

## EXP 14

## TREE TRAVERSAL

### Program

```
#include <stdio.h>

#include <stdlib.h>

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

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

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

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

```

}

void postorderTraversal(struct Node* root) {
    if (root == NULL)
        return;

    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%d ", root->data);
}

int main() {
    // Manually creating a simple binary tree:
    //      1
    //     /\
    //    2 3
    //   /\
    //  4 5

    struct Node* 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

The screenshot shows a C++ IDE with a code editor on the left and a terminal on the right. The code defines three traversal functions: `inorderTraversal`, `preorderTraversal`, and `postorderTraversal`, along with a `main` function that manually creates a binary tree with root 1, left child 2, and right child 3. The terminal displays the output of these traversals.

```

17         return;
18         inorderTraversal(root->left);
19         printf("%d ", root->data);
20         inorderTraversal(root->right);
21     }
22     void preorderTraversal(struct Node* root) {
23         if (root == NULL)
24             return;
25         printf("%d ", root->data);
26         preorderTraversal(root->left);
27         preorderTraversal(root->right);
28     }
29     void postorderTraversal(struct Node* root) {
30         if (root == NULL)
31             return;
32         postorderTraversal(root->left);
33         postorderTraversal(root->right);
34         printf("%d ", root->data);
35     }
36     int main() {
37         // Manually creating a simple binary tree:
38         //      1
39         //     /\
40         //    2 3

```

```

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

-----
Process exited after 1.947 seconds with return value 0
Press any key to continue . . .

```