Write a C program to implement binary tree and demonstrate in-order traversals.

Code:

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

// Define the structure for a binary tree node
struct TreeNode {
    int data;
    struct TreeNode *left;
    struct TreeNode *right;
};

// Function to create a new 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 insert a node into the binary tree
struct TreeNode* insertNode(struct TreeNode* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }

    if (data < root->data) {
        root->left = insertNode(root->left, data);
    } else {
        root->right = insertNode(root->right, data);
    }

    return root;
}

// Function to perform in-order traversal
void inOrderTraversal(struct TreeNode* root) {
    if (root == NULL) {
        return;
    }

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

int main() {
    struct TreeNode* root = NULL;
    int num, value;
```

```
    printf("Enter the number of elements you want to insert in the binary tree: ");
    scanf("%d", &num);

    for (int i = 0; i < num; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &value);
        root = insertNode(root, value);
    }

    printf("\nIn-order traversal: ");
    inOrderTraversal(root);

    return 0;
}
```

Output:

```
Enter the number of elements you want to insert in the binary tree: 5
Enter element 1: 9
Enter element 2: 2
Enter element 3: 4
Enter element 4:
5
Enter element 5: 8
In-order traversal: 2 4 5 8 9
```

Write a C program to implement binary tree and demonstrate pre-order traversals.

Code:

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

// Define the structure for a binary tree node
struct TreeNode {
    int data;
    struct TreeNode *left;
    struct TreeNode *right;
};

// Function to create a new 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 insert a node into the binary tree
struct TreeNode* insertNode(struct TreeNode* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }

    if (data < root->data) {
        root->left = insertNode(root->left, data);
    } else {
        root->right = insertNode(root->right, data);
    }

    return root;
}

// Function to perform pre-order traversal
void preOrderTraversal(struct TreeNode* root) {
    if (root == NULL) {
        return;
    }

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

int main() {
    struct TreeNode* root = NULL;
    int num, value;
```

```
    printf("Enter the number of elements you want to insert in the binary tree: ");
    scanf("%d", &num);

    for (int i = 0; i < num; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &value);
        root = insertNode(root, value);
    }

    printf("\nPre-order traversal: ");
    preOrderTraversal(root);

    return 0;
}
```

Output:

```
Enter the number of elements you want to insert in the binary tree: 5
Enter element 1: 8
Enter element 2: 3
Enter element 3: 4
Enter element 4: 5
Enter element 5: 9
Pre-order traversal: 8 3 4 5 9
```

Write a C program to implement binary tree and demonstrate pre-order traversals.

Code:

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

// Define the structure for a binary tree node
struct TreeNode {
    int data;
    struct TreeNode *left;
    struct TreeNode *right;
};

// Function to create a new 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 insert a node into the binary tree
struct TreeNode* insertNode(struct TreeNode* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }

    if (data < root->data) {
        root->left = insertNode(root->left, data);
    } else {
        root->right = insertNode(root->right, data);
    }

    return root;
}

// Function to perform post-order traversal
void postOrderTraversal(struct TreeNode* root) {
    if (root == NULL) {
        return;
    }

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

int main() {
    struct TreeNode* root = NULL;
    int num, value;
```

```
    printf("Enter the number of elements you want to insert in the binary tree: ");
    scanf("%d", &num);

    for (int i = 0; i < num; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &value);
        root = insertNode(root, value);
    }

    printf("\nPost-order traversal: ");
    postOrderTraversal(root);

    return 0;
}
```

Output:

```
Enter the number of elements you want to insert in the binary tree: 5
Enter element 1: 2
Enter element 2: 9
Enter element 3: 5
Enter element 4: 6
Enter element 5: 3
Post-order traversal: 3 6 5 9 2
```

C program to search an element is tree data structure

Code:

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

// Define the structure for a binary tree node
struct TreeNode {
    int data;
    struct TreeNode *left;
    struct TreeNode *right;
};

// Function to create a new 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 insert a node into the binary search tree
struct TreeNode* insertNode(struct TreeNode* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }

    if (data < root->data) {
        root->left = insertNode(root->left, data);
    } else {
        root->right = insertNode(root->right, data);
    }

    return root;
}

// Function to search for a value in the binary search tree
struct TreeNode* searchNode(struct TreeNode* root, int key) {
    if (root == NULL || root->data == key) {
        return root;
    }

    if (key < root->data) {
        return searchNode(root->left, key);
    }

    return searchNode(root->right, key);
}

int main() {
    struct TreeNode* root = NULL;
```

```
    int num, value, searchValue;

    printf("Enter the number of elements you want to insert in the binary search tree: ");
    scanf("%d", &num);

    for (int i = 0; i < num; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &value);
        root = insertNode(root, value);
    }

    printf("Enter the value to search for: ");
    scanf("%d", &searchValue);

    struct TreeNode* result = searchNode(root, searchValue);
    if (result) {
        printf("Value %d found in the binary search tree.\n");
    } else {
        printf("Value %d not found in the binary search tree.\n");
    }

    return 0;
}
```

Output:

```
Enter the number of elements you want to insert in the binary search tree: 5
Enter element 1: 9
Enter element 2: 2
5Enter element 3:
5
Enter element 4: 6
Enter element 5: 3
Enter the value to search for: 5
Value 5 found in the binary search tree.
```