Laboratory 9

a. Questions

a) Write a C program to construct a binary search tree and perform the Preorder, post order and in order traversal.
b) Write a C program to implement a linked list to construct a tree and count the number of leaves in a tree.

b. Algorithm

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

```
Step1: start

Step2:to construct a binary search tree
      2.1 Start from root.
      2.2 Compare the inserting element with root, if less than
      root, then recurse for left, else recurse for right.
      2.3 After reaching end, just insert that node at left (if
      less than current) else right.

Step3: 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)

Step4: 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)

Step5: 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.

Step6: call the function accordingly in the main body

Step7: 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
```

c. Program

```c
// program to construct a binary search tree and perform the Preorder, post order and in order traversal
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>

struct node{
  int info;
  struct node *left,*right;
};

struct node *newTreeNode(int val){
  struct node *temp = (struct node *)malloc(sizeof(struct node));
  temp->info = val;
  temp->left= temp->right = NULL;
}

struct node *insert(struct node *root, int key){
  if (root == NULL) {
    return newTreeNode(key);
  }
  if (key<root->info) {
    root->left = insert(root->left,key);
  } else if (key>root->info) {
    root->right = insert(root->right,key);
  }
  return root;
}

void inorder(struct node *root){
  if (root->left != NULL) {
    inorder(root->left);
  }
  printf("%d-",root->info);
  if (root->right != NULL) {
    inorder(root->right);
  }
}

void postorder(struct node *root){
  if (root == NULL) return;
  postorder(root->left);
  postorder(root->right);
  printf("%d-",root->info);
}

void preorder(struct node *root){
  if (root == NULL) return;
  printf("%d-",root->info);
  preorder(root->left);
  preorder(root->right);
}

int main(int argc, char const *argv[]) {
  int a[] = {2,5,84,15,36,4,10};
  struct node *root = NULL;
  root = insert(root,a[0]);
  for (size_t i = 1; i < 7; i++) {
    insert(root,a[i]);
  }
  printf("InOrder:\n");
  inorder(root);

  printf("\nPreOrder:\n");
  preorder(root);

  printf("\nPostOrder:\n");
  postorder(root);
  return 0;
}

```

Figure 1 C program to construct a binary search tree and perform the Preorder , postorder and inorder traversal

```c
// program to implement a linked list to construct a tree and count the number of leaves in a tree.
#include <stdio.h>
#include <stdlib.h>

struct li_node{
  int val;
  struct li_node *next;
};

struct li_node *head = NULL;

struct li_node *newNode(int v){
  struct li_node *temp = (struct li_node *)malloc(sizeof(struct li_node));
  temp->val = v;
  temp->next = NULL;
}

void push (int v){
  struct li_node *temp = newNode(v);
  if (head == NULL) {
    head = temp;
    head->next = NULL;
  }
  else{
    temp->next = head;
    head = temp;
  }
}

void print(){
  struct li_node *temp = head;
  while (temp!= NULL) {
    printf("%d -> ",temp->val);
    temp = temp->next;
  }
  printf("NULL\n");
}

struct node{
  int info;
  int count;
  struct node *left,*right;
};

struct node *newTreeNode(int val){
  struct node *temp = (struct node *)malloc(sizeof(struct node));
  temp->info = val;
  temp->count = 0;
  temp->left= temp->right = NULL;
}

struct node *insert(struct node *root, int key){
  if (root == NULL) {
    return newTreeNode(key);
  }
  if (root->left == NULL) {
    root->left = newTreeNode(key);
    root->count +=1;
  }
  else if (root->right == NULL) {
    root->right = newTreeNode(key);
    root->count +=1;
  }
  else {
    struct node *temp = root->left;
    if (temp->count!=2)
      insert(root->left,key);
    else{
      temp = root->right;
      if (temp->count!=2)
        insert(root->right,key);
      else
        insert(root->left,key);
    }
    }
  return root;
}

int getLeafCount(struct node* root)
{
    if(root == NULL)
        return 0;
    if(root->left == NULL && root->right == NULL)
        return 1;
    else
        return getLeafCount(root->left)+getLeafCount(root->right);
}

int main(int argc, char const *argv[]) {
  int c = 0;
  int v;
  while (c!=3) {
    printf("1. Insert into linked list\n2. get Binary tree leaf count\n3. Exit");
    printf("\nEnter your choice: ");
    scanf("%d",&c);
    if (c == 1) {
      printf("enter a value:");
      scanf("%d",&v);
      push(v);
    }
    else if (c == 2) {
      printf("\nThe linked List is: ");
      print();
      struct node *root = NULL;
      root = insert(root,head->val);
      struct li_node *temp =head->next;
      for (size_t i = 1; temp!=NULL; i++) {
        insert(root,temp->val);
        temp = temp->next;
      }
      printf("\n Number of leaves in the binary tree is %d\n",getLeafCount(root));

    }
    else if (c == 3) {
      printf("Exit");
    }
    else{
      printf("\ninvalid choice\n");
    }
  }
  return 0;
}

