Ex 9: Binary Search Tree

**Name: Sathvikha Reg no:231801160**

PROGRAM:

#include <stdio.h>

#include <stdlib.h>

structBinaryTreeNode {

int key;

structBinaryTreeNode \*left, \*right;

};

structBinaryTreeNode\* newNodeCreate(int value)

{

structBinaryTreeNode\* temp

= (structBinaryTreeNode\*)malloc(

sizeof(structBinaryTreeNode));

temp->key = value;

temp->left = temp->right = NULL;

return temp;

}

structBinaryTreeNode\*

searchNode(structBinaryTreeNode\* root, int target)

{

if (root == NULL || root->key == target) {

return root;

}

if (root->key < target) {

returnsearchNode(root->right, target);

}

returnsearchNode(root->left, target);

}

structBinaryTreeNode\*

insertNode(structBinaryTreeNode\* node, int value)

{

if (node == NULL) {

returnnewNodeCreate(value);

}

if (value < node->key) {

node->left = insertNode(node->left, value);

}

else if (value > node->key) {

node->right = insertNode(node->right, value);

}

return node;

}

voidpostOrder(structBinaryTreeNode\* root)

{

if (root != NULL) {

postOrder(root->left);

postOrder(root->right);

printf(" %d ", root->key);

}

}

voidinOrder(structBinaryTreeNode\* root)

{

if (root != NULL) {

inOrder(root->left);

printf(" %d ", root->key);

inOrder(root->right);

}

}

voidpreOrder(structBinaryTreeNode\* root)

{

if (root != NULL) {

printf(" %d ", root->key);

preOrder(root->left);

preOrder(root->right);

}

}

structBinaryTreeNode\* findMin(structBinaryTreeNode\* root)

{

if (root == NULL) {

return NULL;

}

else if (root->left != NULL) {

returnfindMin(root->left);

}

return root;

}

structBinaryTreeNode\* delete (structBinaryTreeNode\* root,

int x)

{

if (root == NULL)

return NULL;

if (x > root->key) {

root->right = delete (root->right, x);

}

else if (x < root->key) {

root->left = delete (root->left, x);

}

else {

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

free(root);

return NULL;

}

else if (root->left == NULL

|| root->right == NULL) {

structBinaryTreeNode\* temp;

if (root->left == NULL) {

temp = root->right;

}

else {

temp = root->left;

}

free(root);

return temp;

}

else {

structBinaryTreeNode\* temp

= findMin(root->right);

root->key = temp->key;

root->right = delete (root->right, temp->key);

}

}

return root;

}

int main()

{

structBinaryTreeNode\* root = NULL;

root = insertNode(root, 50);

insertNode(root, 30);

insertNode(root, 20);

insertNode(root, 40);

insertNode(root, 70);

insertNode(root, 60);

insertNode(root, 80);

if (searchNode(root, 60) != NULL) {

printf("60 found");

}

else {

printf("60 not found");

}

printf("\n");

postOrder(root);

printf("\n");

preOrder(root);

printf("\n");

inOrder(root);

printf("\n");

structBinaryTreeNode\* temp = delete (root, 70);

printf("After Delete: \n");

inOrder(root);

return 0;

}

OUTPUT:

