Tree.

A tree is a special case of a digraph used to express hierarchical relationships or multiway decisions in computing

eg: game tree
company structure
family tree
language tree
expression tree
Searching tree etc ...

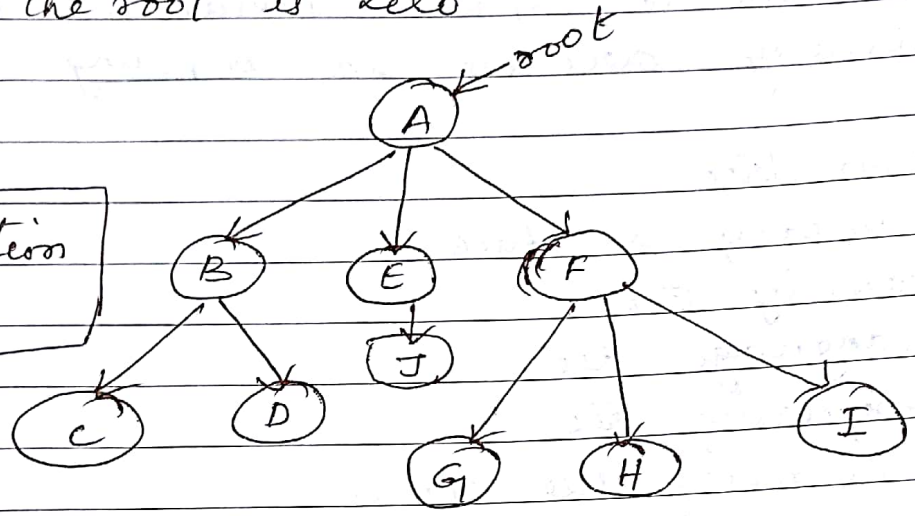
Basic tree concepts.

A tree consists of a finite set of elements called nodes, and a finite set of directed lines, called branches, that connect the nodes.

The number of branches associated with a node is the degree of the node. When the branch is directed toward the node, it is an indegree branch. When the branch is directed away from the node, it is an outdegree branch. The sum of the indegree and outdegree branches is the degree of the node

If the tree is not empty, the first node is called the root. The indegree of the root is zero

Representation of tree



With the exception of the root, all of the nodes in a tree must have an indegree of exactly one, that is, they may have only one predecessor.

Terminology

Many different terms are used to describe the attributes of a tree.

leaf is any node with an outdegree zero

internal node A node that is not a root or a leaf is known as an internal node

A node is a **parent** if it has successor nodes

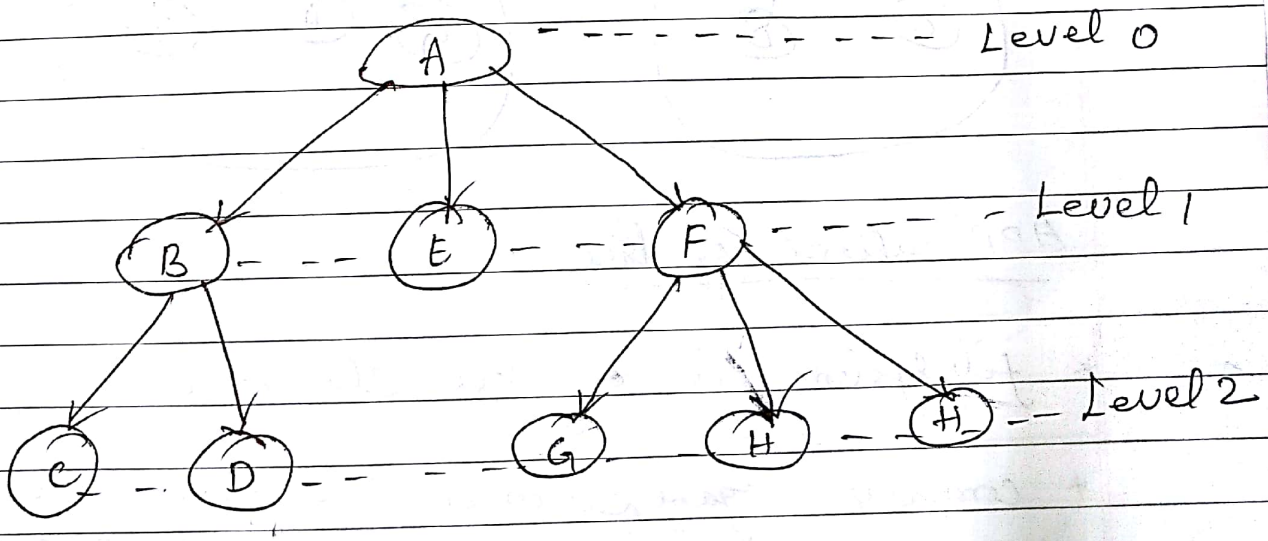
A node with a predecessor is a **child**

Two or more nodes with the same parent are siblings.

