# Laboratory 9

1. Questions
2. Write a C program to construct a binary search tree and perform the Preorder, post order and in order traversal.
3. Write a C program to implement a linked list to construct a tree and count the number of leaves in a tree.
4. Algorithm

**2.1 C program to construct a binary search tree and perform the Preorder , postorder and inorder traversal.**

Step1: start

Step2: Inorder(tree)

1. Traverse the left subtree, i.e., call Inorder(left- subtree)

2. Visit the root.

3. Traverse the right subtree, i.e., call Inorder(right-subtree)

Step3: Preorder(tree)

1. Visit the root.

2. Traverse the left subtree, i.e., call Preorder(left-subtree)

3. Traverse the right subtree, i.e., call Preorder(right-subtree)

Step4: Postorder(tree)

1. Traverse the left subtree, i.e., call Postorder(left-subtree)

2. Traverse the right subtree, i.e., call Postorder(right-subtree)

3. Visit the root.

Step5: call the function accordingly in the main body

Step6: stop

**2.2 C program to implement a linked list to construct a tree and count the number of leaves in a tree.**

Step 1: start

Step 2: allocate node

Step 3: put in the data

Step 4: push data into a linked list

4.1 if (head == NULL):

4.2 head = temp;

4.3 head->next = NULL

4.4 else: temp->next= head and head = temp

Step 5: print the linked list

Step 6: allocate a newtreenode

6.1 temp ->info=value

6.2 emp->count = 0;

6.3 temp->left= temp->right = NULL;

Step 7: insert function

7.1 if root=NULL: return newtreenode(key)

7.2 if root->left = NULL: root->left = newTreeNode(key);

7.3 if root->right = NULL: root->right = newTreeNode(key);

7.4 if (temp->count! =2): insert(root->left,key);

else {temp = root->right;

7.4.1 if (temp->count! =2)

insert(root->right,key);

7.4.2 else

insert(root->left,key);

7.5 return root

Step 8: Step 8: for getting leaf count

8.1 if (root == NULL): return 0;

8.2 if(root->left = NULL && root->right = NULL): return 1;

8.3 else

Return LeafCount(root->left)+LeafCount(root->right);

Step 9: stop

1. Program

Figure C program to construct a binary search tree and perform the Preorder , postorder and inorder traversal

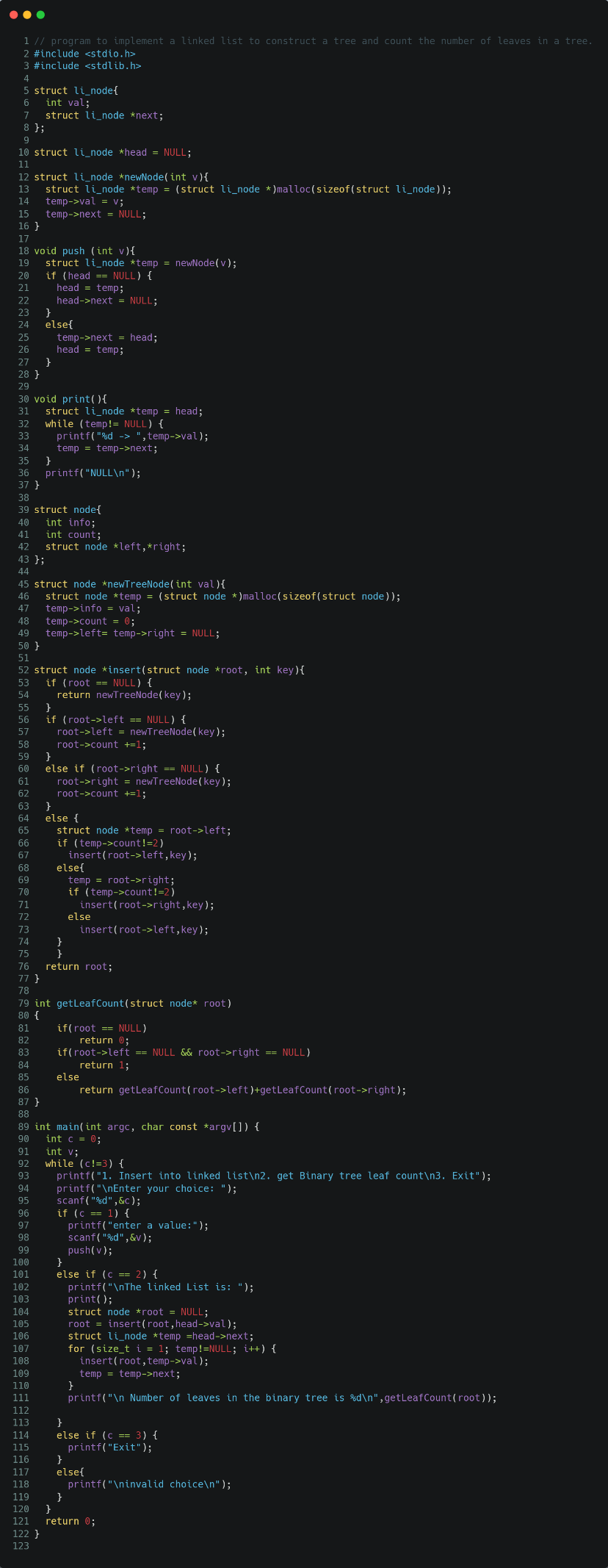


Figure C program to implement a linked list to construct a tree and count the number of leaves in a tree.

1. Presentation of Results

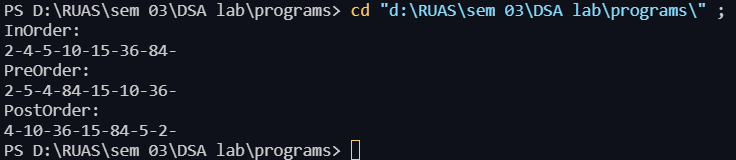


Figure output of program to construct a binary search tree and perform the Preorder , postorder and inorder traversal

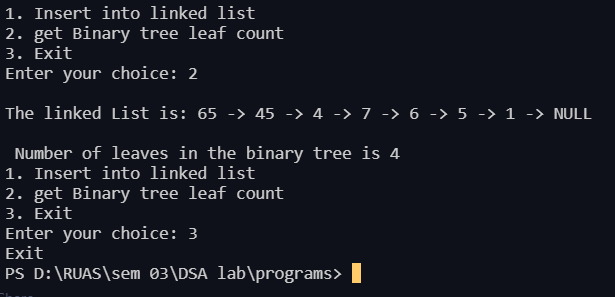


Figure output of program to implement a linked list to construct a tree and count the number of leaves in a tree.

1. Conclusions

Learning happened:

**Trees** are hierarchical data structures.

The topmost node is called **root** of the tree. The elements that are directly under an element are called its **children**. The element directly above something is called its parent.

**Binary Tree**: A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

Tree traversal Techniques are-

* **Inorder (Left, Root, Right)**
* **Preorder (Root, Left, Right)**
* **Postorder (Left, Right, Root)**

**Binary Search Tree**, is a node-based binary tree data structure which has the following properties:

* The left subtree of a node contains only nodes with keys lesser than the node’s key.
* The right subtree of a node contains only nodes with keys greater than the node’s key.
* The left and right subtree each must also be a binary search tree.