Source Code-

```
#include <iostream>
#include <algorithm>
using namespace std;

struct node
{
    int data;
    node *left;
    node *right;
};

struct node* createNode(int data){
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

struct node* insert(node* Node,int data){
    if(Node == NULL){
        return createNode(data);
    }
    if(data < Node->data){
        Node->left = insert(Node->left,data);
        cout<<"Inserted "<<data<<" to the left of "<<Node->data<<endl;
    }else if(data > Node->data){
        Node->right = insert(Node->right,data);
        cout<<"Inserted "<<data<<" to the right of "<<Node->data<<endl;
    }else if(data==Node->data){
        cout<<"Value already present"<<endl;
        return Node;
    }
    return Node;
}

void search(node* Node,int data){
    if(Node == NULL){
        cout<<"Value not found"<<endl;
    }else if(data < Node->data){
        return search(Node->left,data);
    }else if(data > Node->data){
        return search(Node->right,data);
    }else if(data == Node->data){
        cout<<"Value found"<<endl;
    }

}

void displayLeafNodes(node* Node){
    if(Node == NULL){
        return;
    }else if(Node->left == NULL && Node->right == NULL){
```

```
      cout<<Node->data<<" ";
    }
    displayLeafNodes(Node->left);
    displayLeafNodes(Node->right);
}

void displayParentChild(node* Node){
    if(Node->left != NULL && Node->right != NULL){
        cout<<"Parent is "<<Node->data<<" and children are
"<<(Node->left->data)<<" and "<<(Node->right->data)<<endl;
        displayParentChild(Node->left);
        displayParentChild(Node->right);
    }else if(Node->left != NULL && Node->right == NULL){
        cout<< "Parent is "<<(Node->data)<<" and child is
"<<(Node->left->data)<<endl;
        displayParentChild(Node->left);
    }else if(Node->left == NULL && Node->right != NULL){
        cout<<"Parent is "<<(Node->data)+" and child is
"<<(Node->right->data)<<"\n";
        displayParentChild(Node->right);
    }else{
        return;
    }


}

int displayDepth(node* Node){
    if(Node == NULL){
        return 0;
    }
    int leftDepth = displayDepth(Node->left);
    int rightDepth = displayDepth(Node->right);
    if(leftDepth > rightDepth){
        return leftDepth+1;
    }else{
        return rightDepth+1;
    }
}

struct node* deleteNode(node* Node,int data){
    if(Node == NULL){
        return NULL;
    }
    if(data < Node->data){
        Node->left = deleteNode(Node->left,data);
    }else if(data > Node->data){
        Node->right = deleteNode(Node->right,data);
    }else{
        if(Node->left == NULL && Node->right == NULL){
            free(Node);
            return NULL;
        }else if(Node->left == NULL || Node->right == NULL){
            struct node* temp;
```

```
            if(Node->left == NULL){
                temp = Node->right;
            }else{
                temp = Node->left;
            }
            free(Node);
            return temp;
        }else{
            struct node* temp = Node->right;
            while(temp->left != NULL){
                temp = temp->left;
            }
            Node->data = temp->data;
            Node->right = deleteNode(Node->right,temp->data);
        }
    }
    return Node;
}

void inorder(node* Node){
    if(Node == NULL){
        return;
    }
    inorder(Node->left);
    cout<<Node->data<<" ";
    inorder(Node->right);
}

int main(){
      cout<<"Enter data for root node: "<<endl;
      int data;
      cin>>data;
      string choice;
      int option;
      struct node* root = createNode(data);
      cout<<"Please enter choice of operation: "<<endl;
      cout<<"1. Insert data"<<endl;
      cout<<"2. Delete data"<<endl;
      cout<<"3. Search data"<<endl;
      cout<<"4. Display tree"<<endl;
      cout<<"5. Display leaf nodes"<<endl;
      cout<<"6. Display parent and child nodes"<<endl;
      cout<<"7. Display depth of tree"<<endl;
            do{

          cout<<"Enter option: "<<endl;
          cin>>option;
          switch(option){
              case 1:
                cout<<"Enter data for new node: "<<endl;
                cin>>data;
                insert(root,data);
                break;
              case 2:
```

```
                cout<<"Enter data to delete: "<<endl;
                cin>>data;
                deleteNode(root,data);
                break;
              case 3:
                cout<<"Enter data to search: "<<endl;
                cin>>data;
                search(root,data);
                break;
              case 4:
                inorder(root);
                break;
              case 5:
                cout<<"Leaf nodes are: "<<endl;
                displayLeafNodes(root);
                break;
              case 6:
                displayParentChild(root);
                break;
              case 7:
                cout<<"Depth of tree is: "<<displayDepth(root)<<endl;
                break;
              default:
                cout<<"Invalid option"<<endl;
                break;
          }
          cout<<"Do you want to continue? (yes/no)"<<endl;
          cin>>choice;
      }while(choice == "yes");



    return 0;
}
```

Output-

```
Enter data for root node:
50
Please enter choice of operation:
1. Insert data
2. Delete data
3. Search data
4. Display tree
5. Display leaf nodes
6. Display parent and child nodes
7. Display depth of tree
Enter option:
1
Enter data for new node:
25
Inserted 25 to the left of 50
Do you want to continue? (yes/no)
yes
Enter option:
1
```

```
Enter data for new node:
75
Inserted 75 to the right of 50
Do you want to continue? (yes/no)
yes
Enter option:
1
Enter data for new node:
50
Value already present
Do you want to continue? (yes/no)
yes
Enter option:
1
Enter data for new node:
20
Inserted 20 to the left of 25
Inserted 20 to the left of 50
Do you want to continue? (yes/no)
yes
Enter option:
1
Enter data for new node:
40
Inserted 40 to the right of 25
Inserted 40 to the left of 50
Do you want to continue? (yes/no)
yes
Enter option:
1
Enter data for new node:
60
Inserted 60 to the left of 75
Inserted 60 to the right of 50
Do you want to continue? (yes/no)
yes
Enter option:
1
Enter data for new node:
80
Inserted 80 to the right of 75
Inserted 80 to the right of 50
Do you want to continue? (yes/no)
yes
Enter option:
2
Enter data to delete:
25
Do you want to continue? (yes/no)
yes
Enter option:
4
20 40 50 60 75 80 Do you want to continue? (yes/no)
yes
```

```
Enter option:
3
Enter data to search:
45
Value not found
Do you want to continue? (yes/no)
yes
Enter option:
4
20 40 50 60 75 80 Do you want to continue? (yes/no)
yes
Enter option:
3
Enter data to search:
20
Value found
Do you want to continue? (yes/no)
yes
Enter option:
5
Leaf nodes are:
20 60 80 Do you want to continue? (yes/no)
yes
Enter option:
6
Parent is 50 and children are 40 and 75
Parent is 40 and child is 20
Parent is 75 and children are 60 and 80
Do you want to continue? (yes/no)
yes
Enter option:
7
Depth of tree is: 3
Do you want to continue? (yes/no)
no
```