An ancestor is any node in the path from the root to the node.

A descendent is any node in the path below the parent node.

The level of a node is its distance from the root.

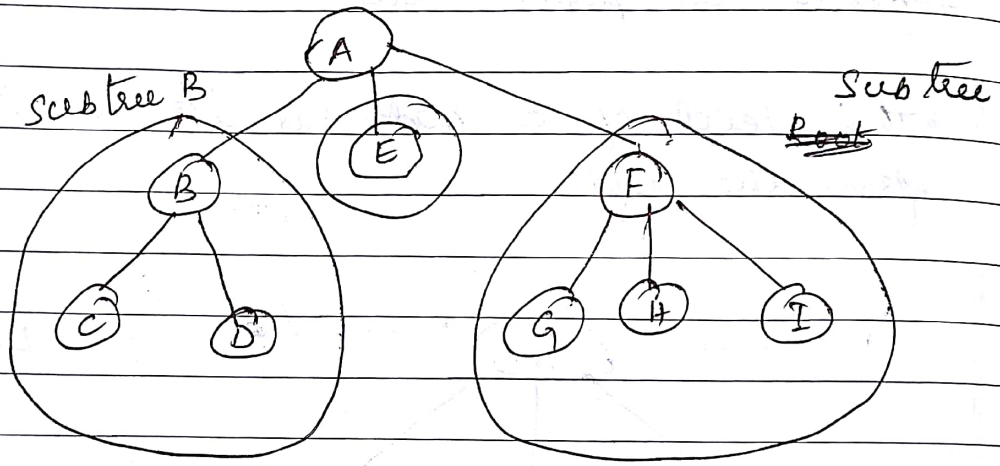


Root A
Parents A, B, F
Children B, E, F, C, D, G, H

Siblings {B, E, F}, {C, D} {G, H, I}
Leaves : C, D, E, G, H, I
Internal nodes } B, F

The height of the tree is the level of the leaf in the longest path from the root plus 1.

A tree may be divided into subtrees. A subtree is any connected structure below the root. The first node in a subtree is known as the root of the subtree and is used to name the subtree. Subtrees can also be further subdivided into subtrees.

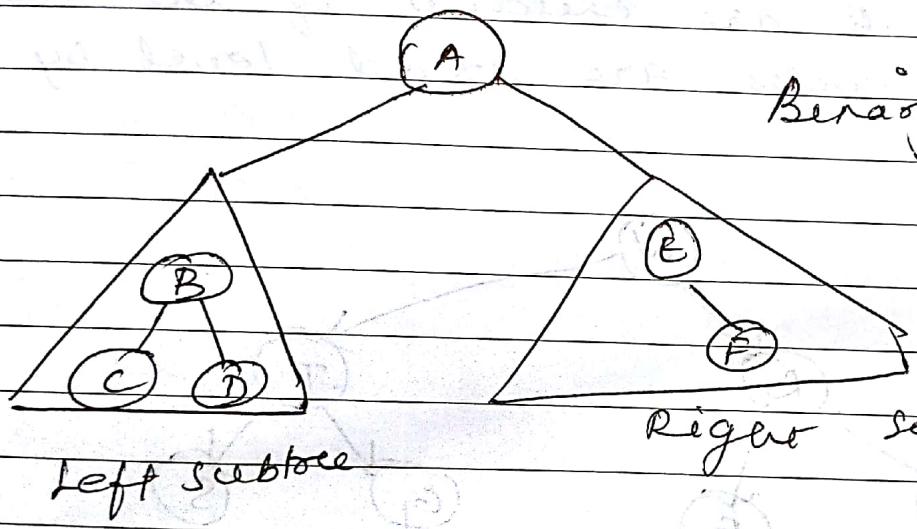


Applications of tree

* file system is a tree structure
* company organizational structure : division, dept, etc.
* Database indexes are normally stored as variants of B^+ tree.
* tree structure for location (country, region, state, town etc)

