**Code:**

#include<iostream>

#include<cstdlib>

using namespace std;

class myBST {

public:

int data;

static int count;

myBST\* left;

myBST\* right;

myBST(int val) {

data = val;

left = NULL;

right = NULL;

}

myBST() {

data = 0;

left = NULL;

right = NULL;

}

void insertNode(int);

void removeNode(int);

void inorder(myBST\*);

void preorder(myBST\*);

void postorder(myBST\*);

void smallest(myBST\*);

int largest(myBST\*);

void search(int);

};

myBST\* root = NULL;

int myBST::count = 0;

void myBST::insertNode(int val) {

myBST\* temp = new myBST(val);

myBST\* trav = root;

myBST\* hold = NULL;

if (root == NULL) {

root = temp;

}

else {

while (trav != NULL) {

hold = trav;

if (val > trav->data) {

trav = trav->right;

}

else if (val < trav->data) {

trav = trav->left;

}

else {

cout << "Duplicate data";

delete temp;

return;

}

}

if (val > hold->data) {

hold->right = temp;

}

else if (val < hold->data) {

hold->left = temp;

}

}

}

void myBST::inorder(myBST\* r) {

if (r != NULL) {

r->inorder(r->left);

cout << r->data << " ";

r->inorder(r->right);

}

}

void myBST::preorder(myBST\* r) {

if (r != NULL) {

cout << r->data << " ";

r->preorder(r->left);

r->preorder(r->right);

}

}

void myBST::postorder(myBST\* r) {

if (r != NULL) {

r->postorder(r->left);

r->postorder(r->right);

cout << r->data << " ";

}

}

void myBST::smallest(myBST\* r) {

if (r->left != NULL) {

r->smallest(r->left);

}

else {

cout << "\nSmallest element in the tree is " << r->data;

}

}

int myBST::largest(myBST\* r) {

if (r->right != NULL) {

return r->largest(r->right);

}

else

return r->data;

}

void myBST::search(int val) {

myBST\* trav = root;

int flag = 1;

while (trav != NULL) {

if (val > trav->data) {

trav = trav->right;

}

else if (val < trav->data) {

trav = trav->left;

}

else {

flag = 0;

break;

}

}

if (flag == 0)

cout << "\nElement Found";

else

cout << "\nNot Element Found";

}

void myBST::removeNode(int val) {

myBST\* trav = root;

myBST\* hold = NULL;

bool findflag = false;

bool isleft = false;

while (trav != NULL) {

if (val > trav->data) {

hold = trav;

trav = trav->right;

isleft = false;

}

else if (val < trav->data) {

hold = trav;

trav = trav->left;

isleft = true;

}

else {

findflag = true;

break;

}

}

if (findflag == true) {

if (trav->left == NULL && trav->right == NULL) {

delete trav;

cout << "\nDeleted";

if (isleft == true) {

hold->left = NULL;

}

else {

hold->right = NULL;

}

if (trav == root) {

root = NULL; // Update root if the deleted node is the root

}

}

else if (trav->left == NULL && trav->right != NULL) {

if (isleft == true) {

hold->left = trav->right;

}

else {

hold->right = trav->right;

}

delete trav;

cout << "\nDeleted";

}

else if (trav->left != NULL && trav->right == NULL) {

if (isleft == true) {

hold->left = trav->left;

}

else {

hold->right = trav->left;

}

delete trav;

cout << "\nDeleted";

}

else {

int temp = trav->left->largest(trav->left);

root->removeNode(temp);

trav->data = temp;

cout << "\nDeleted";

}

}

else {

cout << "\nElement Not Found";

}

}

int main() {

int ch, p;

cout << "1) Insert element to tree " << endl;

cout << "2) Delete element from tree " << endl;

cout << "3) Display all the elements of tree by Inorder:" << endl;

cout << "4) Display all the elements of tree by Preorder:" << endl;

cout << "5) Display all the elements of tree by Postorder:" << endl;

cout << "6) Display the element of tree by Largest:" << endl;

cout << "7) Display the element of tree by Smallest:" << endl;

cout << "8) Search the element of tree " << endl;

cout << "9) Exit" << endl;

do {

cout << "\nEnter your choice : " << endl;

cin >> ch;

switch (ch) {

case 1:

cout << "\nEnter Element: ";

cin >> p;

root->insertNode(p);

break;

case 2:

cout << "\nEnter Element: ";

cin >> p;

root->removeNode(p);

cout << "\nAfter Element removed: ";

break;

case 3:

cout << "\nDisplay Elements Inorder: ";

root->inorder(root);

break;

case 4:

cout << "\nDisplay Elements Preorder: ";

root->preorder(root);

break;

case 5:

cout << "\nDisplay Elements Postorder: ";

root->postorder(root);

break;

case 6:

cout << "\nLargest in Tree:" << root->largest(root);

break;

case 7:

cout << "\nSmallest in Tree:";

root->smallest(root);

break;

case 8:

cout << "\nEnter Element: ";

cin >> p;

root->search(p);

break;

case 9:

cout << "Exit" << endl;

exit(1);

default:

cout << "Invalid choice" << endl;

}

} while (ch != 9);

return 0;

}

