Assignment No-5

Name- PRAFUL SHEWALE
Roll no=SI- 48 Class=SE-IT

Aim- BINARY SEARCH TREE

```
#include<iostream>
using namespace std;
struct node
{
node *left;
int data;
node *right;
};
class BST
{
node *root;
public :
BST()
{
root=NULL;
}
void createBST(node *root);
void displayPreorder(node *root);
void search(node *root);
int depth(node *root);
void findleaf(node *root);
};
int main()
{
int choice, value;
int d;
BST b;
node *root;
root=new node;
```

```
cout<<"Enter value for root node";
cin>>value;
root->data=value;
root->left=NULL;
root->right=NULL;
do
{
cout<<"\nSelect any one opration from : \n 1.CreateBST \n
2.DisplayPreorder \n 3.Search key \n 4.Find Depth \n 5.Find leaf nodes
\n 6.Exit\n";
cin>>choice;
switch(choice)
{
case 1: b.createBST(root);
break;
case 2: cout<<"Display Preorder Output:";
b.displayPreorder(root);
break;
case 3: b.search(root);
break;
case 4: d=b.depth(root);
cout<<"Depth of BST="<<d-1;
break;
case 5: b.findleaf(root);
break;
case 6:cout<<"EXIT";
break;
default: cout<<"Wrong choice";
}
}while(choice !=6);
return 0;
}
void BST :: createBST(node *root)
{
node *newnode, *temp;
char op;
do{
newnode=new node;
cout<<"Enter data for newnode=>";
cin>>newnode->data;
```

```
newnode->left=NULL;
newnode->right=NULL;
temp=root;
while(1)
{
if(newnode->data < temp->data)
{
if(temp->left==NULL)
{
temp->left=newnode;
break;
}
else
temp=temp->left;
}
else
{
if(newnode->data>temp->data)
{
if(temp->right==NULL)
{
temp->right=newnode;
break;
}
else
temp=temp->right;
}
}
}
cout<<"Do u want to create another newnode?press y or n \n";
cin>>op;
}while(op=='y');
}
void BST :: displayPreorder(node *root)
{
node *temp;
temp=root;
if(temp!=NULL)
{
cout<<temp->data<<"\t";
```

```
displayPreorder(temp->left);
displayPreorder(temp->right);
}
}
void BST :: search(node *root)
{
int key;
int flag=0;
node *temp;
cout<<"Enter value to be searched in BST\n";
cin>>key;
temp=root;
while(temp!=NULL)
{
if(key==temp->data)
{
flag=1;
break;
}
else
if(key < temp->data)
temp=temp->left;
else
temp=temp->right;
}
if(flag==1)
cout<<"Key value found in BST\n";
else
cout<<"Key value NOT FOUND in BST\n";
}
int BST :: depth(node *root)
{
int Ldepth, Rdepth;
if(root==NULL)
{
return 0;
}
Ldepth=depth(root->left);
Rdepth=depth(root->right);
if(Ldepth>Rdepth
```

```
return Ldepth+1;
else
return Rdepth+1;
}
void BST :: findleaf(node *root)
{
node *temp;
temp=root;
if(temp !=NULL)
{
if(temp->left==NULL && temp->right==NULL)
{
cout<<"Leaf Node="<<temp->data<<"\n";
}
findleaf(temp->left);
findleaf(temp->right);
}
}
```

OUTPUT

```
Enter value for root node: 10

Select any one operation from :
1. Create BST
2. Display Preorder
3. Search key
4. Find Depth
5. Find leaf nodes
6. Exit
1
Enter data for new node: 40
Do you want to create another new node? (y/n): y
Enter data for new node: 50
Do you want to create another new node? (y/n): y
Enter data for new node: 100
Do you want to create another new node? (y/n): n
```

Select any one operation from :
1. Create BST
2. Display Preorder
3. Search key
4. Find Depth
5. Find leaf nodes
6. Exit
2
Display Preorder Output: 10 40 50 100
Select any one operation from :
1. Create BST
2. Display Preorder
3. Search key
4. Find Depth
5. Find leaf nodes
6. Exit
4
Depth of BST = 3
Select any one operation from :
1. Create BST
2. Display Preorder
3. Search key
4. Find Depth
5. Find leaf nodes
6. Exit
5
Leaf Node = 100

Select any one operation from :
1. Create BST
2. Display Preorder
3. Search key
4. Find Depth
5. Find leaf nodes
6. Exit
3
Enter value to be searched in BST: 100
Key value found in BST

Select any one operation from :

1. Create BST
2. Display Preorder
3. Search key
4. Find Depth
5. Find leaf nodes
6. Exit
6.
EXIT

=== Code Execution Successful ===