GROUP MEMBERS:

1. MWANGI ANN MWENDE SCT222-0302/2022
2. BETH WAMUGWE KAMAU SCT222-0297/2022
3. BRENDA MAINA SCT222-0194/2021

Required:

Consider a binary search tree (BST) represented by an array shown below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 30 | 20 | 40 | 10 | 25 | 35 | 45 | 5 | 15 |

30 20 40 10 25 35 45 5 15

Write a C program with functions to do the following tasks

1. Create the BST represented by an array above
2. Delete any node from the BST
3. Print the height of the BST
4. Print the level and height of any node in the BST

**Solution**

#include <stdio.h>

#include <stdlib.h>

// Structure for a node in the BST

struct Node {

int data;

struct Node \*left, \*right;

};

// Function to create a new node

struct Node\* newNode(int key) {

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

temp->data = key;

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

return temp;

}

// Function to insert a new node into the BST

struct Node\* insert(struct Node\* root, int key) {

if (root == NULL) return newNode(key);

if (key < root->data)

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

else if (key > root->data)

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

return root;

}

// Function to delete a node from the BST

struct Node\* deleteNode(struct Node\* root, int key) {

if (root == NULL) return root;

if (key < root->data)

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

else if (key > root->data)

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

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 = root->right;

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

temp = temp->left;

root->data = temp->data;

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

}

return root;

}

// Function to find the height of the BST

int height(struct Node\* root) {

if (root == NULL) return -1;

int left\_height = height(root->left);

int right\_height = height(root->right);

return 1 + (left\_height > right\_height ? left\_height : right\_height);

}

// Function to print the level and height of a node in the BST

void printLevelAndHeight(struct Node\* root, int key, int level) {

if (root == NULL) {

printf("Node not found!\n");

return;

}

if (root->data == key) {

printf("Level: %d\n", level);

printf("Height: %d\n", height(root));

return;

}

if (key < root->data)

printLevelAndHeight(root->left, key, level + 1);

else

printLevelAndHeight(root->right, key, level + 1);

}

// Function to print the inorder traversal of the BST

void inorder(struct Node\* root) {

if (root != NULL) {

inorder(root->left);

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

inorder(root->right);

}

}

int main() {

int arr[] = {30, 20, 40, 10, 25, 35, 45, 5, 15};

int n = sizeof(arr) / sizeof(arr[0]);

struct Node\* root = NULL;

// Creating the BST represented by the array

for (int i = 0; i < n; i++)

root = insert(root, arr[i]);

printf("Inorder traversal of the BST: ");

inorder(root);

printf("\n");

// Deleting node 25

root = deleteNode(root, 25);

printf("Inorder traversal after deleting node 25: ");

inorder(root);

printf("\n");

// Printing the height of the BST

printf("Height of the BST: %d\n", height(root));

// Printing the level and height of node 40

int key = 40;

printLevelAndHeight(root, key, 0);

return 0;

}