DSA LAB-4

21BCE7371

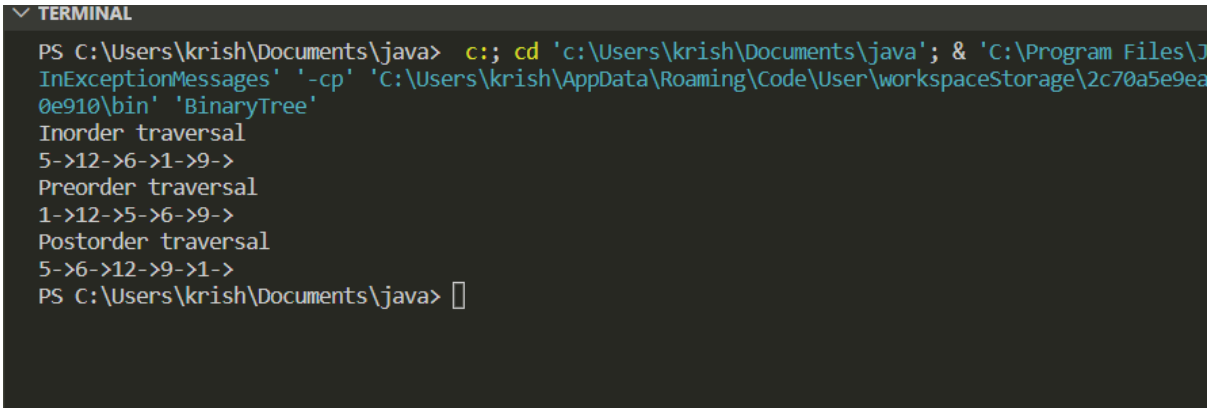
Radha Krishna garg

```
// Tree traversal in Java
class Node
{ int item;
Node left, right;
public Node(int key)
{ item = key;
left = right = null;
} }
class BinaryTree {
// Root of Binary Tree
Node root;
BinaryTree() {
root = null;
}
void postorder(Node node)
{ if (node == null)
return;
// Traverse left
postorder(node.left);
// Traverse right
postorder(node.right);
// Traverse root
System.out.print(node.item + "->");
}
void inorder(Node node)
{
if (node == null)
return;
// Traverse left
inorder(node.left);
// Traverse root
System.out.print(node.item + "->");
// Traverse right
inorder(node.right); }
void preorder(Node node)
{
if (node == null) return;
// Traverse root
System.out.print(node.item + "->");
// Traverse left
```

```
preorder(node.left);
// Traverse right
preorder(node.right); }
public static void main(String[] args) { BinaryTree tree = new BinaryTree();
     tree.root = new Node(1);
    tree.root.left = new Node(12);
    tree.root.right = new Node(9);

    tree.root.left.left = new Node(5);
    tree.root.left.right = new Node(6);
    System.out.println("Inorder traversal");
    tree.inorder(tree.root);
    System.out.println("\nPreorder traversal ");
    tree.preorder(tree.root);
    System.out.println("\nPostorder traversal");
    tree.postorder(tree.root);
} }
```

OUTPUT



Python

code

```
from asyncio import Queue


class Node:
    def __init__(self, item):
     self.left = None
     self.right = None
     self.val = item
# creating a tree data structure
def inorder(root):
#checking if the root is null or not
   if root:
     inorder(root.left)
# recursively calling left subtree
     print(str(root.val) + " ", end = '')
     inorder(root.right)
# recursively calling right subtree
def postorder(root):
    if root:
     postorder(root.left)
     postorder(root.right)
     print(str(root.val) + " ", end = '')
def preorder(root):
    if root:
     print(str(root.val) + " ", end = '')
     preorder(root.left)
     preorder(root.right)
def levelOrder(root):
     queue = list()
     queue.append(root)
while len(Queue):
    current = Queue[0]
    Queue = Queue[1: ]

print(str(current.val) + " ", end = "")
if current.left:
 Queue.append(current.left)
if current.right:
   Queue.append(current.right)
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
```

```
root.left.right = Node(5)
root.right.left = Node(6)
root.right.right = Node(7)
print("\nLevelOrder traversal:\t", end = " ")
levelOrder(root)
print("\nInorder traversal:\t", end = " ")
inorder(root)
print("\nPreorder traversal:\t", end = " ")
preorder(root)
print("\nPostorder traversal:\t", end = " ")
postorder(root)
```