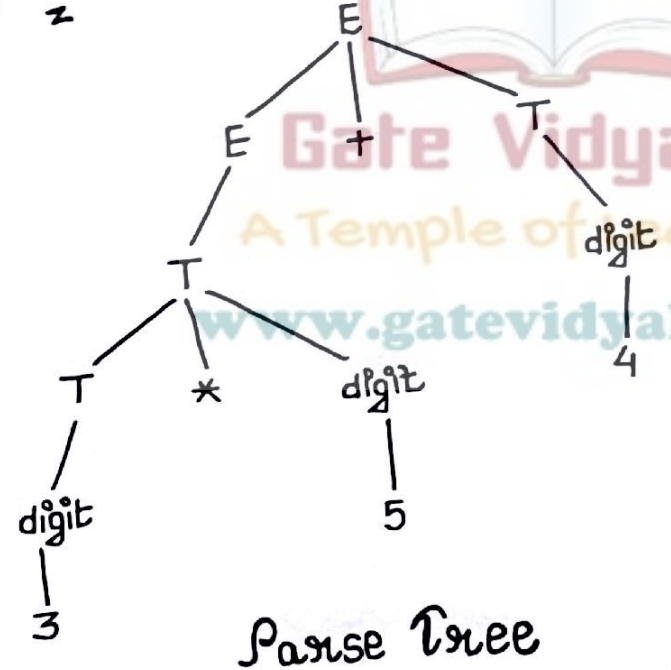
# Syntax Tree

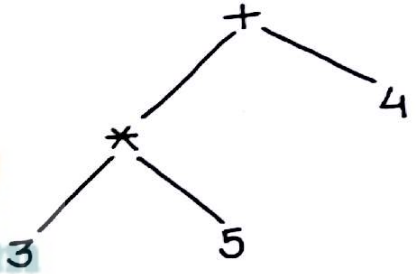
# Definition

A syntax tree (also known as abstract syntax tree) is a condensed form of parse tree.

Example:




Parse Tree




Syntax Tree

# Parse Tree Vs Syntax Tree

| Parse Tree | Syntax Tree |
| --- | --- |
| A parse tree is a graphical representation of the replacement process in a derivation. | A syntax tree is a condensed form of parse tree. |
| In parse trees,<br>→ each interior node represents a grammar rule<br>→ each leaf node represents a terminal | In syntax trees,<br>→ each interior node represents an operator<br>→ each leaf node represents an operand |

Parse trees represent every detail from the real syntax

Syntax trees does not represent every detail from the real syntax (that's why they are called abstract). For example- no rule nodes, no parentheses etc.

Parse trees are less dense compared to syntax trees for the same language construct

Syntax trees are more dense compared to parse trees for the same language construct.



