Aim:

To implement **tree traversals** (Inorder, Preorder, and Postorder) for a binary tree using recursion.

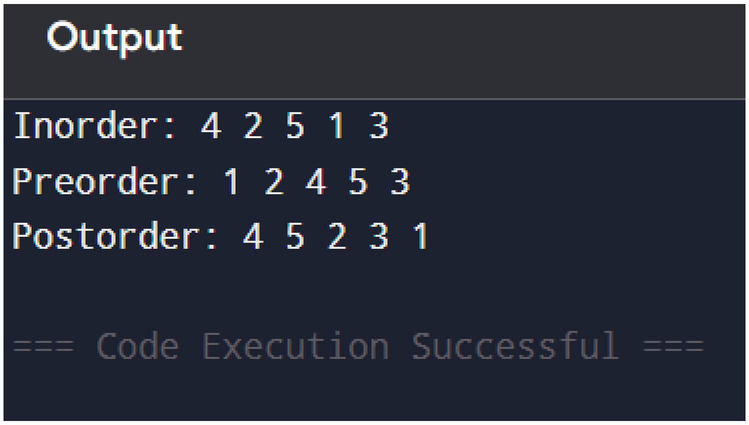
Algorithm:

1. Start
2. Create a binary tree node with `data`, `left`, and `right` pointers.
3. **Inorder Traversal:**
    - Traverse left subtree.
    - Visit root.
    - Traverse right subtree.
4. **Preorder Traversal:**
    - Visit root.
    - Traverse left subtree.
    - Traverse right subtree.
5. **Postorder Traversal:**
    - Traverse left subtree.
    - Traverse right subtree.
    - Visit root.
6. Stop

CODE:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node *left, *right;
};

struct Node* newNode(int val) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = val;
    node->left = node->right = NULL;
    return node;
}

void inorder(struct Node* root) {
    if (root == NULL) return;
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}

void preorder(struct Node* root) {
    if (root == NULL) return;
    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
}

void postorder(struct Node* root) {
    if (root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}
```

```
int main() {
    // Example tree
    struct Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    printf("Inorder: ");
    inorder(root);
    printf("\nPreorder: ");
    preorder(root);
    printf("\nPostorder: ");
    postorder(root);
    return 0;
}
```



```
Inorder: 4 2 5 1 3
Preorder: 1 2 4 5 3
Postorder: 4 5 2 3 1


=== Code Execution Successful ===
```

RESULT:

The program successfully executed and displayed the tree traversals for binary tree using recursion.