14. Tree Traversals

Aim:

To write a C program for tree traversals (Inorder, Preorder, Postorder).

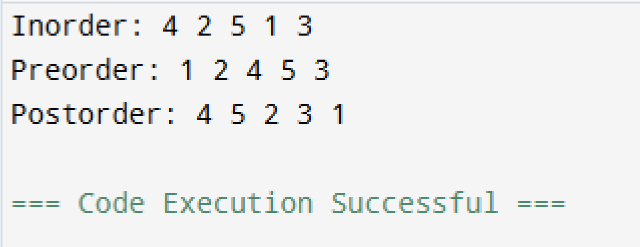
Algorithm:

1. Define binary tree structure.
2. Use recursion for traversals.
3. Inorder: Left → Root → Right.
4. Preorder: Root → Left → Right.
5. Postorder: Left → Right → Root.

Code:

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

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

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

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

int main() {
    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;
}
```

Sample Output:



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

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

Result:

Tree traversals were successfully implemented.