EXPERIMENT [14](in order and post order)

CODE:

#include <stdio.h>

#include <stdlib.h>

// Structure for tree node

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

// Create a new node

struct Node\* createNode(int value) {

struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));

newNode->data = value;

newNode->left = newNode->right = NULL;

return newNode;

}

// Inorder Traversal: Left -> Root -> Right

void inorder(struct Node\* root) {

if (root != NULL) {

inorder(root->left);

printf("%d ", root->data);

inorder(root->right);

}

}

// Preorder Traversal: Root -> Left -> Right

void preorder(struct Node\* root) {

if (root != NULL) {

printf("%d ", root->data);

preorder(root->left);

preorder(root->right);

}

}

// Postorder Traversal: Left -> Right -> Root

void postorder(struct Node\* root) {

if (root != NULL) {

postorder(root->left);

postorder(root->right);

printf("%d ", root->data);

}

}

// Main function

int main() {

// Manually creating the binary tree

/\*

1

/ \

2 3

/ \ \

4 5 6

\*/

struct Node\* root = createNode(1);

root->left = createNode(2);

root->right = createNode(3);

root->left->left = createNode(4);

root->left->right = createNode(5);

root->right->right = createNode(6);

printf("Inorder Traversal: ");

inorder(root);

printf("\n");

printf("Preorder Traversal: ");

preorder(root);

printf("\n");

printf("Postorder Traversal: ");

postorder(root);

printf("\n");

return 0;

}

OUTPUT:

