***GROUP MEMBERS***

JOY WANJIRU –SCT 221-0210/2022

MARGARET KARANJA – SCT 221-0229/2022

LISA WANJIKU –SCT 221-0562/2022

***ASSIGNMENT***

***Construction of binary search tree***

#include <stdio.h>

#include <stdlib.h>

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

};

struct TreeNode\* newNode(int val) {

struct TreeNode\* node = (struct TreeNode\*)malloc(sizeof(struct TreeNode));

node->val = val;

node->left = NULL;

node->right = NULL;

return node;

}

int search(int arr[], int start, int end, int value) {

for (int i = start; i <= end; i++) {

if (arr[i] == value)

return i;

}

return -1; // Not found

}

// Recursive function to construct BST from in-order and post-order arrays

struct TreeNode\* buildTree(int in[], int post[], int inStart, int inEnd, int\* postIndex) {

if (inStart > inEnd)

return NULL;

struct TreeNode\* node = newNode(post[(\*postIndex)--]);

if (inStart == inEnd)

return node;

int inIndex = search(in, inStart, inEnd, node->val);

node->right = buildTree(in, post, inIndex + 1, inEnd, postIndex);

node->left = buildTree(in, post, inStart, inIndex - 1, postIndex);

return node;

}

struct TreeNode\* bst\_construct(int in[], int post[], int n) {

int postIndex = n - 1;

return buildTree(in, post, 0, n - 1, &postIndex);

}

void printLevelOrder(struct TreeNode\* root) {

if (root == NULL)

return;

// Create an array to hold the nodes at each level

struct TreeNode\*\* queue = (struct TreeNode\*\*)malloc(sizeof(struct TreeNode\*) \* 1000); // Assuming a maximum of 1000 nodes

int front = 0, rear = 0;

queue[rear++] = root;

while (front < rear) {

struct TreeNode\* temp = queue[front++];

printf("%d ", temp->val);

if (temp->left != NULL)

queue[rear++] = temp->left;

if (temp->right != NULL)

queue[rear++] = temp->right;

}

free(queue);

}

// Main function

int main() {

int inOrder[] = {5, 10, 10, 15, 20, 20, 25, 25, 30, 30, 45};

int postOrder[] = {5, 15, 10, 10, 20, 45, 30, 30, 25, 25, 20};

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

struct TreeNode\* root = bst\_construct(inOrder, postOrder, n);

printf("Breadth-First Search Traversal:\n");

printLevelOrder(root);

printf("\n");

return 0;

}