DFS with Stack & Tree Traversals (Python)

# 1. Node Class (N)

This class defines a simple binary tree node. It stores a value (v) and has left (l) and right (r) child pointers. Initially, the left and right children are set to None.

# 2. dfs(r)

This function performs a depth-first search (DFS) using an explicit stack. It starts from the root node 'r', and visits nodes in Preorder (Root-Left-Right) order. It returns a list of visited node values.

# 3. pre(r)

This is the Preorder traversal function implemented recursively. It visits Root → Left → Right.

# 4. ino(r)

This is the Inorder traversal function implemented recursively. It visits Left → Root → Right.

# 5. post(r)

This is the Postorder traversal function implemented recursively. It visits Left → Right → Root.

# 6. Tree Creation

We create a sample binary tree with nodes A, B, C, D, and E. A is the root, B is A's left child, C is A's right child, D is B's left child, and E is B's right child.

# 7. Function Calls & Output

We call dfs(root) to get the DFS order as a list.  
We call pre(root) to print Preorder traversal.  
We call ino(root) to print Inorder traversal.  
We call post(root) to print Postorder traversal.

The expected output is:  
DFS: ['A', 'B', 'D', 'E', 'C']  
Preorder: A B D E C  
Inorder: D B E A C  
Postorder: D E B C A