DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
Mrs. MEENATCHI R M.E.,
Assistant Professor
E-mail: meenatchi.r@atria.edu

Course Name: Data Structures and Applications
Course Code: 21CS32

Timings:

Monday	:	NIL
Tuesday	:	12:15-01:15/2.00-5.00(B2)
Wednesday	:	10.00-11.00
Thursday	:	10.00-11.00/2.00-5.00(B1)
Friday	:	10.00-11.00

MODULE-4

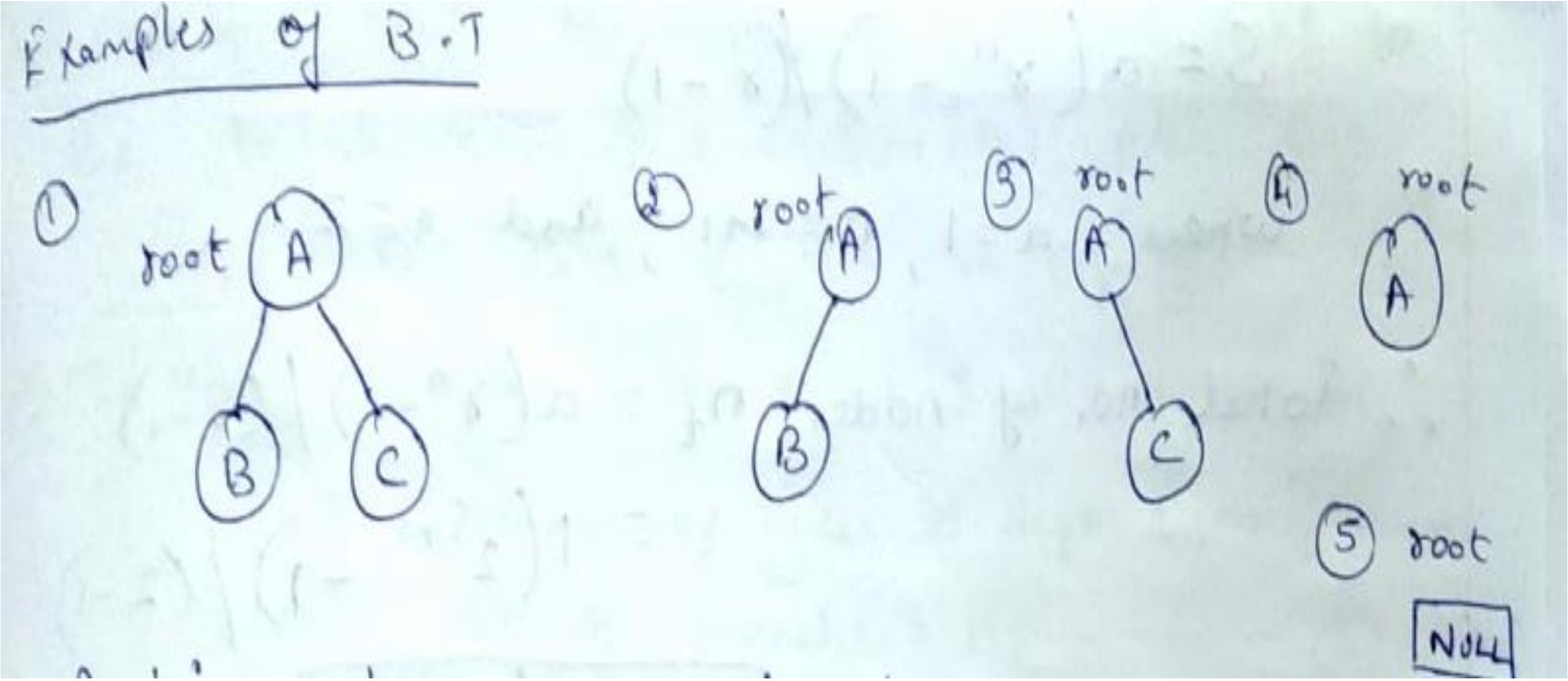
Trees 1: Terminologies, Binary Trees, Properties of Binary trees, Array and linked Representation of Binary Trees, Binary Tree Traversals - In order, post order, preorder; Threaded binary trees, Binary Search Trees – Definition, Insertion, Deletion, Traversal, and Searching

Binary tree

- A binary tree is a tree which has finite set of nodes that is either empty or consist of a root and two sub trees called left sub tree and right sub tree
- Root-If tree is nit empty ,the First node in the tree
- Left sub tree- connected to the left of root
- Right sub tree- connected to the right of root

Binary tree

❖A binary tree means at most two ie., zero ,one or two subtrees are possible. But more than **two sub trees are not permitted**

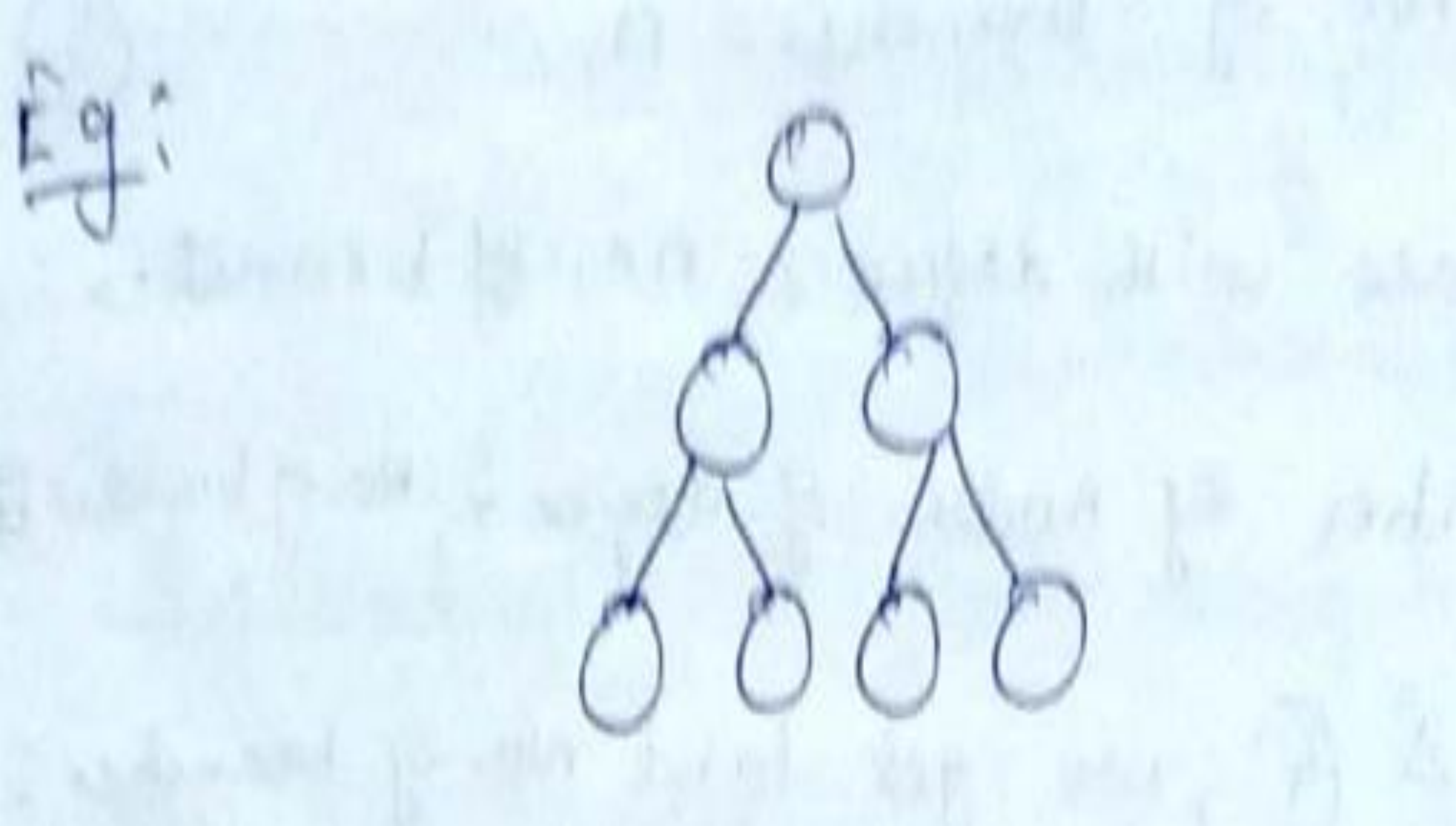


Types of Binary tree

- ❖Strictly Binary Tree
- ❖Skewed Binary Tree
- ❖Complete Binary Tree
- ❖Expression tree
- ❖Binary Search Tree

Strictly Binary Tree

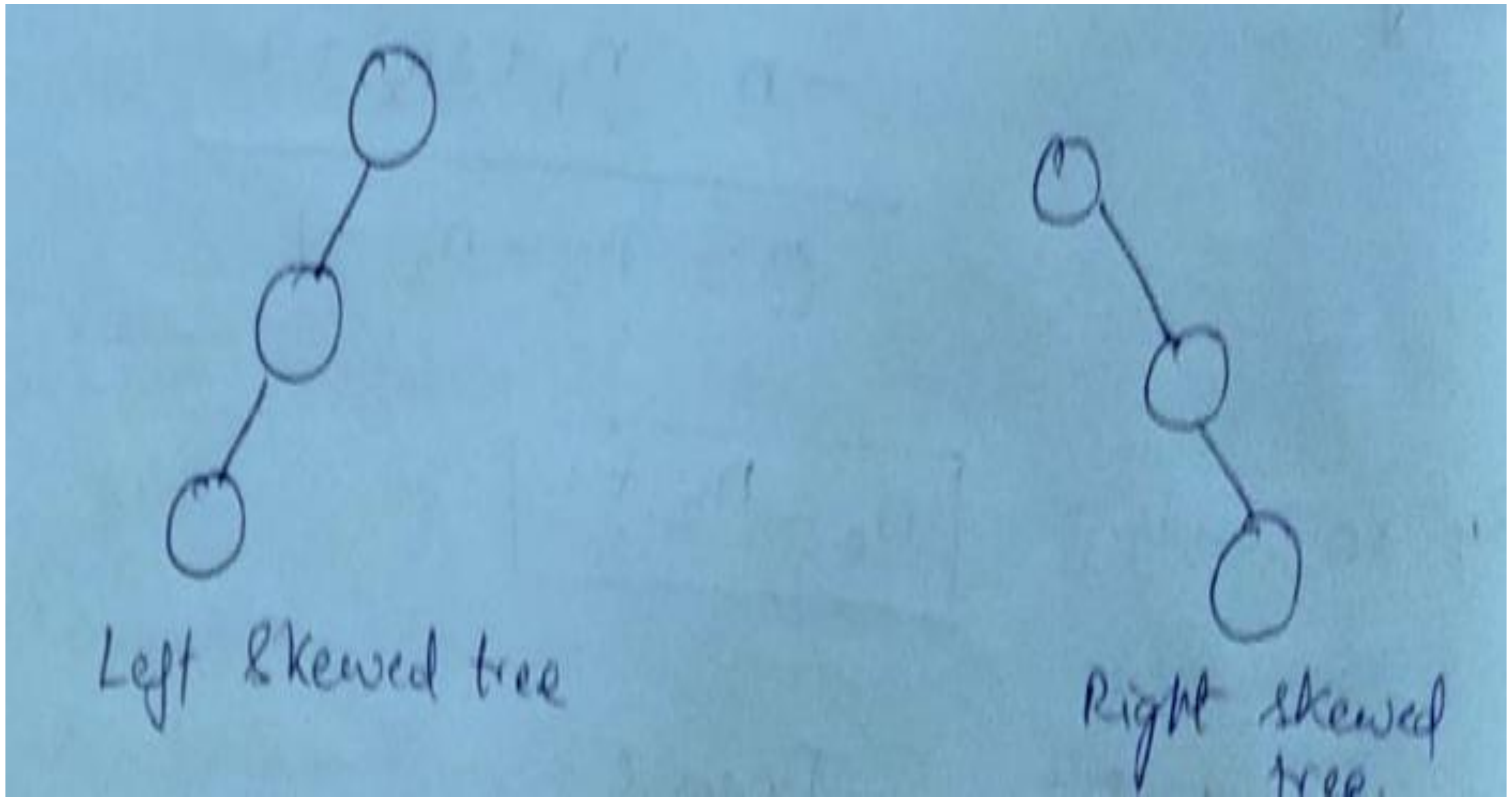
- A binary tree having 2i nodes in any given level I,is called as strictly BT.here every node other than the leaf node has two children



