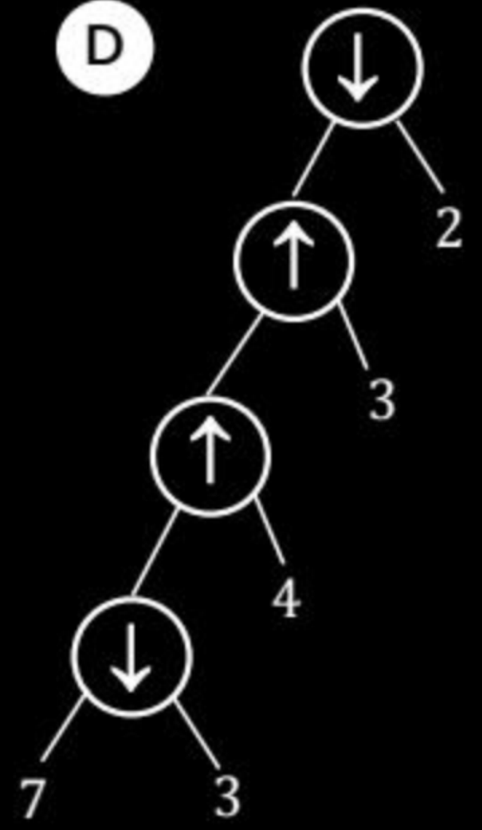
B



C



D



MCQ

A student wrote two context-free grammars G1 and G2 for generating a single C-like array declaration. The dimension of the array is at least one. For example,

int a[10][3];

The grammars use D as the start symbol, and use six terminal symbols int; id [ ] num.

Grammar G1

D → int L;
L → id [E
E → num]
E → num][E

Grammar G2

D → int L;
L → id E
E → E[num]
E → [num]

Which of the grammars correctly generate the declaration mentioned above?

(A) Both G1 and G2
(B) Only G1
(C) Only G2
(D) Neither G1 nor G2

[GATE-2016-CS: 2M]

Summary

↳ Syntax Analysis ✓

Next : SDTs


Thank you
PW
Soldiers