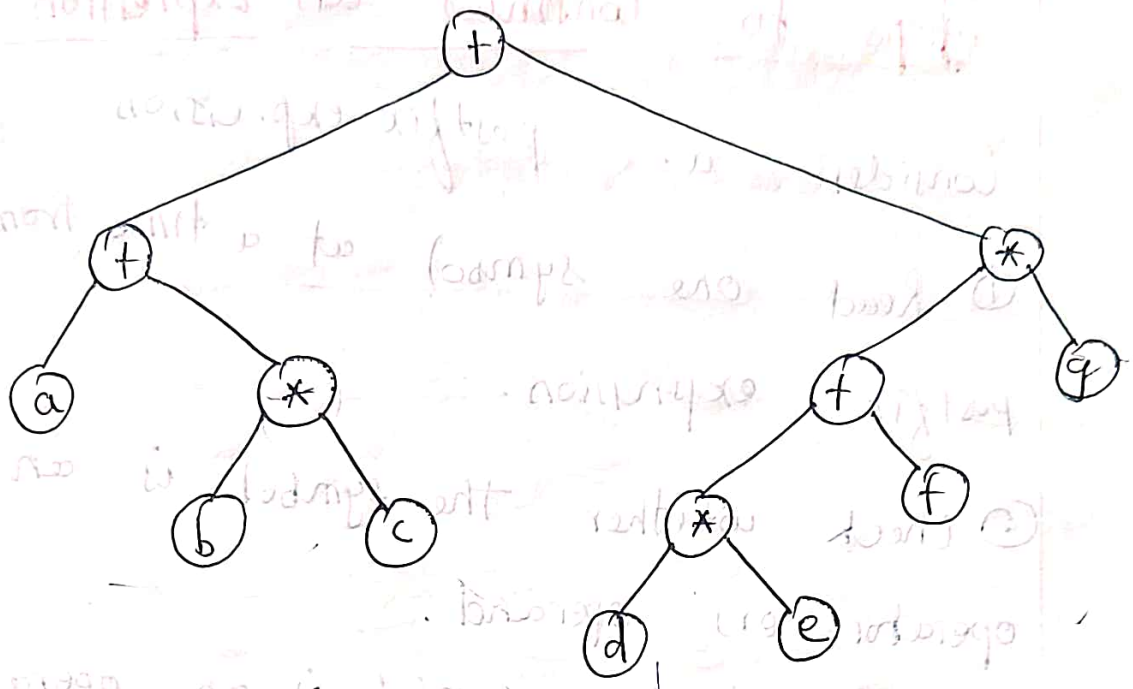
Expression Tree

The expression tree for the expression (a+b*c)+((d+e+f)*g) will shown in following figure



* The leaves of an expression tree are operands, such as constants or variable names, and the other nodes contain operators

* The tree happens to be binary; because all the operations are binary. we can evaluate an expression tree T, by applying the operator at the root to the values obtained by recursively evaluating the left and right subtrees.

* In above example the left subtree evaluates a+(b*c) and right subtree evaluates ((d*e+f)*g)

Steps to Construct an Expression Tree

Consider a postfix expression

① Read one symbol at a time from postfix expression.

② Check whether the symbol is an operator (or) operand.

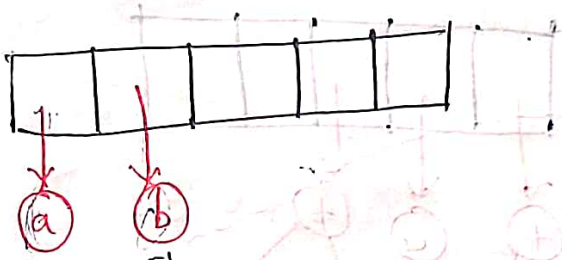
(a) If the symbol is an operand create one node tree and push a pointer on to the stack.

(b) If the symbol is an operator pop two pointers from the stack namely T_1 & T_2 and form a new tree with operator as the root and T_1 as left child T_2 as right child. A pointer to new tree is pushed on to the stack.

