#pragma once

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#include <iostream>

using namespace std;

class BalanceBST

{

public:

BalanceBST();

~BalanceBST();

//Node structure with two pointers as normal for a binary tree

struct node {

char data;

struct node\* left;

struct node\* right;

};

struct node\* createNewNode(char value) {

//Allocate space needed for new node

struct node\* node = (struct node\*)

malloc(sizeof(struct node));

node->data = value;

node->left = NULL;

node->right = NULL;

return(node);

};

/\*

struct node\* swapWithLeftChild(node\* node) {

struct node\* temp = node->left;

temp->left = node;

node = temp;

return temp;

};

struct node\* swapWithRightChild(node\* node) {

struct node\* temp = node->right;

temp->right = node;

node = temp;

return temp;

}

int balanceTree(node\* node) {

if (node->left->data > node->data)

swapWithLeftChild(node);

if (node->right->data > node->data)

swapWithRightChild(node);

return balanceTree(node->left) && balanceTree(node->right);

}

\*/

//The Above is code that i thought i might need but current do not

//I am keeping it as reference for future projects

void createBalancedTree(node \*\*root, char arr[], int start, int end) {

if (start <= end)

{

int mid = (start + end + 1) / 2;

\*root = createNewNode(arr[mid]);

//Creates a binary tree based on middle positioning, ensuring that nodes are placed accurately

createBalancedTree(&((\*root)->left), arr, start, mid - 1);

createBalancedTree(&((\*root)->right), arr, mid + 1, end);

}

};

void printTree(node\* node) {

cout << node->data << " ";

printTree(node->left);

printTree(node->right);

}

int main() {

//Create and initialize alphabet array

char array[26] = {'a', 'b', 'c', 'd', 'e', 'f', 'g',

'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',

'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'

};

struct node\* root = createNewNode(array[13]);

createBalancedTree(&root, array, 0, 26);

printTree(root);

};

};