# Mahad Ghauri

233523

Lab Task 08

#include <iostream>

#include <string>

using namespace std;

class Node

{

private:

int data;

Node \*left, \*right;

public:

Node()

{

}

Node(int x)

{

data = x;

left = right = NULL;

}

void setData(int value)

{

data = value;

}

int getData()

{

return data;

}

void setLeft(Node \*other)

{

left = other;

}

Node \*getLeft()

{

return left;

}

void setRight(Node \*other)

{

right = other;

}

Node \*getRight()

{

return right;

}

};

Node \*insertNode(int arr[], int start, int end)

{

if (start > end)

{

return NULL;

}

int indexNumber = (start + end) / 2;

Node \*root = new Node(arr[indexNumber]);

root->setLeft(insertNode(arr, start, indexNumber - 1));

root->setRight(insertNode(arr, indexNumber + 1, end));

return root;

}

int searchNode(Node \*root, int target)

{

if (!root)

{

return -1;

}

int closestNumber = root->getData();

Node \*current = root;

while (current)

{

// the abs is a built in function which will give the absolute value of the number

if (abs(current->getData() - target) < abs(closestNumber - target))

{

closestNumber = current->getData();

}

if (target < current->getData())

{

current = current->getLeft();

}

else if (target > current->getData())

{

current = current->getRight();

}

else

{

return current->getData();

}

}

return closestNumber;

}

bool isValueCorrectlyPlaced(Node \*root, int value)

{

Node \*current = root;

while (current)

{

if (value < current->getData())

{

if (current->getLeft() && current->getLeft()->getData() == value)

{

return true;

}

current = current->getLeft();

}

else if (value > current->getData())

{

if (current->getRight() && current->getRight()->getData() == value)

{

return true;

}

current = current->getRight();

}

else

{

return false;

}

}

return false;

}

void inOrder(Node \*root)

{

if (root == NULL)

{

return;

}

inOrder(root->getLeft());

cout << root->getData() << " ";

inOrder(root->getRight());

}

void preOrder(Node \*root)

{

if (root == NULL)

return;

cout << " " << root->getData();

preOrder(root->getLeft());

preOrder(root->getRight());

}

void postOrder(Node \*root)

{

if (root == NULL)

{

return;

}

postOrder(root->getLeft());

postOrder(root->getRight());

cout << root->getData() << " ";

}

int main()

{

Node \*root = NULL;

int arr[] = {1, 2, 3, 4, 5, 6, 7};

root = insertNode(arr, 0, 6);

cout << "\n InOrder: ";

inOrder(root);

cout << "\n PreOrder: ";

preOrder(root);

cout << "\n PostOrder:";

postOrder(root);

cout << "\n";

int targetValue = 8;

int found = searchNode(root, targetValue);

cout << "\n The closest number to the " << targetValue << " is: " << found << endl;

int randomValue = 4;

if (isValueCorrectlyPlaced(root, randomValue))

{

cout << "The value " << randomValue << " is correctly placed in the BST." << endl;

}

else

{

cout << "The value " << randomValue << " is NOT correctly placed in the BST." << endl;

}

return 0;

}

