Expt no: 10

**PROGRAM: C Program to implement Binary Search Tree and its operations and traversals.**

**Input:**

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* createNode(int value)

{

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

newNode->left = newNode->right = NULL;

return newNode;

}

struct Node\* insert(struct Node\* root, int data)

{

if (root == NULL)

{

return createNode(data);

}

if (data < root->data) {

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

} else if (data > root->data) {

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

}

return root;

}

struct Node\* minValueNode(struct Node\* node)

{

struct Node\* current = node;

while (current && current->left != NULL)

{

current = current->left;

}

return current;

}

struct Node\* deleteNode(struct Node\* root, int data)

{

if (root == NULL)

{

return root;

}

if (data < root->data)

{

root->left = deleteNode(root->left, data);

} else if (data > root->data)

{

root->right = deleteNode(root->right, data);

} else {

if (root->left == NULL)

{

struct Node\* temp = root->right;

free(root);

return temp;

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

{

struct Node\* temp = root->left;

free(root);

return temp;

}

struct Node\* temp = minValueNode(root->right);

root->data = temp->data;

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

}

return root;

}

void inorder(struct Node\* root)

{

if (root != NULL)

{

inorder(root->left);

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

inorder(root->right);

}

}

void preorder(struct Node\* root)

{

if (root != NULL)

{

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

preorder(root->left);

preorder(root->right);

}

}

void postorder(struct Node\* root)

{

if (root != NULL)

{

postorder(root->left);

postorder(root->right);

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

}

}

struct Node\* search(struct Node\* root, int key)

{

if (root == NULL || root->data == key)

{

return root;

}

if (root->data < key)

{

return search(root->right, key);

}

return search(root->left, key);

}

int main()

{

struct Node\* root = NULL;

root = insert(root, 50);

insert(root, 30);

insert(root, 20);

insert(root, 40);

insert(root, 70);

insert(root, 60);

insert(root, 80);

printf("Inorder traversal: ");

inorder(root);

printf("\n");

printf("Preorder traversal: ");

preorder(root);

printf("\n");

printf("Postorder traversal: ");

postorder(root);

printf("\n");

int key = 40;

struct Node\* result = search(root, key);

if (result != NULL)

{

printf("Element %d found in the tree.\n", key);

} else

{

printf("Element %d not found in the tree.\n", key);

}

root = deleteNode(root, 20);

printf("Inorder traversal after deletion: ");

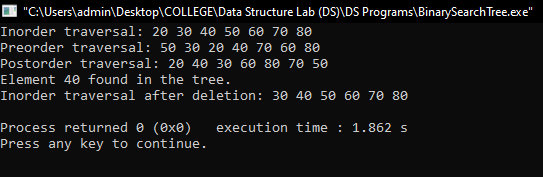
inorder(root);

printf("\n");

return 0;

}

**Output:**

****