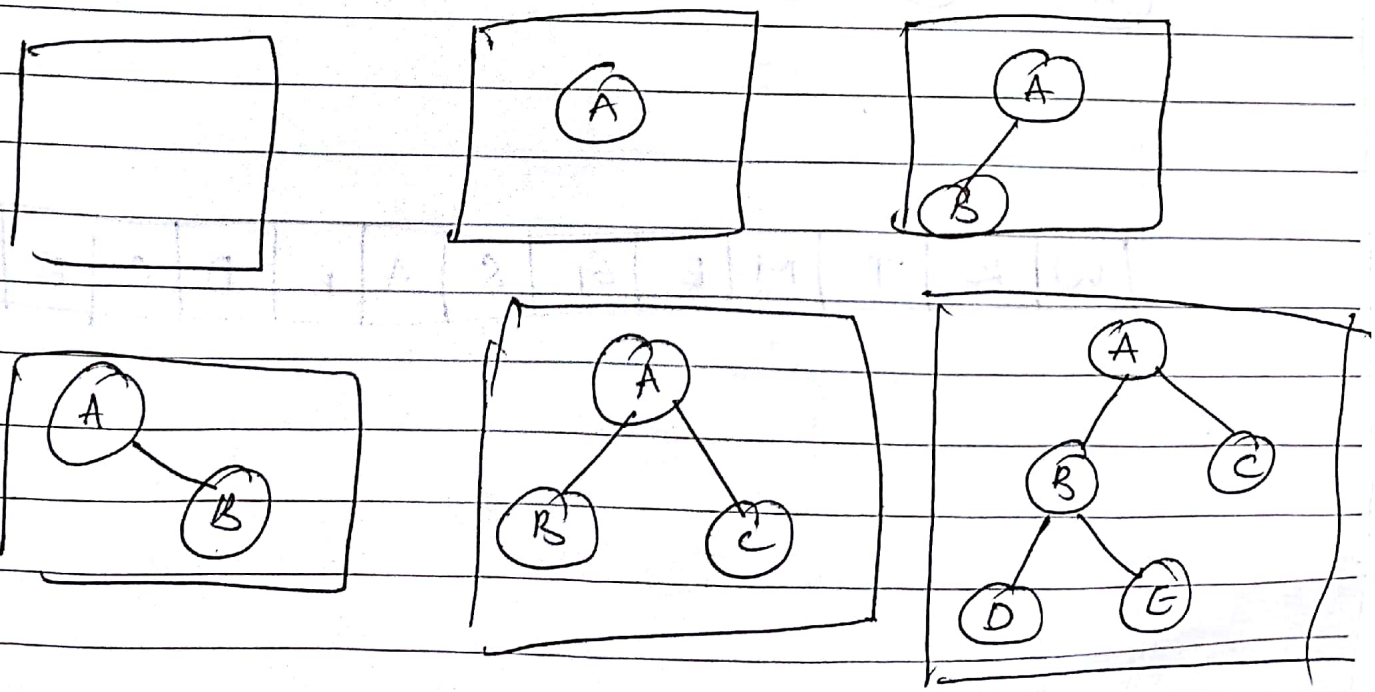
Binary tree

A binary tree is a tree in which no node can have more than two subtrees. The maximum outdegree for a node is two. In other words, a node can have zero, one or two subtrees. These subtrees are designated as the left subtree and the right subtree.



Binary tree

Collection of Binary Trees



Traversing a Binary Tree

Traversing means visiting each node exactly one in a systematic manner. It mainly helps in performing display, search, insertion, deletion on the binary tree elements

In Singly Linked list, this concept is very simple as the links are linearly arranged. However, in a binary tree, there are two links left link and right link. Hence three ways of traversing a binary tree.

1) Preorder <root> <left> <right>
2) Inorder <left> <root> <right>
3) Postorder. <left> <right> <root>

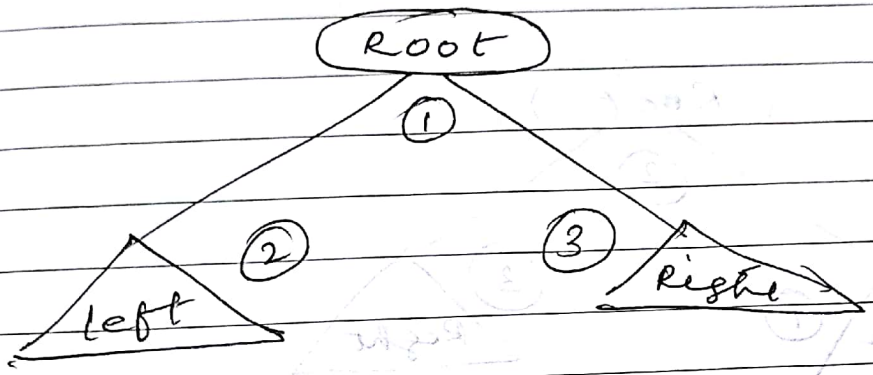
Each of these method involve the visiting of the root and traversing its left and right subtrees.

The only difference among the methods is the order in which these three operations are performed.

Preorder Traveral

Preorder [R L R]

1) visit the Root node
2) Traverse the left subtree recursively
3) Traverse the Right subtree recursively

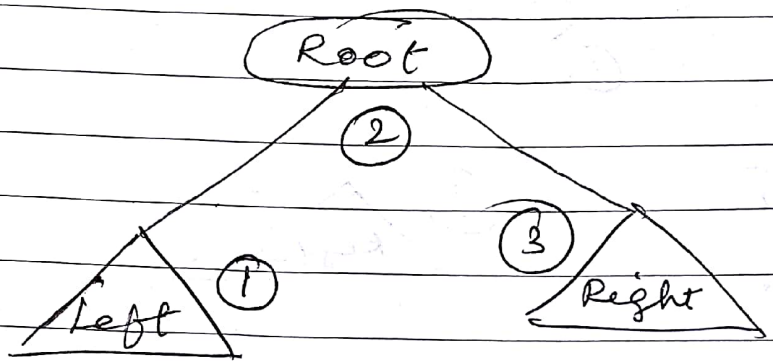



```
void preorder (Struct node * root)
{
    Struct node *r;
    r = root;
    if (r != NULL)
    {
        printf("%d", r->data);
        preorder(r->left);
        preorder(r->right);
    }
}
```

Inorder Traversal

Inorder [L R R]
↓root

1) Traverse the left subtree recursively
2) Visit the root node
3) Traverse the Right subtree recursively



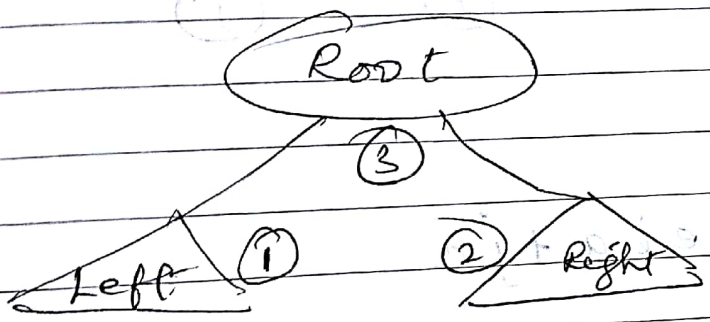

```
void inorder (struct node *root)
{
  Struct node *r;
  r = root;

  if (r != NULL)
  {
    inorder (r → left);
    Printf("%d", r → data);
    inorder (r → right);
  }
}
```

Postorder Traversal

Postorder [L R R(root)]

1) Traverse the left subtree recursively
2) Traverse the Right subtree recursively
3) Visit the root node

