Tree Traversal

Write a C program to implement a Binary tree and perform the following tree traversal operation.

(i) Inorder Traversal

(ii) Preorder Traversal

(iii) Postorder Traversal

PROGRAM:

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

// Binary Tree Node structure
struct TreeNode {
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
};

// Function to create a new tree node
struct TreeNode* createNode(int data) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

// Function to perform Inorder traversal of the binary tree
void inorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
    }
}
```

```
// Function to perform Preorder traversal of the binary tree
void preorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorderTraversal(root->left);
        preorderTraversal(root->right);
    }
}

// Function to perform Postorder traversal of the binary tree
void postorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        postorderTraversal(root->left);
        postorderTraversal(root->right);
        printf("%d ", root->data);
    }
}

// Main function
int main() {
    // Constructing the binary tree
    struct TreeNode* root = createNode(1);
    root->left = createNode(2);
    root->right = createNode(3);
    root->left->left = createNode(4);
    root->left->right = createNode(5);

    printf("Inorder traversal: ");
    inorderTraversal(root);
    printf("\n");

    printf("Preorder traversal: ");
    preorderTraversal(root);
```

```
    printf("\n");

    printf("Postorder traversal: ");
    postorderTraversal(root);
    printf("\n");

    return 0;
}
```

OUTPUT:

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