#include <stdio.h>

#include <stdlib.h>

// Define the structure for a binary tree node

struct node {

int data;

struct node \*left;

struct node \*right;

};

// Function prototypes

void create(struct node \*\*);

struct node \*insert(struct node \*, int);

void inorder(struct node \*);

void preorder(struct node \*);

void postorder(struct node \*);

int main() {

int choice, x;

struct node \*tree = NULL; // Initialize the tree pointer

create(&tree);

do {

printf("\n\*\*\* --- Operations Available --- \*\*\*");

printf("\n1. Insert a Node");

printf("\n2. Display Inorder Traversal");

printf("\n3. Display Preorder Traversal");

printf("\n4. Display Postorder Traversal");

printf("\n5. Exit\n");

printf("Please enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("\nEnter the data to be inserted: ");

scanf("%d", &x);

tree = insert(tree, x); // Update the tree after insertion

break;

case 2:

printf("\nElements in the inorder traversal are: ");

inorder(tree);

printf("\n");

break;

case 3:

printf("\nElements in the preorder traversal are: ");

preorder(tree);

printf("\n");

break;

case 4:

printf("\nElements in the postorder traversal are: ");

postorder(tree);

printf("\n");

break;

case 5:

printf("Exit: Program Finished !!\n");

break;

default:

printf("\nPlease enter a valid option (1, 2, 3, 4, 5).\n");

break;

}

} while (choice != 5);

return 0;

}

// Function to create an empty binary tree

void create(struct node \*\*tree) {

\*tree = NULL;

}

// Function to insert a new node

struct node \*insert(struct node \*tree, int x) {

if (tree == NULL) {

struct node \*p = (struct node \*)malloc(sizeof(struct node));

if (p == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

p->data = x;

p->left = NULL;

p->right = NULL;

return p;

}

if (x < tree->data) {

tree->left = insert(tree->left, x);

} else {

tree->right = insert(tree->right, x);

}

return tree;

}

// Function for inorder traversal

void inorder(struct node \*tree) {

if (tree != NULL) {

inorder(tree->left);

printf("%d\t", tree->data);

inorder(tree->right);

}

}

// Function for preorder traversal

void preorder(struct node \*tree) {

if (tree != NULL) {

printf("%d\t", tree->data);

preorder(tree->left);

preorder(tree->right);

}

}

// Function for postorder traversal

void postorder(struct node \*tree) {

if (tree != NULL) {

postorder(tree->left);

postorder(tree->right);

printf("%d\t", tree->data);

}

}

