Program:

#include <iostream>

#include<stdlib.h>

using namespace std;

struct node

{

int data;

struct node \*left;

struct node \*right;

};

node \*insert(node \*root, int val){

if (root == NULL){

node \*temp; //new node temp

temp=new node;

temp->data=val;

temp->left=temp->right=NULL; //left and right is NULL bcz only one

node create

return temp; // return single node

}

if (val < root->data){

root->left = insert(root->left, val);

}

else{

//val>root->data

root->right = insert(root->right, val);

}

return root;

}

void inorder(node \*root)

{

if (root == NULL){

return;

}

inorder(root->left);

cout << root->data << " ";

inorder(root->right);64

}

node\* inorderSucc(node\* root){

node\* curr=root;

while(curr && curr->left!=NULL){

curr=curr->left;

}

return curr;

}

node \*delet(node \*root, int key){

if(key<root->data){

root->left=delet(root->left, key);

}

else if(key>root->data){

root->right=delet(root->right,key);

}

//if key==root->data

else{

if(root->left==NULL){

node\* temp=root->right;

free(root);

return temp;

}

else if(root->right==NULL){

node\* temp=root->left;

free(root);

return temp;

}

node\* temp=inorderSucc(root->right);

root->data=temp->data;

root->right=delet(root->right, temp->data);

}

return root;

}

node \*search(node\* root, int val){

if(root==NULL){

return NULL;

}65

if(val>root->data){

return search(root->right, val);

}

else if(val<root->data){

return search(root->left, val);

}

else{

return root;

}

}

void mirrorImg(node\* root){

if(root==NULL){

return;

}

else{

struct node \*temp;

mirrorImg(root->left);

mirrorImg(root->right);

swap(root->left,root->right);

}

}

node \*copy(node \*root){

node \*temp=NULL;

if(root!=NULL){

temp=new node();

temp->data=root->data;

temp->left=copy(root->left);

temp->right=copy(root->right);

}

return temp;

}

void leafNodes(node\* root){

if(root==NULL){

return;

}

if(!root->left && !root->right){

cout<<root->data<<" ";66

return;

}

if(root->right)

leafNodes(root->right);

if(root->left)

leafNodes(root->left);

}

int calHeight(node\* root){

if(root==NULL){

return 0;

}

int lheight=calHeight(root->left);

int rheight=calHeight(root->right);

return max(lheight,rheight)+1;

}

node \*findMin(node \*root){

if(root==NULL){

return NULL;

}

if(root->left)

return findMin(root->left);

else

return root;

}

node \*findMax(node \*root){

if(root==NULL){

return NULL;

}

if(root->right)

return findMax(root->right);

else

return root;

}

int main()

{

node \*root=NULL, \*temp; //initially tree is NULL

int ch;67

while (1){

cout<<"\n\n\t1)Insert" << endl;

cout<<"\t2)Delete" << endl;

cout<<"\t3)Search" << endl;

cout<<"\t4)Create the copy "<<endl;

cout<<"\t5)Display leaf nodes "<<endl;

cout<<"\t6)Height of the tree"<<endl;

cout<<"\t7)Find the minimum"<<endl;

cout<<"\t8)Find the maximum"<<endl;

cout<<"\t9)Mirror image"<<endl;

cout<<"\t10)Exit"<<endl;

cout<<"\nEnter your choice: ";

cin>>ch;

switch (ch){

case 1:

cout << "Enter the element to be insert: ";

cin >> ch;

root= insert(root, ch);

cout << "\*\*\*\*\*Elements in BST are\*\*\*\*\*: ";

inorder(root);

break;

case 2:

cout<<"Enter the element to be deleted: ";

cin>>ch;

root=delet(root, ch);

cout<<"Element deleted successfully !!";

cout<<"\n\*\*\*\*\*After deletion the elements in the BST are\*\*\*\*\*: ";

inorder(root);

break;

case 3:

cout<<"Enter the element to be searched: ";

cin>>ch;

temp=search(root, ch);

if(temp==NULL){

cout<<"\*\*\*\*\*Element is not found\*\*\*\*\*";

}

else{68

cout<<"\*\*\*\*\*Element is found\*\*\*\*\*";

}

break;

case 4:

cout<<"The copy of the tree is: ";

root=copy(root);

inorder(root);

break;

case 5:

cout<<"The leaf nodes are: ";

leafNodes(root);

break;

case 6:

cout<<"Height of the binary search tree is: "<<calHeight(root);

break;

case 7:

temp=findMin(root);

cout<<"\nMinimum element is : "<<temp->data;

break;

case 8:

temp=findMax(root);

cout<<"\nMaximum element is: "<<temp->data;

break;

case 9:

cout<<" inorder tree: ";

inorder(root);

cout<<endl;

mirrorImg(root);

cout<<"mirror image is: ";

inorder(root);

break;

case 10:

return 0;

default:

cout<<"\nInvalid choice !! Please enter your choice again";

}

}69

return 0;

}

**OUTPUT:**

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 1

Enter the element to be insert: 2

\*\*\*\*\*Elements in BST are\*\*\*\*\*: 2

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 1

Enter the element to be insert: 6

\*\*\*\*\*Elements in BST are\*\*\*\*\*: 2 6

1)Insert70

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 1

Enter the element to be insert: 3

\*\*\*\*\*Elements in BST are\*\*\*\*\*: 2 3 6

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 1

Enter the element to be insert: 7

\*\*\*\*\*Elements in BST are\*\*\*\*\*: 2 3 6 7

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum71

9)Mirror image

10)Exit

Enter your choice: 1

Enter the element to be insert: 8

\*\*\*\*\*Elements in BST are\*\*\*\*\*: 2 3 6 7 8

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 1

Enter the element to be insert: 4

\*\*\*\*\*Elements in BST are\*\*\*\*\*: 2 3 4 6 7 8

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 2

Enter the element to be deleted: 2

Element deleted successfully !!

\*\*\*\*\*After deletion the elements in the BST are\*\*\*\*\*: 3 4 6 7 872

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 3

Enter the element to be searched: 4

\*\*\*\*\*Element is found\*\*\*\*\*

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 4

The copy of the tree is: 3 4 6 7 8

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum73

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 4

The copy of the tree is: 3 4 6 7 8

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 5

The leaf nodes are: 8 4

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 6

Height of the binary search tree is: 3

1)Insert

2)Delete74

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 7

Minimum element is : 3

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 8

Maximum element is: 8

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image75

10)Exit

Enter your choice: 9

inorder tree: 3 4 6 7 8

mirror image is: 8 7 6 4 3

1)Insert

2)Delete

3)Search

4)Create the copy

5)Display leaf nodes

6)Height of the tree

7)Find the minimum

8)Find the maximum

9)Mirror image

10)Exit

Enter your choice: 10