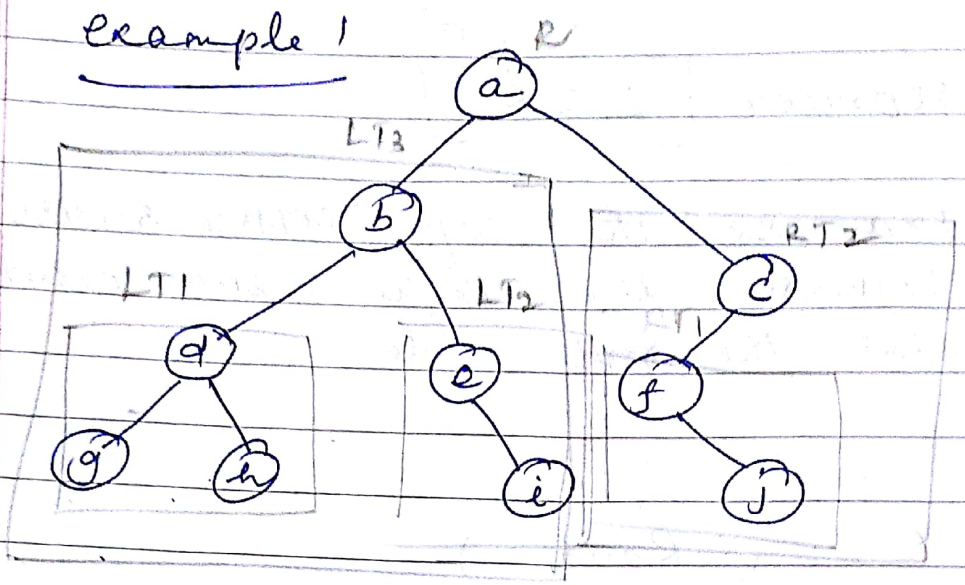



```
Void Postorder (struct node *root)
{
  struct node *r;
  r = root;

  if (r != NULL)
  {
  postorder (r -> left);
  postorder (r -> right);
  printf ("%d", r -> data);
  }
}
```

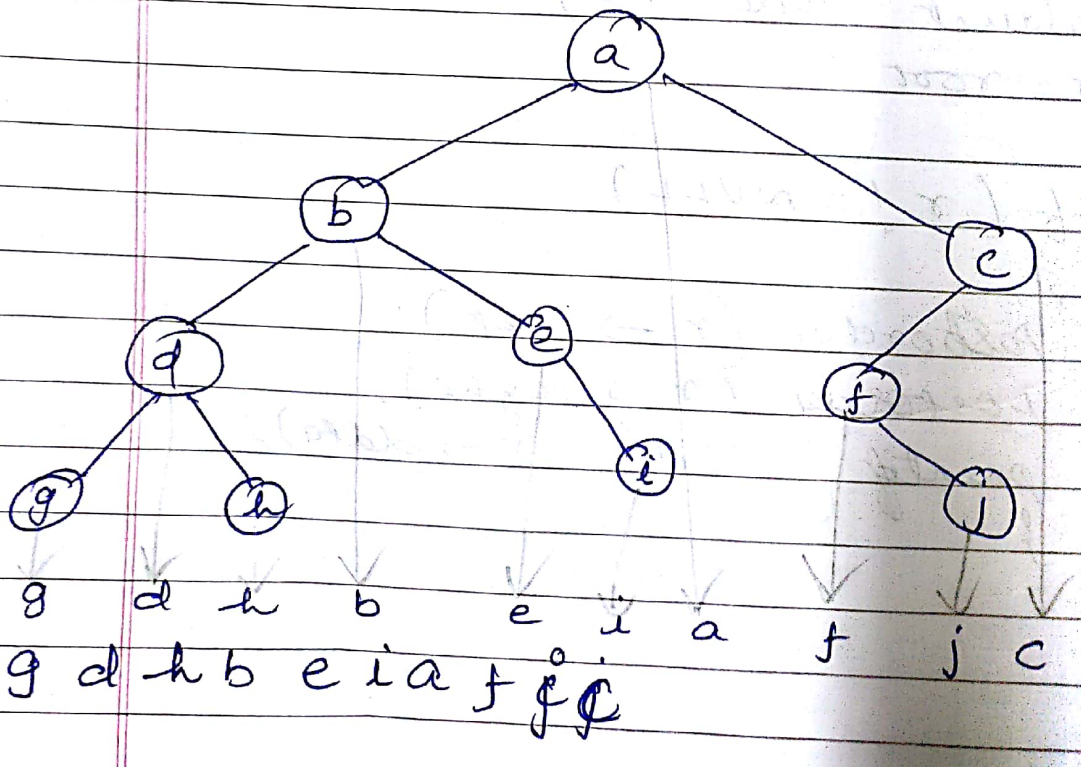
Inorder Traversal (Left, root, Right)

example 1



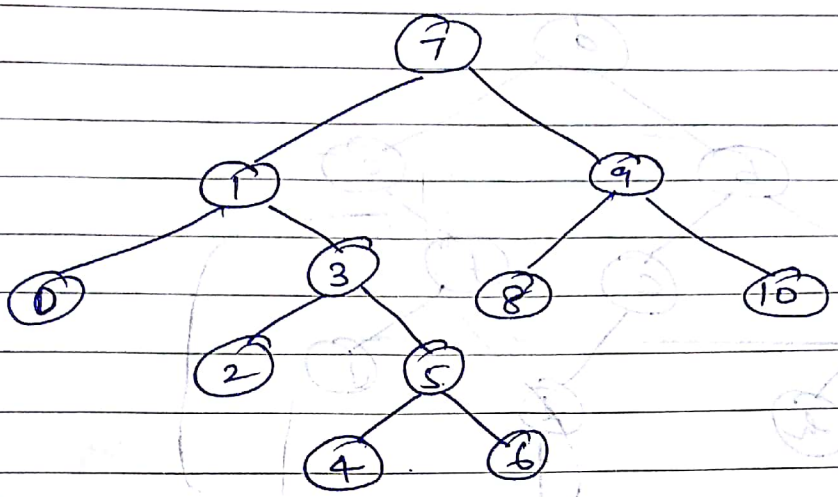
g d h b e i a f j c

Inorder by Projections (Squishing)



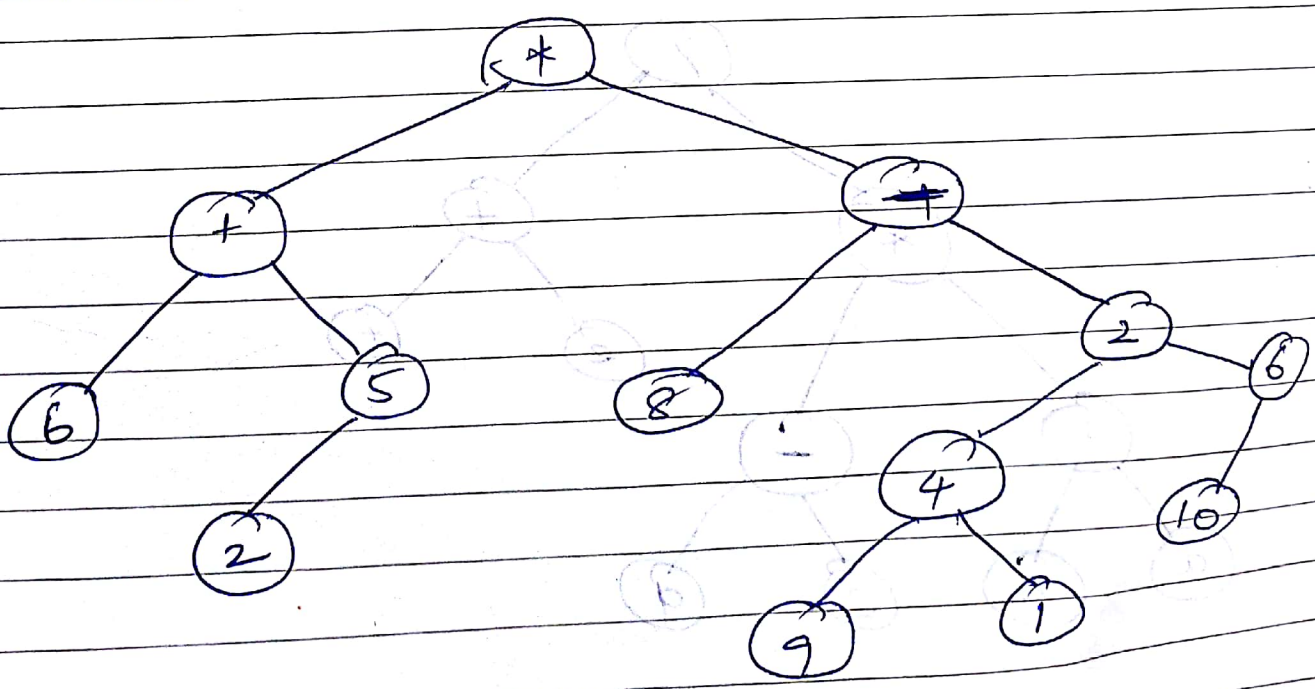
Post.

Example 2



0 1 2 3 4 5 6 7 8 9 10

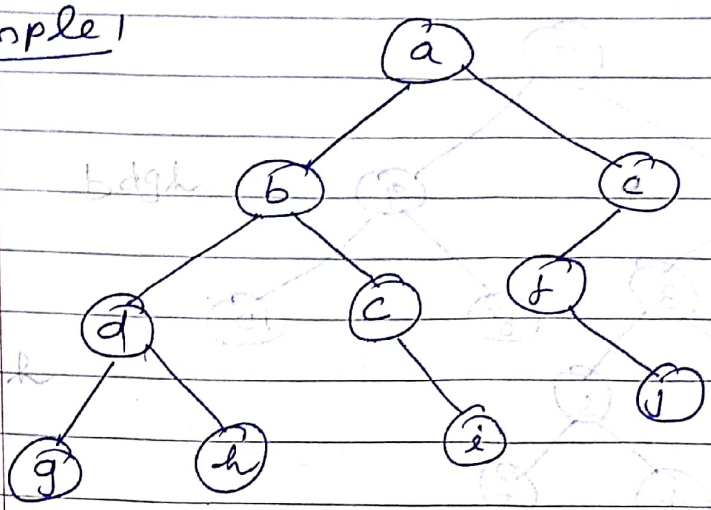
Example 3



6 + 2 5 * 8 - 9 4 1 2 10 6

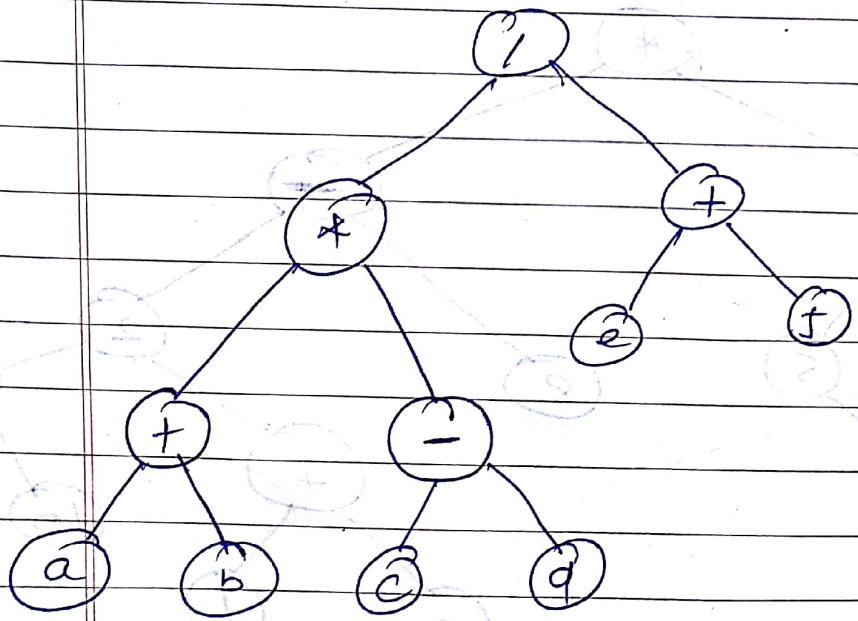
Preorder <root> <left> <right>

example 1



a b d g h c i e f j

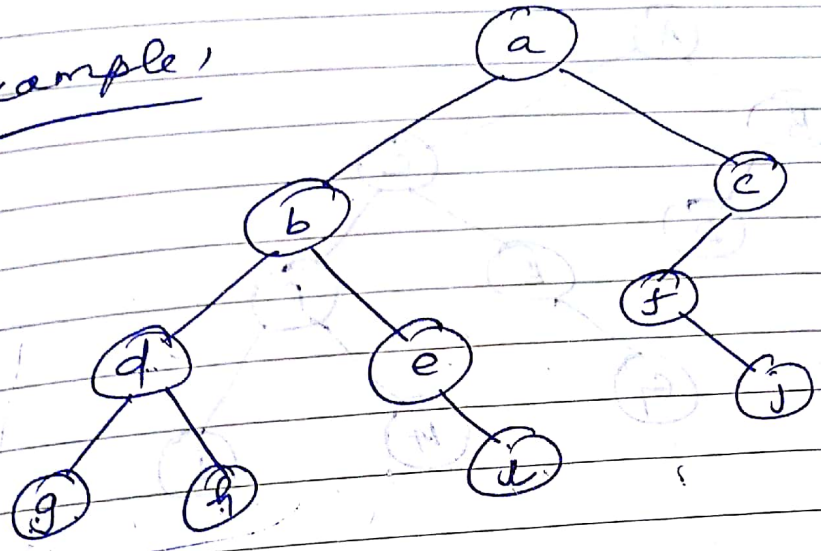
example 2



/ * + a b - c d + e f