```

Figure 2 C program to implement a linked list to construct a tree and count the number of leaves in a tree.

d. Presentation of Results

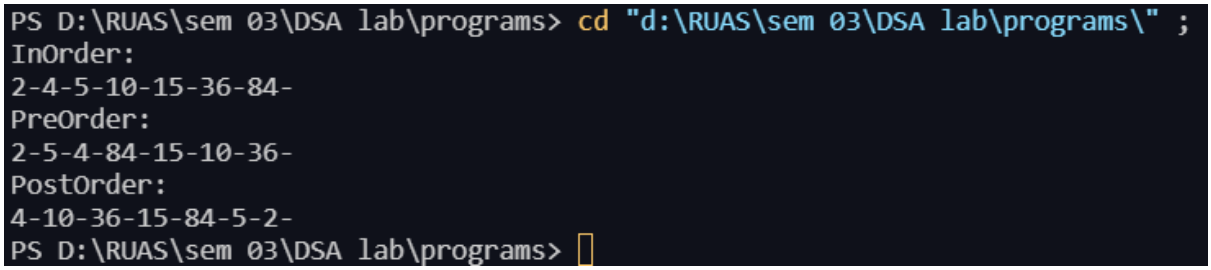

Figure 3 output of program to construct a binary search tree and perform the Preorder , postorder and inorder traversal

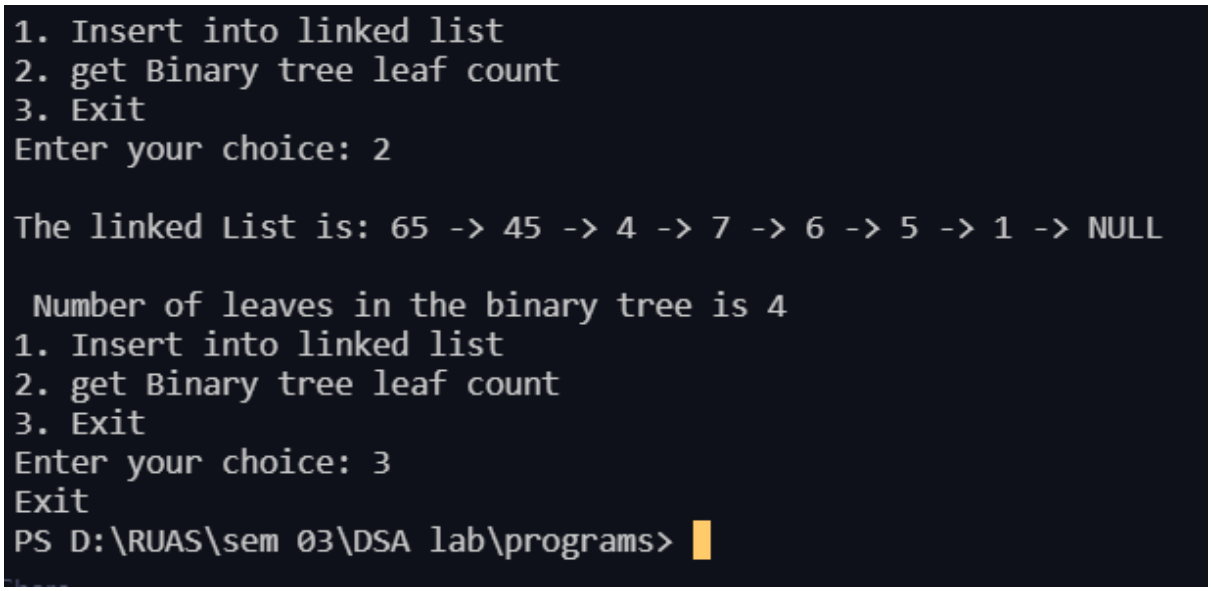

Figure 4 output of program to implement a linked list to construct a tree and count the number of leaves in a tree.

e. 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.