#include <stdio.h>

#include <stdlib.h>

// Define the structure for a tree node

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

// Function to create a new node

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;

}

// Function to insert a new node in the binary tree

struct Node\* insert(struct Node\* root, int data) {

if (root == NULL) {

return createNode(data);

}

if (data < root->data) {

root->left = insert(root->left, data);

} else {

root->right = insert(root->right, data);

}

return root;

}

// Function for in-order traversal

void inOrder(struct Node\* root) {

if (root != NULL) {

inOrder(root->left);

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

inOrder(root->right);

}

}

// Function for pre-order traversal

void preOrder(struct Node\* root) {

if (root != NULL) {

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

preOrder(root->left);

preOrder(root->right);

}

}

// Function for post-order traversal

void postOrder(struct Node\* root) {

if (root != NULL) {

postOrder(root->left);

postOrder(root->right);

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

}

}

// Function to search for a value in the binary tree

struct Node\* search(struct Node\* root, int data) {

if (root == NULL || root->data == data) {

return root;

}

if (data < root->data) {

return search(root->left, data);

}

return search(root->right, data);

}

int main() {

struct Node\* root = NULL;

int choice, value;

while (1) {

printf("\nBinary Tree Operations:\n");

printf("1. Insert\n");

printf("2. In-Order Traversal\n");

printf("3. Pre-Order Traversal\n");

printf("4. Post-Order Traversal\n");

printf("5. Search\n");

printf("6. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf("In-Order Traversal: ");

inOrder(root);

printf("\n");

break;

case 3:

printf("Pre-Order Traversal: ");

preOrder(root);

printf("\n");

break;

case 4:

printf("Post-Order Traversal: ");

postOrder(root);

printf("\n");

break;

case 5:

printf("Enter value to search: ");

scanf("%d", &value);

struct Node\* result = search(root, value);

if (result != NULL) {

printf("Value %d found in the tree.\n", value);

} else {

printf("Value %d not found in the tree.\n", value);

}

break;

case 6:

exit(0);

default:

printf("Invalid choice! Please try again.\n");

}

}

return 0;

}

OUTPUT

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 1

Enter value to insert: 10

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 1

Enter value to insert: 20

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 1

Enter value to insert: 30

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 1

Enter value to insert: 40

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 2

In-Order Traversal: 10 20 30 40

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 3

Pre-Order Traversal: 10 20 30 40

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 4

Post-Order Traversal: 40 30 20 10

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 5

Enter value to search: 20

Value 20 found in the tree.

Binary Tree Operations:

1. Insert

2. In-Order Traversal

3. Pre-Order Traversal

4. Post-Order Traversal

5. Search

6. Exit

Enter your choice: 6

=== Code Execution Successful ===