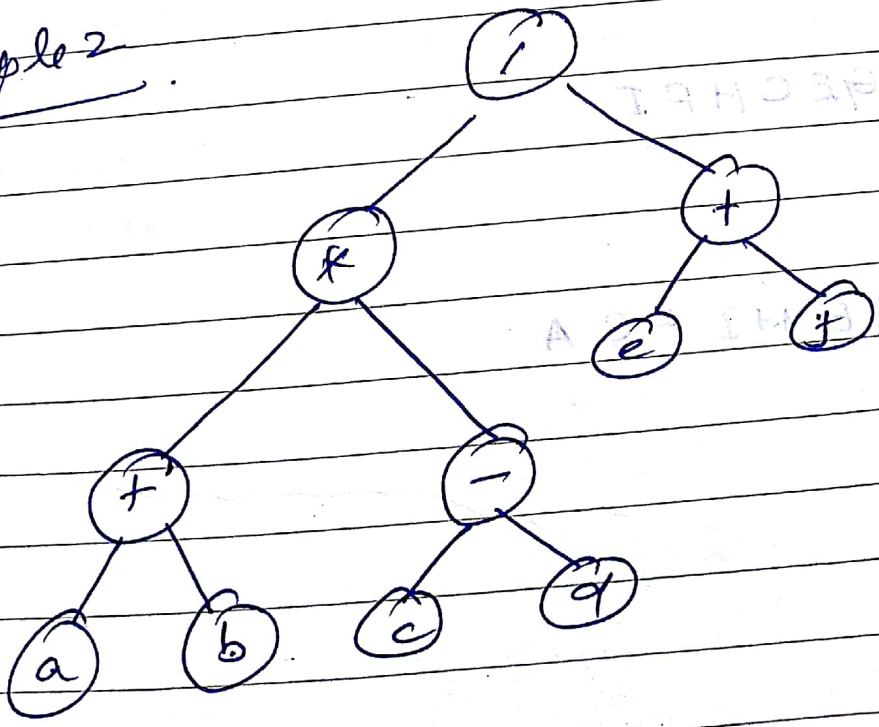
Postorder Traversal <left> <right> <root>

Example 1:



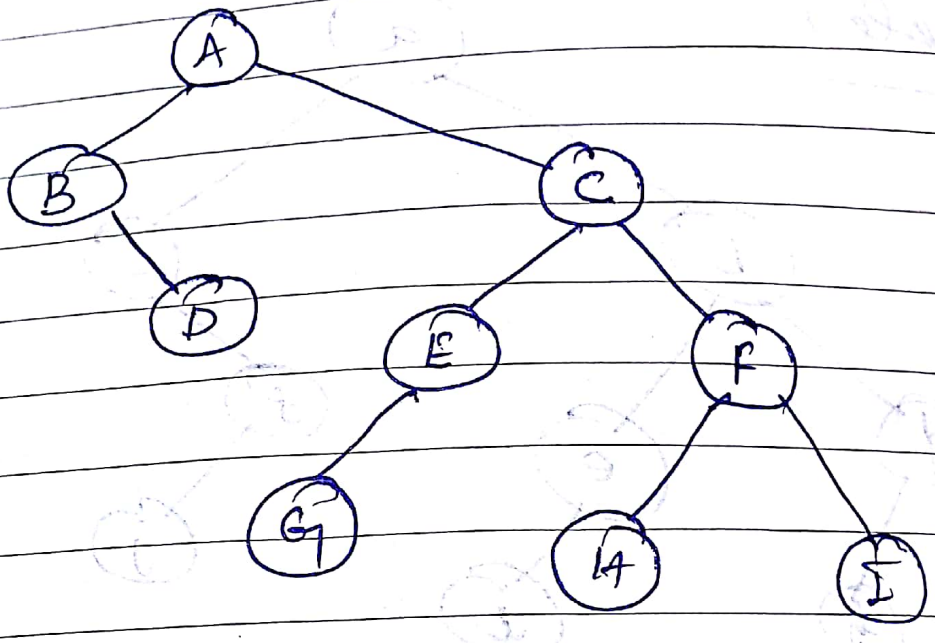
g h d i e b j f c a

Example 2:



a b + c d - * e f + /

Some more example



Preorder

A B D C E G F H I

Inorder

B D A G E C H F I

Postorder

D B G E H I F C A