Laboratory 9

1. **Questions**

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

2. **Algorithm**

    1.**Algorithm for program to construct a binary search tree and perform the Preorder, post order and in order traversal.**

Inorder Traversal

Algorithm 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)

Preorder Traversal

Algorithm 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)

Postorder Traversal

Algorithm 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.

2.**Algorithm to implement a linked list to construct a tree and count the number of leaves in a tree.**

Algorithm to get the leaf node count.

getLeafCount(node)

1) If node is NULL then return 0.

2) Else If left and right child nodes are NULL return 1.

3) Else recursively calculate leaf count of the tree using below formula.

Leaf count of a tree = Leaf count of left subtree +

Leaf count of right subtree

3. Program

```c
// C program for different tree traversals
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Helper function that allocates a new node with the
given data and NULL left and right pointers. */
struct node* newNode(int data)
{
    struct node* node = (struct node*)
                        malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}

/* Given a binary tree, print its nodes according to the
"bottom-up" postorder traversal. */
void printPostorder(struct node* node)
{
    if (node == NULL)
        return;

    // first recur on left subtree
    printPostorder(node->left);

    // then recur on right subtree
    printPostorder(node->right);

    // now deal with the node
    printf("%d ", node->data);
}

/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node)
{
    if (node == NULL)
        return;

    /* first recur on left child */
    printInorder(node->left);

    /* then print the data of node */
    printf("%d ", node->data);

    /* now recur on right child */
    printInorder(node->right);
}

/* Given a binary tree, print its nodes in preorder*/
```

```c
void printPreorder(struct node* node)
{
    if (node == NULL)
        return;

    /* first print data of node */
    printf("%d ", node->data);

    /* then recur on left sutree */
    printPreorder(node->left);

    /* now recur on right subtree */
    printPreorder(node->right);
}

/* Driver program to test above functions*/
int main()
{
    struct node *root = newNode(1);
    root->left            = newNode(2);
    root->right       = newNode(3);
    root->left->left      = newNode(4);
    root->left->right = newNode(5);

    printf("\nPreorder traversal of binary tree is \n");
    printPreorder(root);

    printf("\nInorder traversal of binary tree is \n");
    printInorder(root);

    printf("\nPostorder traversal of binary tree is \n");
    printPostorder(root);

    getchar();
    return 0;
}
```

Fig 1 program to construct a binary search tree and perform the Preorder, post order and in order traversal.

```c
// C implementation to find leaf count of a given Binary tree
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Function to get the count of leaf nodes in a binary tree*/
unsigned int getLeafCount(struct node* node)
{
    if(node == NULL)
        return 0;
    if(node->left == NULL && node->right==NULL)
        return 1;
    else
        return getLeafCount(node->left)+
               getLeafCount(node->right);
}

/* Helper function that allocates a new node with the
given data and NULL left and right pointers. */
struct node* newNode(int data)
{
    struct node* node = (struct node*)
                malloc(sizeof(struct node));
    node->data = data;
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}

/*Driver program to test above functions*/
int main()
{
/*create a tree*/
    struct node *root = newNode(1);
    root->left    = newNode(2);
    root->right   = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

/*get leaf count of the above created tree*/
    printf("Leaf count of the tree is %d", getLeafCount(root));

    getchar();
    return 0;
}
```

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

4. Presentation of Result

```
Preorder traversal of binary tree is
1 2 4 5 3
Inorder traversal of binary tree is
4 2 5 1 3
Postorder traversal of binary tree is
4 5 2 3 1
```

Fig 3 Result of program to construct a binary search tree and perform the Preorder, post order and in order traversal.

```
Leaf count of the tree is 3
```

Fig 4 Result of program to construct a binary search tree and perform the Preorder, post order and in order traversal.

5. Conclusion

In this lab we learnt to Write a C program to construct a binary search tree and perform the Preorder, post order and in order traversal and to Write a C program to implement a linked list to construct a tree and count the number of leaves in a tree.