Skewed Binary Tree

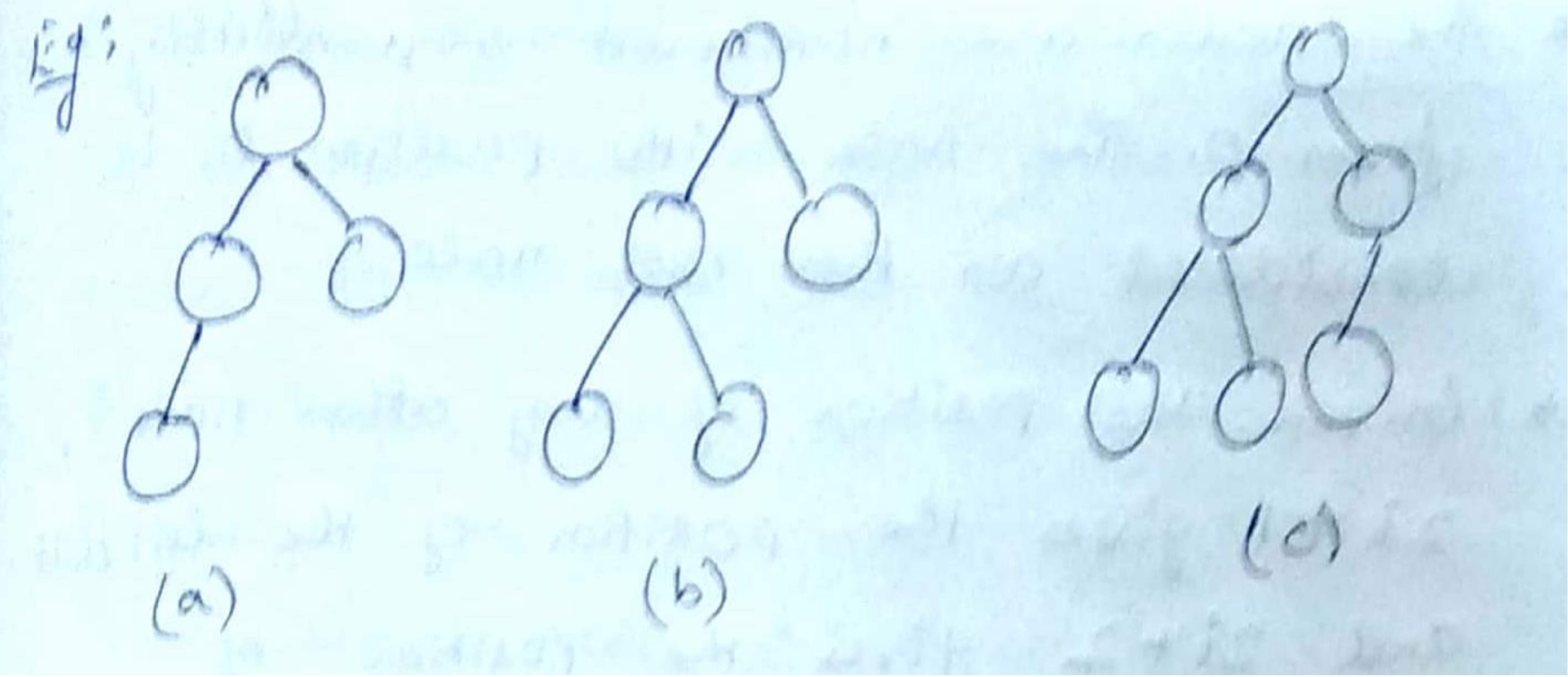
❖A tree consisting of only left subtree or only right subtree is called skewed tree

❖A tree with only left subtree is called left skewed tree,and a tree with only right subtree is called right skewed tree.



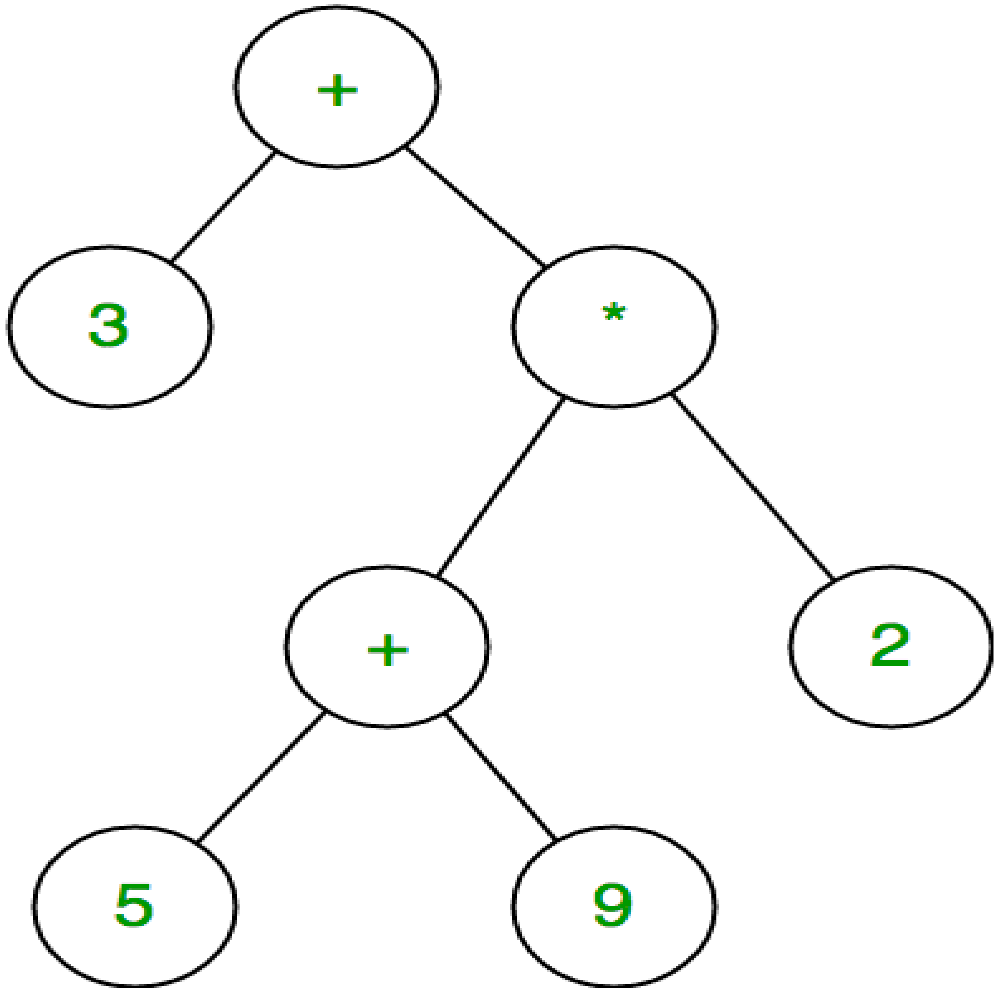
Complete Binary Tree

❖Is a binary tree in which every level, except possibly the last level is completely filled. Also all the nodes should be filled only from left to right (at leaf level)



Expression tree

❖the expression tree is a binary tree in which each internal node corresponds to the operator and each leaf node corresponds to the operand so for example expression tree for 3 + ((5+9)*2) would be:



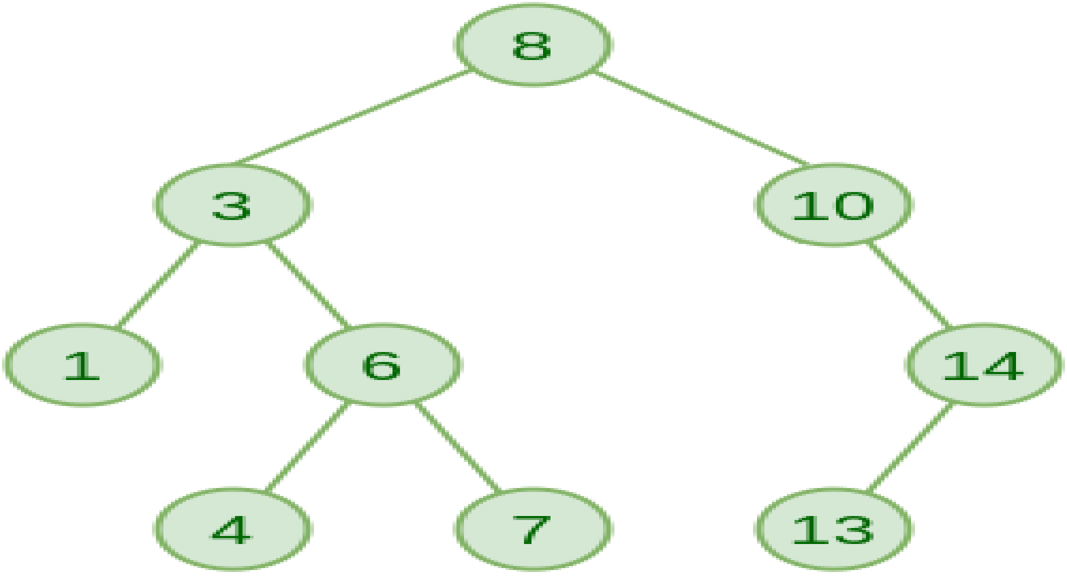
Binary Search Tree

❖Binary Search Tree is a node-based binary tree data structure which has the following properties:

❖The left subtree of a node contains only nodes with keys lesser than the node’s key.

❖The right subtree of a node contains only nodes with keys greater than the node’s key.

❖The left and right subtree each must also be a binary search tree.



Binary Search Tree

Binary Tree Traversals

- ❖Traversing a tree is **visiting each node** in the tree exactly once.
- ❖ When a node is visited, some operation (such as outputting its data field) is performed on it.
- ❖ A full traversal produces a **linear order** for the nodes in a tree.
- ❖When traversing a tree each node and its subtrees must be treated in the same fashion.
- ❖Based on this there are three types of traversals inorder, preorder and postorder traversals.

Inorder Traversal

- ❖in order traversal move down the tree toward the left until we can go no farther.
- ❖Then "visit" the node,
- ❖ move one node to the right and continue.
- ❖ If we cannot move to the right, go back one more node and continue
- ❖A precise way of describing this traversal is by using recursion as follows

Inorder Traversal

```
void inorder(TreeNode * ptr)
{/* inorder tree traversal */ if (ptr)
{
inorder(ptr→leftChild);
printf("%d",ptr→data);
inorder(ptr→rightChild);
}
}
```

Preorder Traversal

- ❖visit a node
- ❖ traverse left, and continue.
- ❖When you cannot continue, move right and begin again or move back until you can move right and resume."

Preorder Traversal

```
void preorder(TreeNode *ptr)
{/* preorder tree traversal */ if (ptr)
{
printf("%d", ptr→data);
preorder(ptr→leftChild);
preorder(ptr→rightChild);
}
}
```