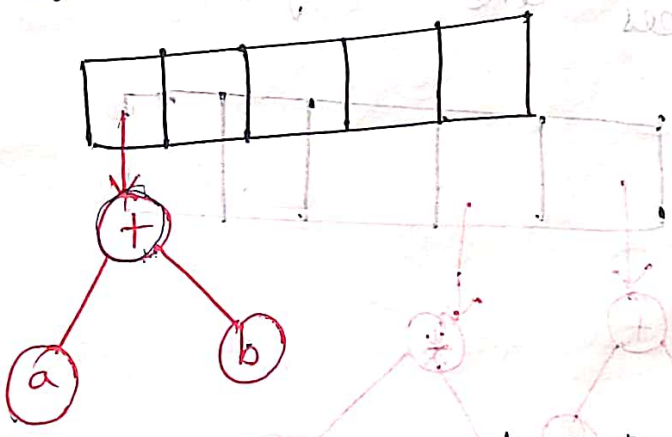
Example

a b + c d e + * *

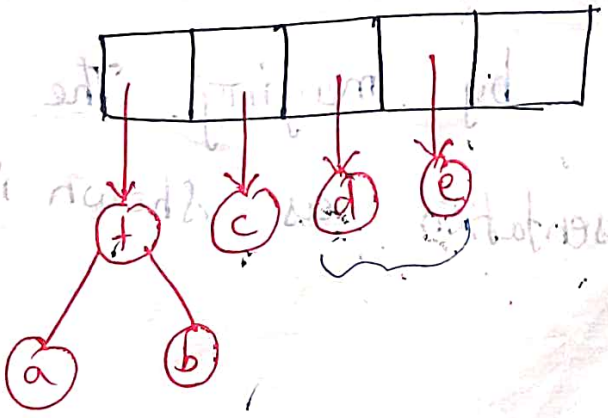
① First two symbols are operands, so we can create one node tree and push their pointers on to the stack.



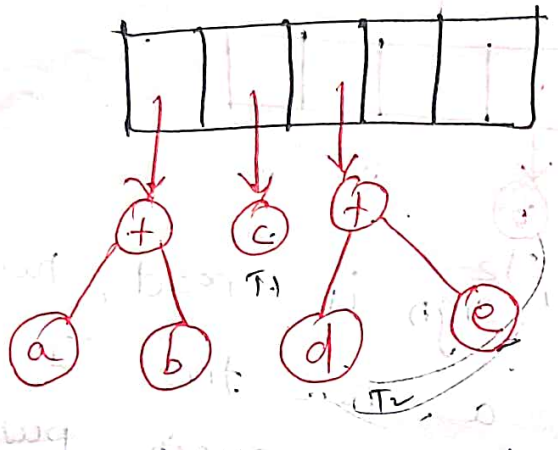
② Next '+' sign is read, two pointers are popped a new tree is formed. and a pointer again pushed on to the stack.



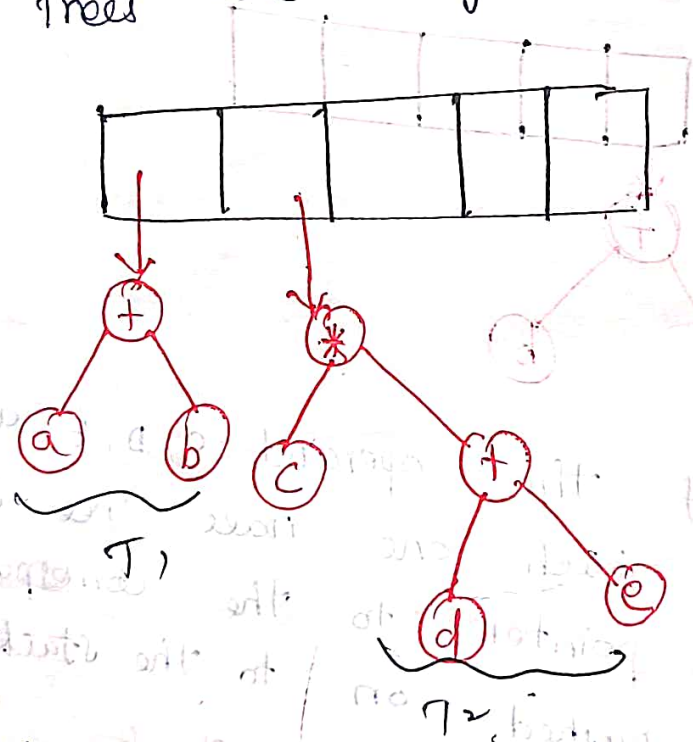
③ Next the operand C, D, E are read and for each one node tree is created and a pointer to the corresponding tree is pushed on to the stack.



