**NAME        FATIMA AKBAR**

**ROLL NO      072**

**TASK NO      05**

**CLASS        BSDS(3A)**

**Lab 4 Tasks**

**DFS Traversals: Inorder, Preorder, Postorder**

**Purpose of Code:**

The purpose of this code is to demonstrate Depth First Search (DFS) traversals of a binary tree without using a Node class. Instead of object-oriented tree representation, a dictionary-based representation is used. The code implements three types of DFS traversals:  
1. Inorder (Left → Root → Right)  
2. Preorder (Root → Left → Right)  
3. Postorder (Left → Right → Root)

**Concepts Used:**

- Depth First Search (DFS): A graph/tree traversal technique that goes deep before exploring siblings.  
- Binary Tree Traversals: Visiting all nodes in specific orders (Inorder, Preorder, Postorder).  
- Recursion: Each traversal function calls itself for left and right children until base case (None).  
- Dictionary Representation: The binary tree is stored in a dictionary where keys are node labels and values are lists [LeftChild, RightChild].

# **Python Code**

tree = {  
 'A': ['B', 'C'],  
 'B': ['D', 'E'],  
 'C': [None, None],  
 'D': [None, None],  
 'E': [None, None]  
}  
  
def inorder(tree, node):  
 if node and node in tree:  
 inorder(tree, tree[node][0])  
 print(node, end=" ")  
 inorder(tree, tree[node][1])  
  
def preorder(tree, node):  
 if node and node in tree:  
 print(node, end=" ")  
 preorder(tree, tree[node][0])  
 preorder(tree, tree[node][1])  
  
def postorder(tree, node):  
 if node and node in tree:  
 postorder(tree, tree[node][0])  
 postorder(tree, tree[node][1])  
 print(node, end=" ")  
  
print("Inorder Traversal:")  
inorder(tree, 'A')  
  
print("\nPreorder Traversal:")  
preorder(tree, 'A')  
  
print("\nPostorder Traversal:")  
postorder(tree, 'A')