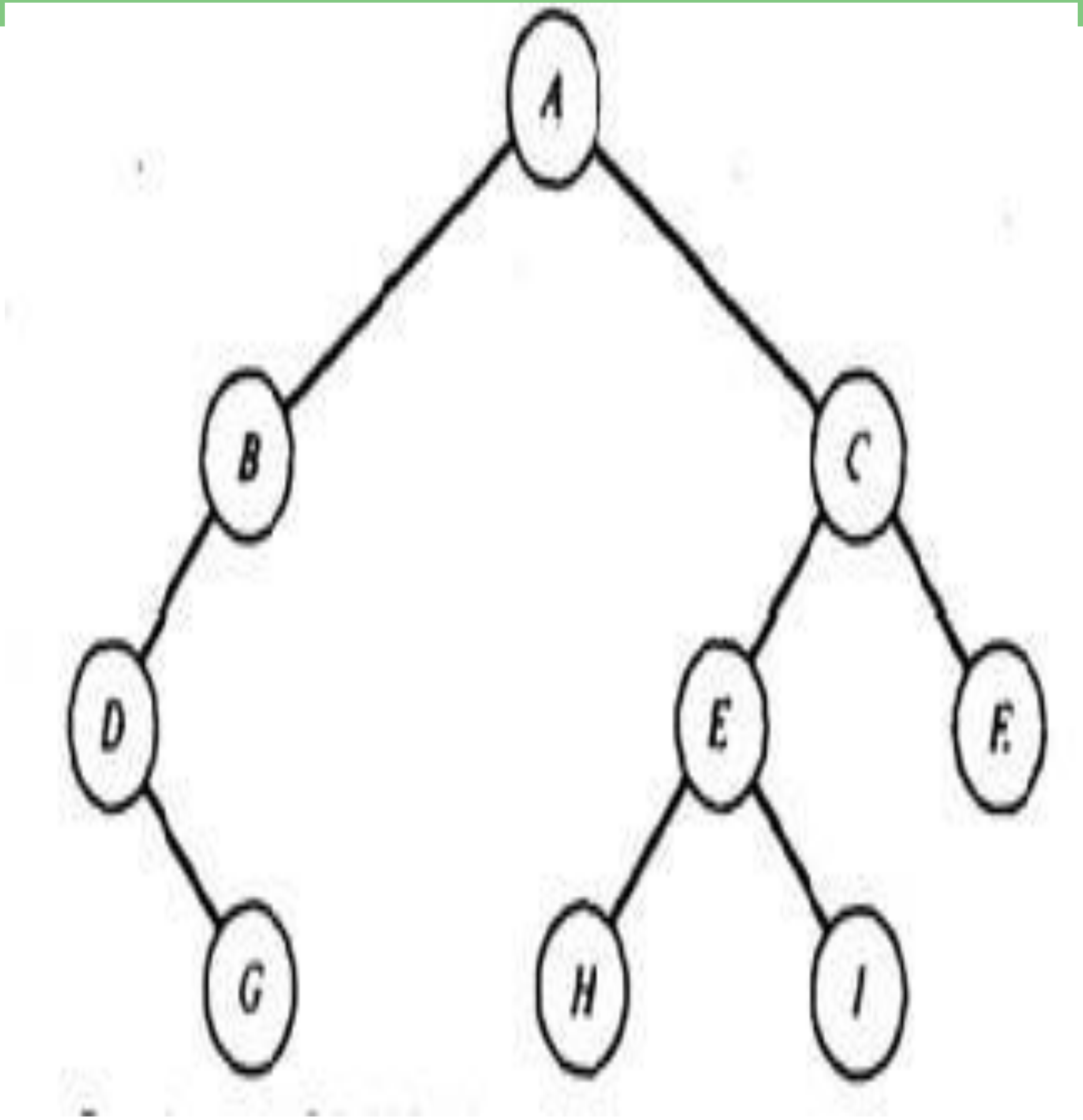
Postorder Traversal

- traverse left, and continue.
- When you cannot continue, move right and traverse right as far as possible
- Visit the node

Preorder Traversal

```
void postorder(TreeNode *ptr)
{/* postorder tree traversal */ if (ptr)
{
postorder(ptr→leftChild);
postorder(ptr→rightChild);
printf("%d",ptr→data);
}
}
```

Example:



In order Traversal: DGBAHEICF

Preorder Traversal: ABDGCEHIF

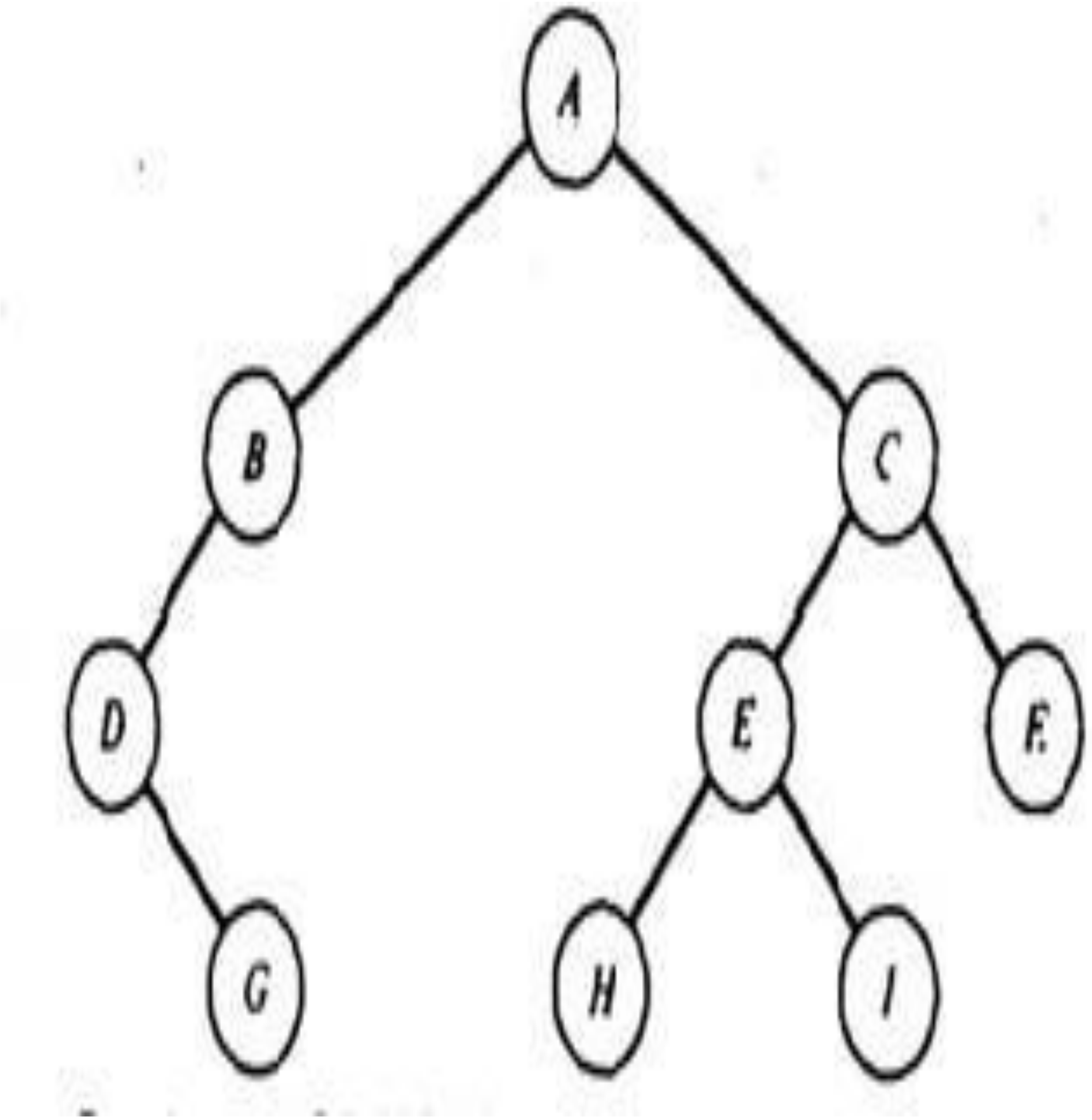
Post order Traversal: GDBHIEFCA

Example

In order Traversal: DGBAHEICF

Preorder Traversal: ABDGCEHIF

Post order Traversal: GDBHIEFCA

