# COMPUTER LAB

**NAME:** SOUMYADIP MAITY

**ROLL NO.:** 22053029

**SEC:** CSE-49

**YEAR:** 2023-24

# *ASSIGNMENT – 10*

## 1. Write the following menu driven program for the binary search tree

```
----------------------------------------
Binary Search Tree Menu
----------------------------------------
0. create
1. Count number of leaf nodes in the tree
2. count number of non-leaf nodes in the tree.
3. find number of nodes in the tree.
4. find sum of all nodes of the tree.
5. print height and depth of the tree.
6. find nodes which are at maximum depth in the tree.
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
int data;
struct node * left,*right;
}node;
node* root=NULL;
node* insert(node*,int);
int countLeaf(node*);
int countNonLeaf(node*);
int countNode(node*);
int sumNode(node*);
void Height(node*);
void maxDepth(node*,int,int);
node* insert(node* root, int data)
{
node* new = malloc(sizeof(node));
new->data = data;
new->left = new->right = NULL;
if (root == NULL)
{
return new;
}
if (root->data < data)
{
root->right = insert(root->right, data);
}
if (root->data > data)
{
root->left = insert(root->left, data);
}
return root;
}
void inorder(node* root)
{
if(root!=NULL)
{
inorder(root->left);
printf("%d\t",root->data);
```

```
inorder(root->right);
}
}
int countLeaf(node* root){
if(root==NULL)
return 0;
if(root->left==NULL&&root->right==NULL)
return 1;
return countLeaf(root->left)+countLeaf(root->right);
}
int countNonleaf(node* root)
{
// Base cases.
if (root == NULL || (root->left == NULL && root->right == NULL))
return 0;// If root is Not NULL and its one of its
// child is also not NULL
return 1 + countNonleaf(root->left) + countNonleaf(root->right);
}
int countNode(node *root)
{
return countNonleaf(root)+countLeaf(root);
}
int sumNode(node* root)
{
if(root==NULL)
return 0;
return(root->data+sumNode(root->left)+sumNode(root->right));
}
int height(node* root)
{
if(root==NULL)
{
return 0;
}
else
{
int ldepth=height(root->left);
int rdepth=height(root->right);
return (ldepth>rdepth)?ldepth+1:rdepth+1;
}
}
void maxDepth(node* root,int level,int depth)
{
if(root==NULL)
return;
else
{
if(level==depth)
{
printf("%d\t",root->data);
}
```

```
maxDepth(root->left,level+1,depth);
maxDepth(root->right,level+1,depth);
}
}
int main()
{
int value,choice;
while(1)
{
printf("1. enter element in tree\n");
printf("2. Count total number of leaf nodes\n");
printf("3. Count no of non leaf nodes\n");
printf("4. count total number of nodes in tree\n");
printf("5. sum of all nodes\n");
printf("6. print height of binary tree\n");
printf("7. print all nodes at max depth of the tree\n");
printf("8. Exit\n");
scanf("%d",&choice);
switch (choice) {
case 1:
printf("Enter element\n");
scanf("%d",&value);
root=insert(root,value);
break;
case 4:
printf("No of nodes = %d\n",countNode(root));
break;
case 2:
printf("No leaf nodes = %d\n",countLeaf(root));
break;
case 3:
printf("Number of non leaf nodes = %d\n",countNonleaf(root));
break;case 5:
printf("Sum of all nodes = %d\n",sumNode(root));
break;
case 6:
printf("Height of tree = %d\n",height(root));
break;
case 7:
maxDepth(root,0,height(root)-1);
printf("\n");
break;
case 8:
exit(0);
break;
default:
printf("Wrong choice\n");
}
}
return 0;
}
```