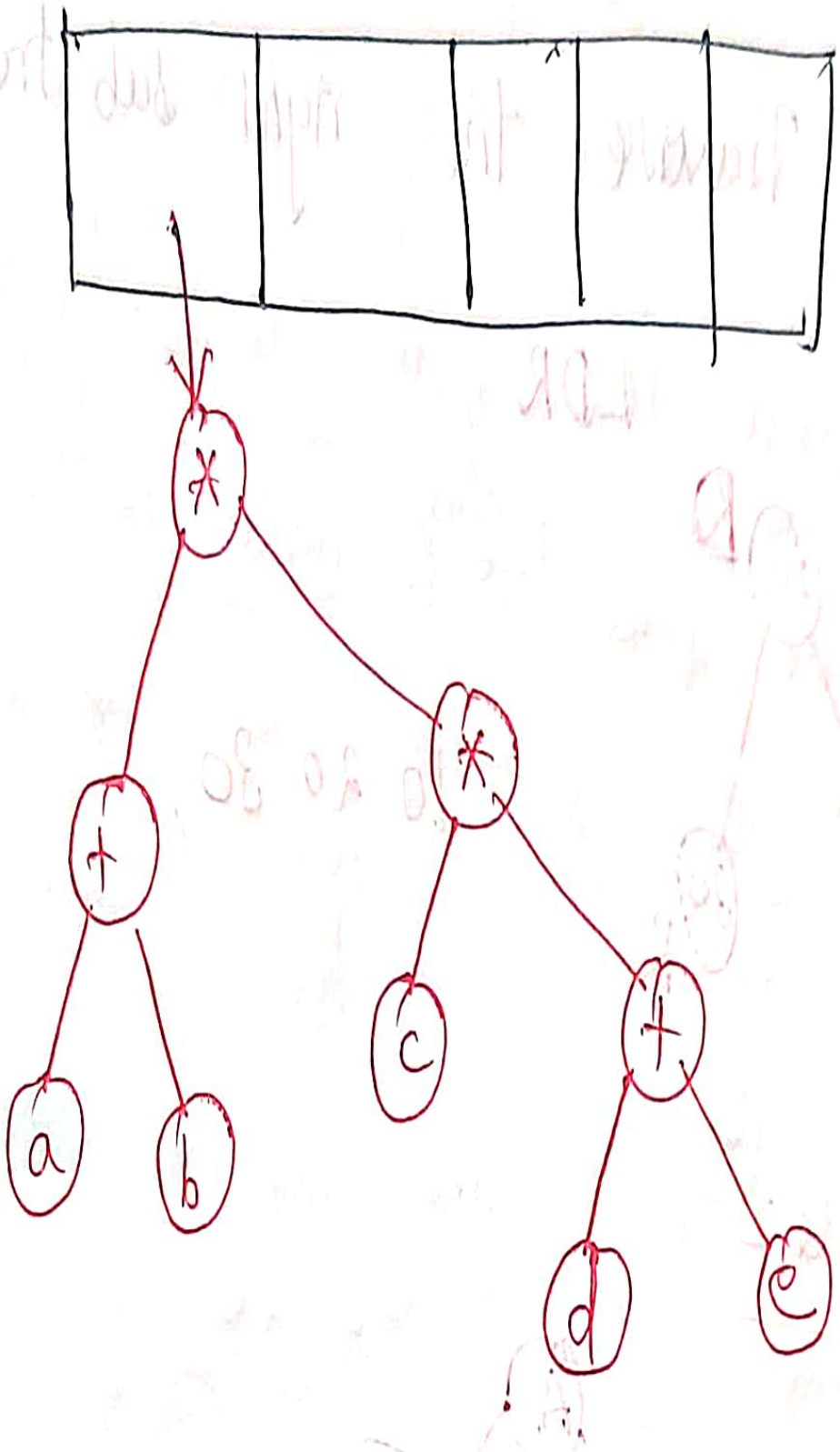
④ Now the '+' is read so last two trees are merged.



⑤ Now the '*' is read so the last two trees are merged.



⑥ Now the '*' is read and tree is formed by merging. The final representation as shown in figure.


*
+
*
a
b
c
+
d
e