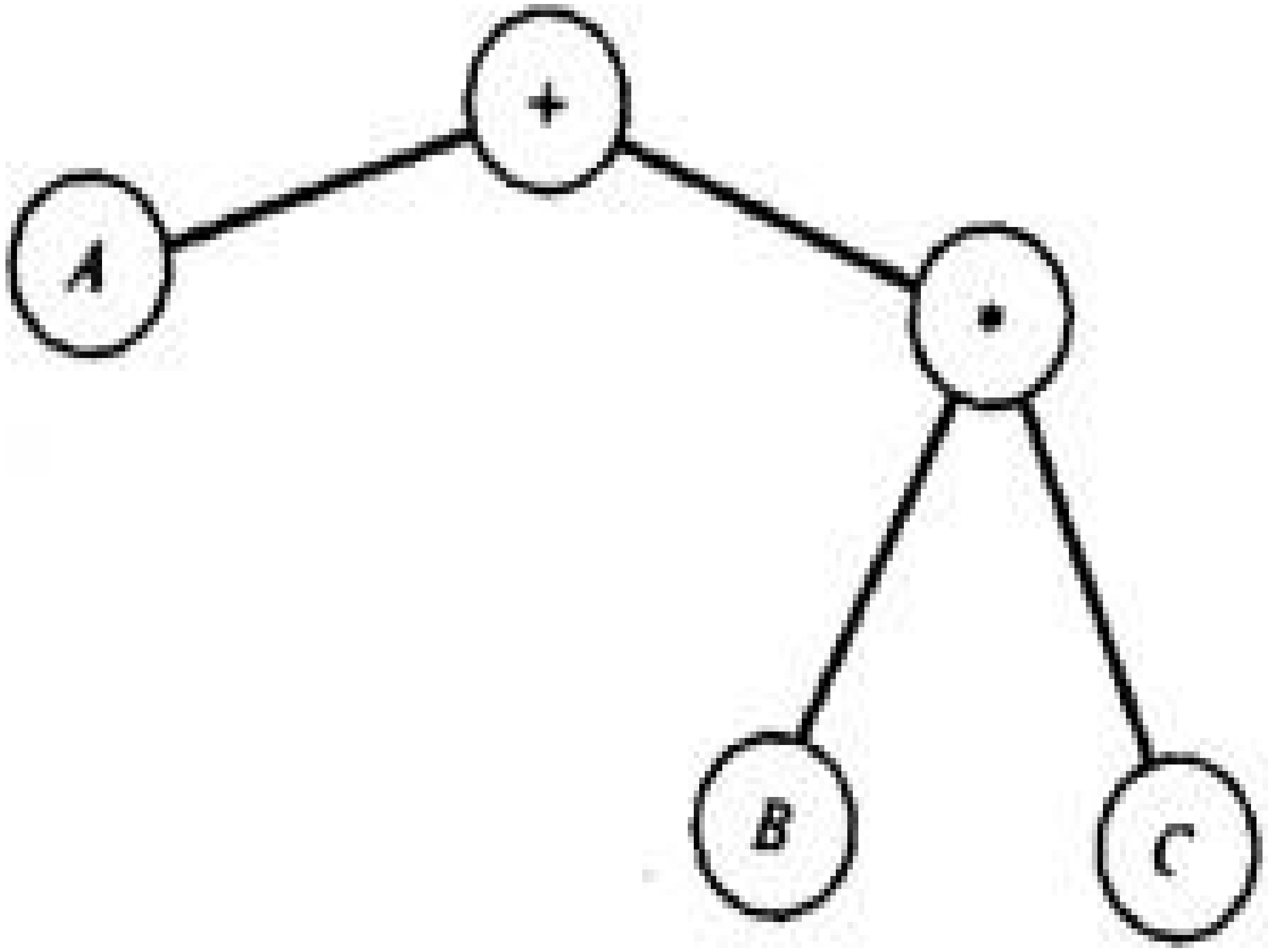


Example

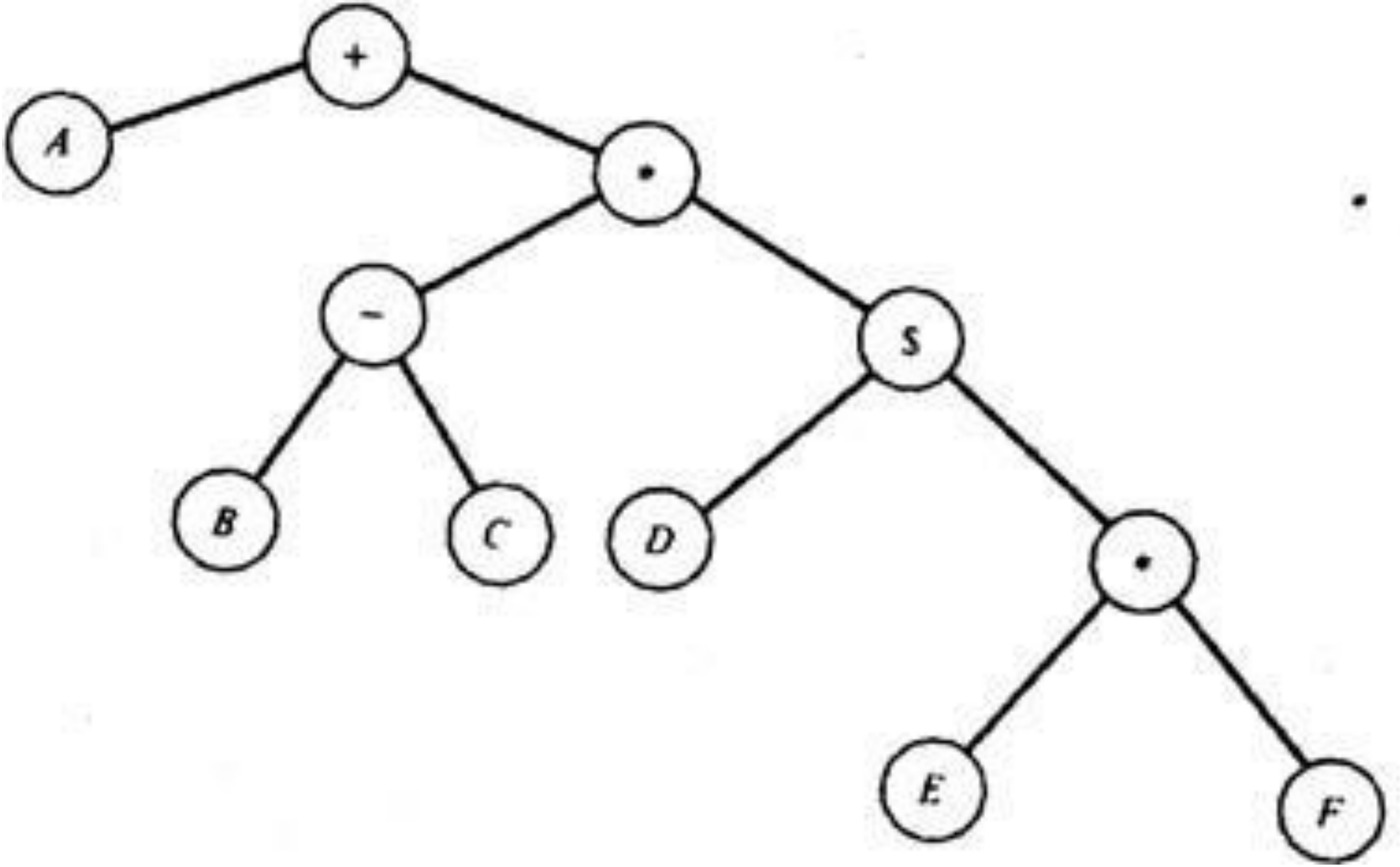
In order Traversal:A+B*C

Preorder Traversal:+A*BC

Post order Traversal:ABC*+



Example



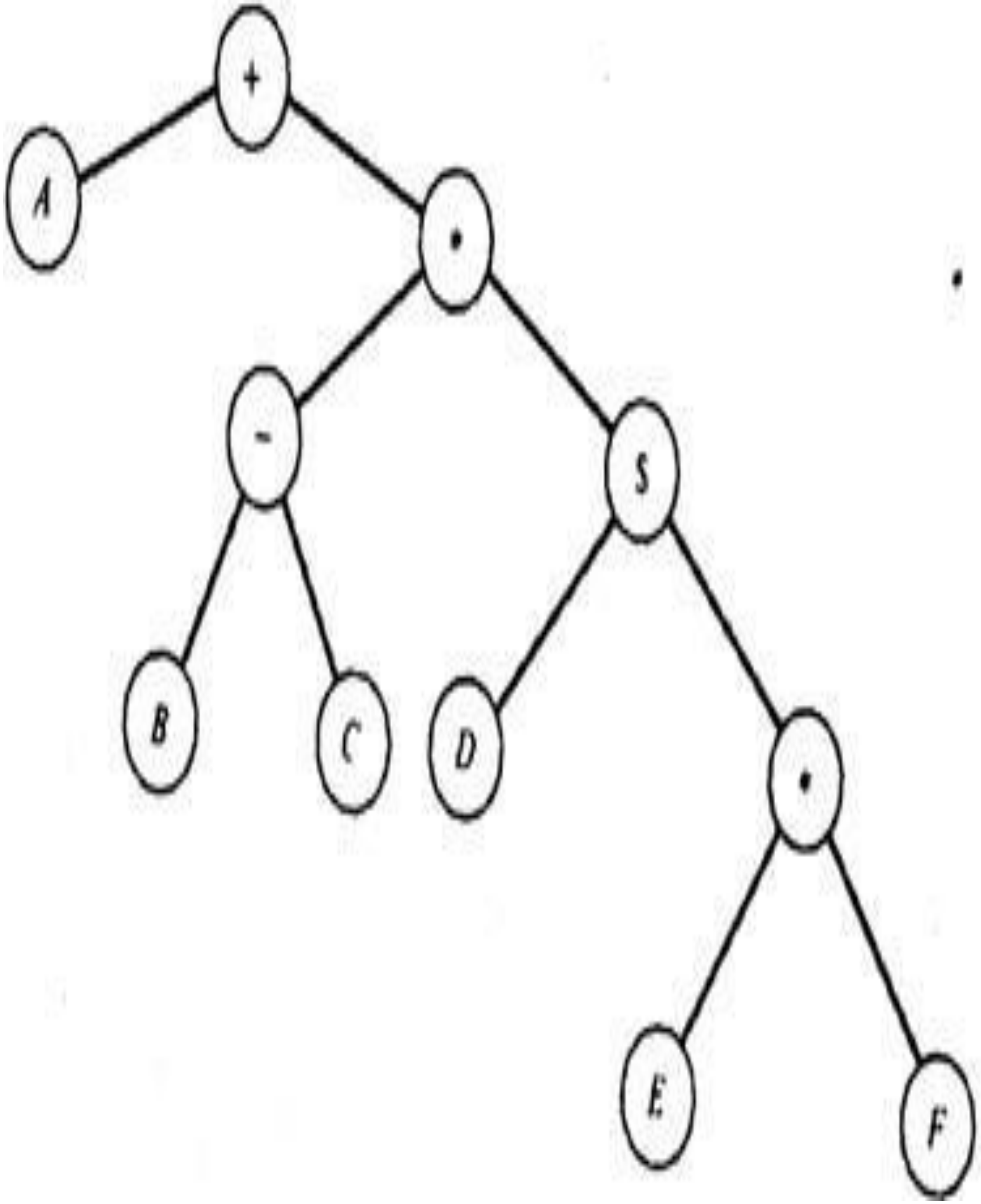
A+(B-C)*D$(E*F) is represented as follows

Example

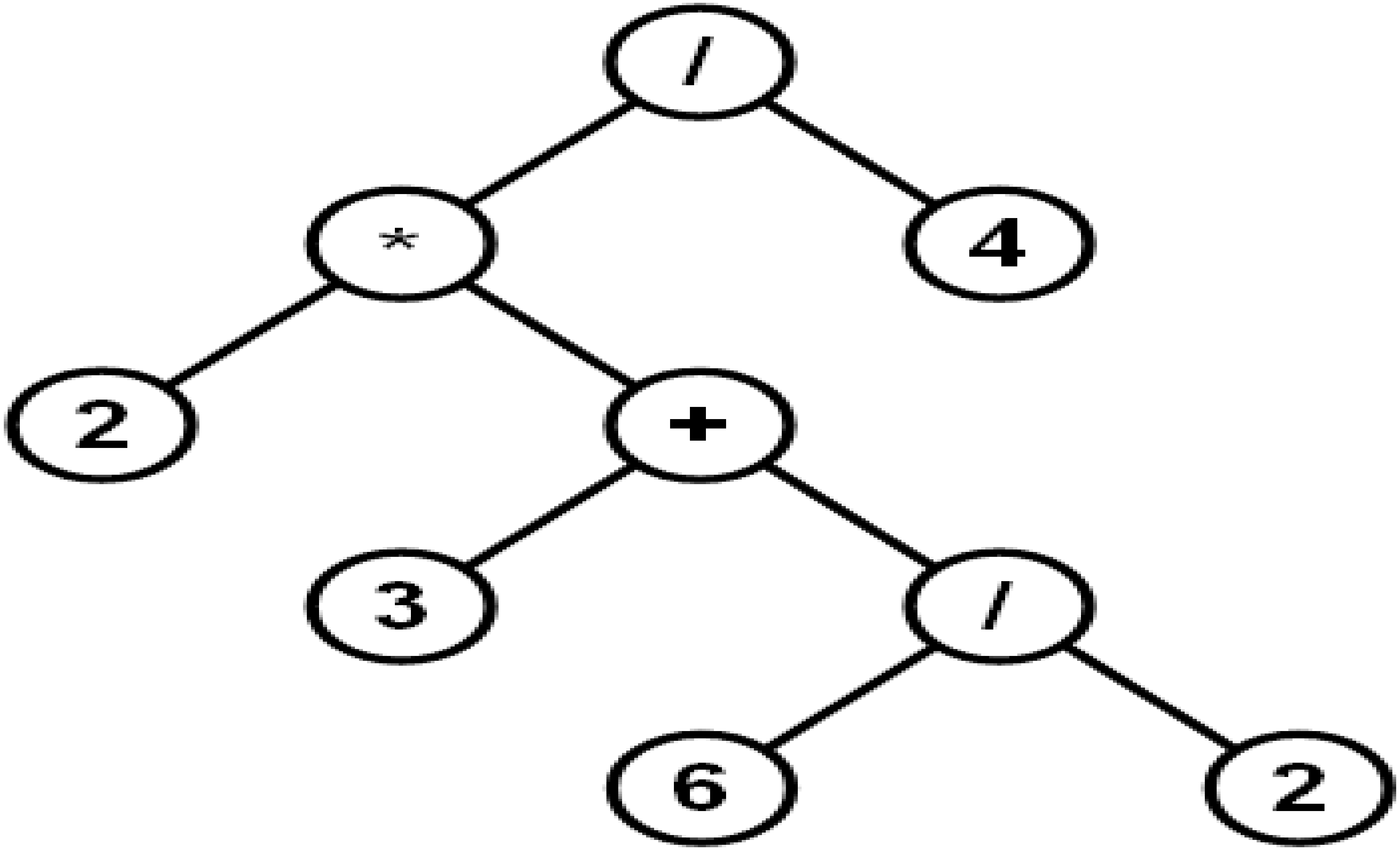
In order Traversal: A+(B-C)*D$(E*F)

Preorder Traversal: +A*-BC$D*EF

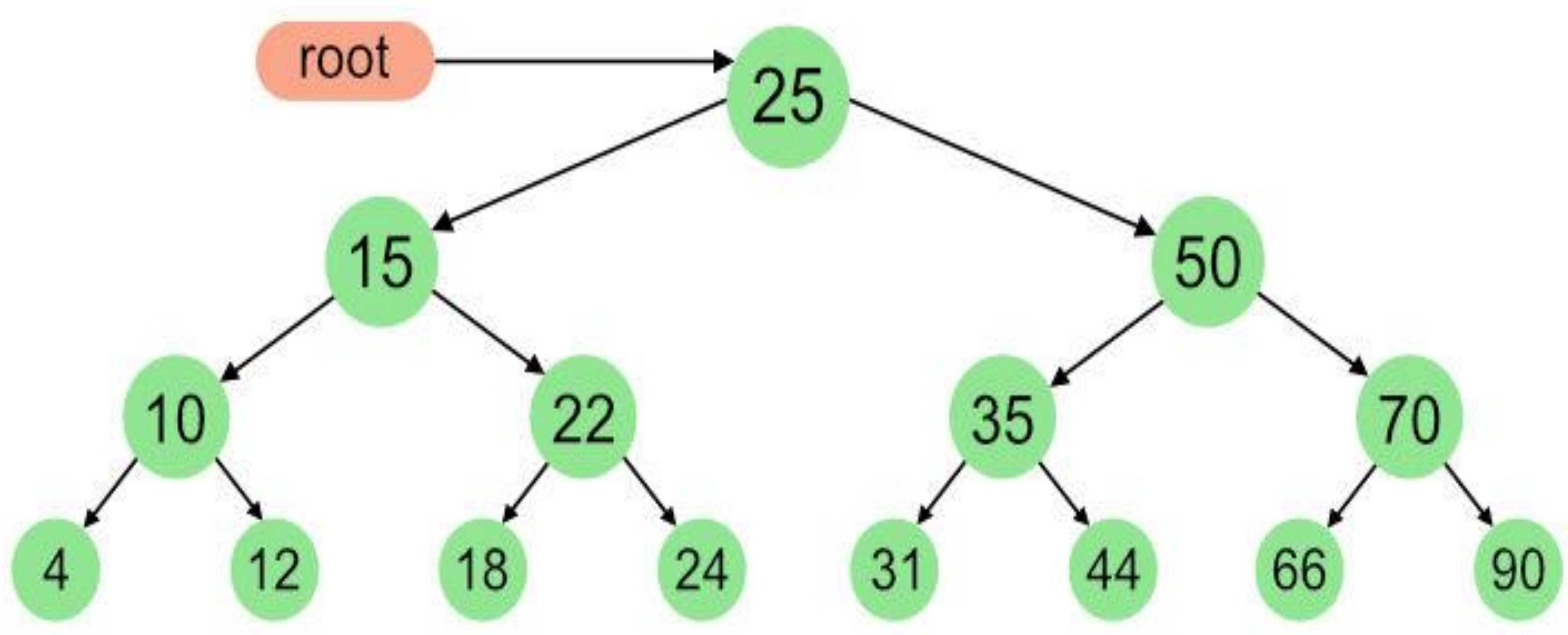
Post order Traversal: ABC-DEF*$*+



Example



Example


root
25
15
50
10
22
35
70
4
12
18
24
31
44
66
90