

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: 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
```

c. Program

```
1 // program to construct a binary search tree and perform the Preorder, post order and in order traversal
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <stdlib.h>
5
6 struct node{
7     int info;
8     struct node *left,*right;
9 };
10
11 struct node *newTreeNode(int val){
12     struct node *temp = (struct node *)malloc(sizeof(struct node));
13     temp->info = val;
14     temp->left = temp->right = NULL;
15 }
16
17 struct node *insert(struct node *root, int key){
18     if (root == NULL) {
19         return newTreeNode(key);
20     }
21     if (key < root->info) {
22         root->left = insert(root->left, key);
23     } else if (key > root->info) {
24         root->right = insert(root->right, key);
25     }
26     return root;
27 }
28
29 void inorder(struct node *root){
30     if (root->left != NULL) {
31         inorder(root->left);
32     }
33     printf("%d-", root->info);
34     if (root->right != NULL) {
35         inorder(root->right);
36     }
37 }
38
39 void postorder(struct node *root){
40     if (root == NULL) return;
41     postorder(root->left);
42     postorder(root->right);
43     printf("%d-", root->info);
44 }
45
46 void preorder(struct node *root){
47     if (root == NULL) return;
48     printf("%d-", root->info);
49     preorder(root->left);
50     preorder(root->right);
51 }
52
53 int main(int argc, char const *argv[]) {
54     int a[] = {2,5,84,15,36,4,10};
55     struct node *root = NULL;
56     root = insert(root, a[0]);
57     for (size_t i = 1; i < 7; i++) {
58         insert(root, a[i]);
59     }
60     printf("InOrder:\n");
61     inorder(root);
62
63     printf("\nPreOrder:\n");
64     preorder(root);
65
66     printf("\nPostOrder:\n");
67     postorder(root);
68     return 0;
69 }
70
```

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

```

1 // program to implement a linked list to construct a tree and count the number of leaves in a tree.
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 struct li_node{
6     int val;
7     struct li_node *next;
8 };
9
10 struct li_node *head = NULL;
11
12 struct li_node *newNode(int v){
13     struct li_node *temp = (struct li_node *)malloc(sizeof(struct li_node));
14     temp->val = v;
15     temp->next = NULL;
16 }
17
18 void push (int v){
19     struct li_node *temp = newNode(v);
20     if (head == NULL) {
21         head = temp;
22         head->next = NULL;
23     }
24     else{
25         temp->next = head;
26         head = temp;
27     }
28 }
29
30 void print(){
31     struct li_node *temp = head;
32     while (temp != NULL) {
33         printf("%d -> ",temp->val);
34         temp = temp->next;
35     }
36     printf("NULL\n");
37 }
38
39 struct node{
40     int info;
41     int count;
42     struct node *left,*right;
43 };
44
45 struct node *newTreeNode(int val){
46     struct node *temp = (struct node *)malloc(sizeof(struct node));
47     temp->info = val;
48     temp->count = 0;
49     temp->left = temp->right = NULL;
50 }
51
52 struct node *insert(struct node *root, int key){
53     if (root == NULL) {
54         return newTreeNode(key);
55     }
56     if (root->left == NULL) {
57         root->left = newTreeNode(key);
58         root->count++;
59     }
60     else if (root->right == NULL) {
61         root->right = newTreeNode(key);
62         root->count++;
63     }
64     else {
65         struct node *temp = root->left;
66         if (temp->count!=2)
67             insert(root->left,key);
68         else{
69             temp = root->right;
70             if (temp->count!=2)
71                 insert(root->right,key);
72             else
73                 insert(root->left,key);
74         }
75     }
76     return root;
77 }
78
79 int getLeafCount(struct node* root)
80 {
81     if(root == NULL)
82         return 0;
83     if(root->left == NULL && root->right == NULL)
84         return 1;
85     else
86         return getLeafCount(root->left)+getLeafCount(root->right);
87 }
88
89 int main(int argc, char const *argv[]) {
90     int c = 0;
91     int v;
92     while (c!=3) {
93         printf("1. Insert into linked list\n2. get Binary tree leaf count\n3. Exit");
94         printf("\nEnter your choice: ");
95         scanf("%d",&c);
96         if (c == 1) {
97             printf("Enter a value:");
98             scanf("%d",&v);
99             push(v);
100         }
101         else if (c == 2) {
102             printf("\nThe linked List is: ");
103             print();
104             struct node *root = NULL;
105             root = insert(root,head->val);
106             struct li_node *temp = head->next;
107             for (size_t i = 1; temp!=NULL; i++) {
108                 insert(root,temp->val);
109                 temp = temp->next;
110             }
111             printf("\n Number of leaves in the binary tree is %d\n",getLeafCount(root));
112         }
113         else if (c == 3) {
114             printf("Exit");
115         }
116         else{
117             printf("\ninvalid choice\n");
118         }
119     }
120 }
121 return 0;
122 }
123

```

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

```
PS D:\RUAS\sem 03\DSA lab\programs> cd "d:\RUAS\sem 03\DSA lab\programs\" ;  
InOrder:  
2-4-5-10-15-36-84-  
PreOrder:  
2-5-4-84-15-10-36-  
PostOrder:  
4-10-36-15-84-5-2-  
PS D:\RUAS\sem 03\DSA lab\programs> █
```

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

```
1. Insert into linked list  
2. get Binary tree leaf count  
3. Exit  
Enter your choice: 2  
  
The linked List is: 65 -> 45 -> 4 -> 7 -> 6 -> 5 -> 1 -> NULL  
  
Number of leaves in the binary tree is 4  
1. Insert into linked list  
2. get Binary tree leaf count  
3. Exit  
Enter your choice: 3  
Exit  
PS D:\RUAS\sem 03\DSA lab\programs> █
```

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.