xxxxxxx

## Problem-01 :

Consider the following grammar-

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

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

$$F \rightarrow (E) \mid id$$

For the string,

$$w = id + id * id$$



Generate-

i) Parse tree

ii) Syntax tree

iii) DAG

**Solution:-** The given grammar is-

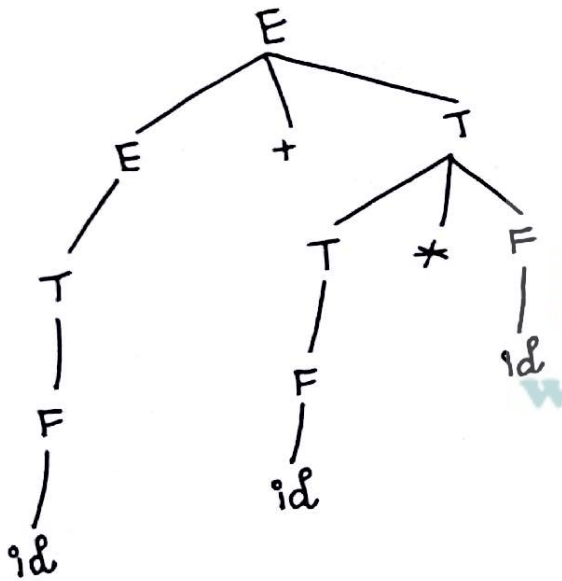
$E \rightarrow E+T / T$

$T \rightarrow T*F / F$

$F \rightarrow (E) / id$

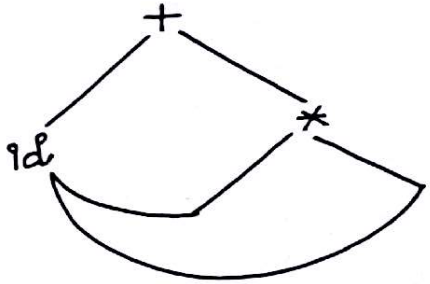
$W = id + id * id$

Parse tree




Syntax tree


DAG






xxxxxxxx

