Name: Wanjau Mwangi

Reg: SCT221-0500/2022

&

Name: Dennis Kariuki

Reg: SCT221-0497/2022

#include <stdio.h>

#include <stdlib.h>

// Structure for a node in BST

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

// Function to create a new node

struct Node\* createNode(int data) {

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

newNode->data = data;

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

return newNode;

}

// Function to insert a node into BST

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;

}

// Function to find minimum value node in a given BST

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

struct Node\* current = node;

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

current = current->left;

return current;

}

// Function to delete a node from 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 = minValueNode(root->right);

root->data = temp->data;

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

}

return root;

}

// Function to find height of BST

int height (struct Node\* node) {

if (node == NULL)

return -1;

else {

int leftHeight = height(node->left);

int rightHeight = height(node->right);

if (leftHeight > rightHeight)

return (leftHeight + 1);

else

return (rightHeight + 1);

}

}

// Function to print level of a node in BST

void printLevel(struct Node\* root, int level, int data, int currentLevel) {

if (root == NULL)

return;

if (root->data == data) {

printf("Level of node %d: %d\n", data, currentLevel);

return;

}

if (level > 1) {

printLevel(root->left, level - 1, data, currentLevel);

printLevel(root->right, level - 1, data, currentLevel);

}

}

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

void printHeightAndLevel(struct Node\* root, int data) {

int h = height(root);

for (int i = 1; i <= h + 1; i++) {

printLevel(root, i, data, i);

}

}

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 BST from the array

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

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

}

// Printing height of BST

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

// Deleting a node from BST

int key = 20;

root = deleteNode(root, key);

printf("Deleted node %d from BST\n", key);

// Printing height of BST after deletion

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

// Printing level and height of any node in BST

int nodeToFind = 30;

printf("Node: %d\n", nodeToFind);

printHeightAndLevel(root, nodeToFind);

return 0;

}