**Problem-02 :-** Construct the syntax tree for the following expression -

$$(a+b) * (c-d) + ((e/f) * (a+b))$$

**Solution :-**

**Step-1 :-** Convert the given expression into a postfix expression :-

$(a+b) * (c-d) + ((e/f) * (a+b))$

→ $ab+ * (c-d) + ((e/f) * (a+b))$

→ $ab+ * cd- + ((e/f) * (a+b))$

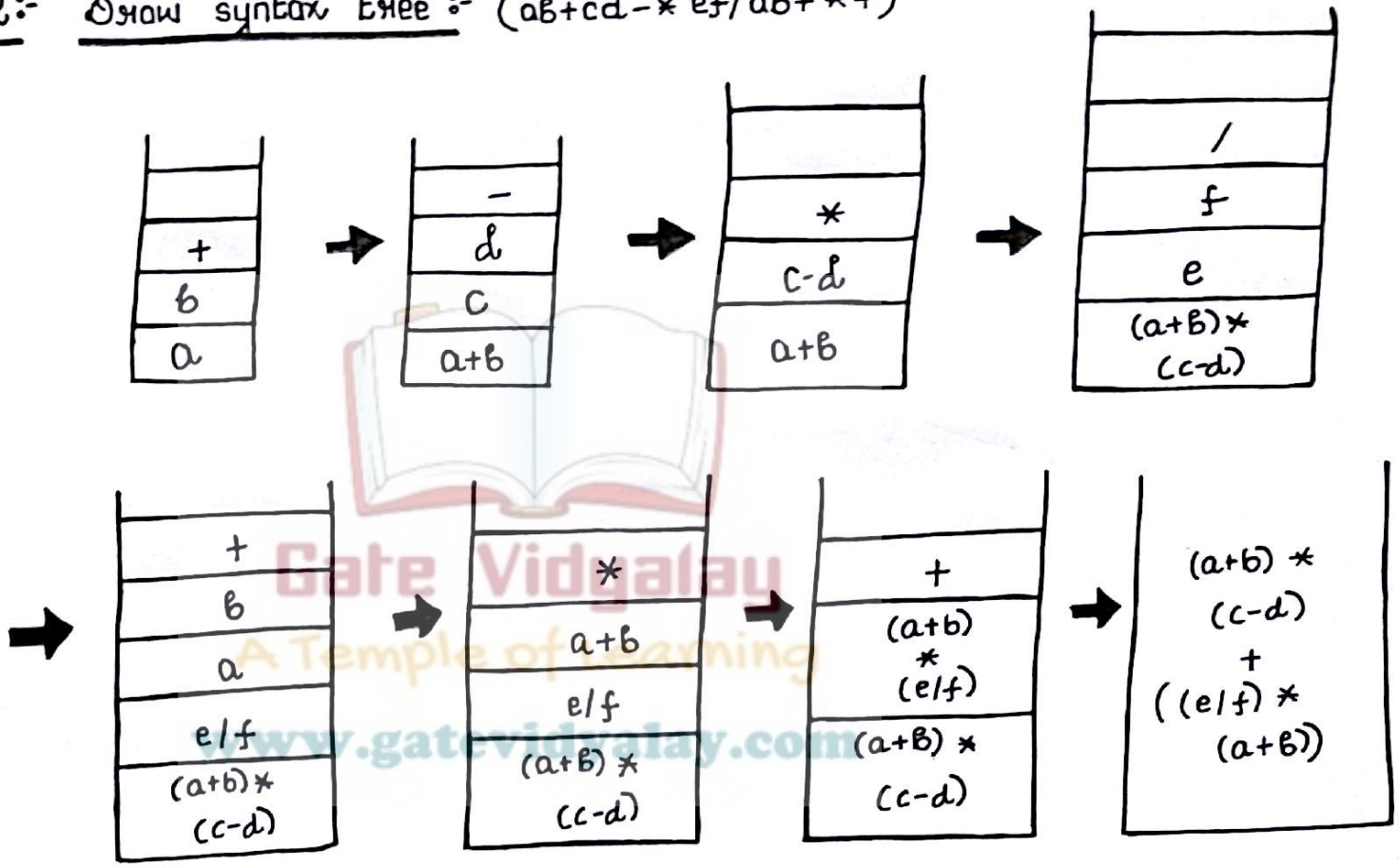
→ $ab+ * cd- + (ef/ * ab+)$

→ $ab+ * cd- + ef/ab+*$

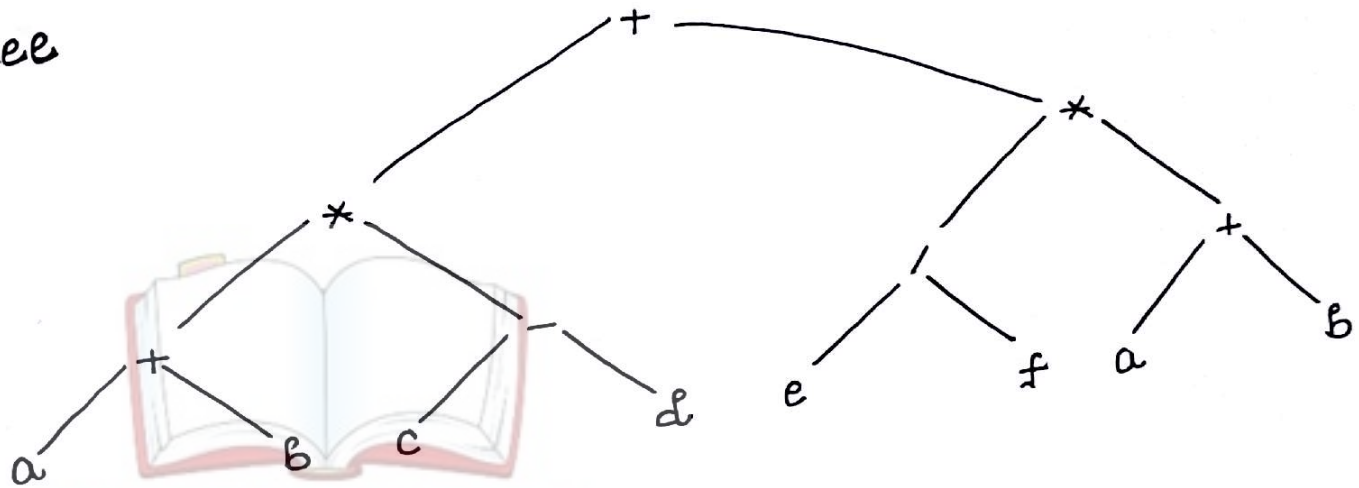
→ $ab+cd-* + ef/ab+*$

→ $ab+cd-*ef/ab+*+$

**Step-2:-** Draw syntax tree :- (ab+cd-* ef/ab+*+)



Syntax tree






xxxxxxx