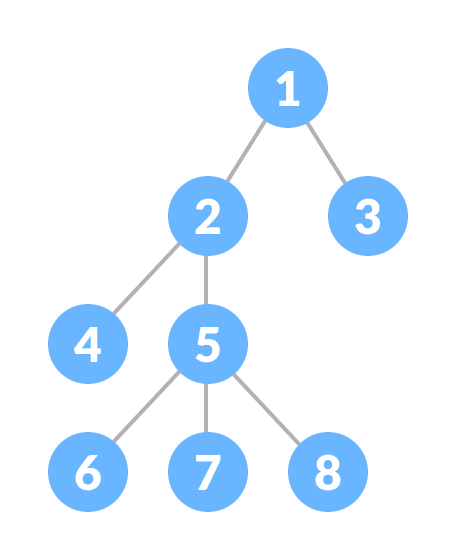
**COMP 203 Lab 8**

**Tree and Queue**



**SOLUTION**

import java.util.LinkedList;

import java.util.Queue;

//5pt for ability to use built-in Queue data struture in java \*\*

class Node<E> { //2pt

E data;

Queue<Node<E>> childrenList; // \*\*

Node<E> parent;

public Node(E data) {//3pt

this.data = data;

this.childrenList = new LinkedList<>();// \*\*

this.parent = null;

}

}

class QTree<E> { //2pt

Node<E> root;

public QTree() {//3pt

this.root = null;

}

public void insertNode(Node<E> root, E parentValue, E addedValue) {//20pt for insertNode function and all its helper functions

Node<E> parentNode = findNode(root, parentValue);

if (parentNode != null) {

Node<E> newNode = new Node<>(addedValue);

newNode.parent = parentNode;

parentNode.childrenList.add(newNode); // \*\* add is a built in function for queue ds in java

} else {

System.out.println("Parent node not found");

}

}

private Node<E> findNode(Node<E> root, E value) {

if (root == null || root.data.equals(value)) {

return root;

}

for (Node<E> child : root.childrenList) {

Node<E> result = findNode(child, value);

if (result != null) {

return result;

}

}

return null;

}

public int size() {//15pt for size function and all its helper functions

return size(root);

}

private int size(Node<E> node) {

if (node == null) {

return 0;

}

int count = 1; // Count current node

for (Node<E> child : node.childrenList) {

count += size(child);

}

return count;

}

public boolean isPresent(Node<E> root, E value) {{//15pt for isPresent function and all its helper functions

return isPresentHelper(root, value);

}

private boolean isPresentHelper(Node<E> node, E value) {

if (node == null) {

return false;

}

if (node.data.equals(value)) {

return true;

}

for (Node<E> child : node.childrenList) {

if (isPresentHelper(child, value)) {

return true;

}

}

return false;

}

}

public class Main {

public static void main(String[] args) {

QTree<Integer> tree = new QTree<>();

// Insert nodes to create the tree //15pt for creating the given tree

//first create the root

tree.root = new Node<>(1);

tree.insertNode(tree.root, 1, 2); //5pt for testing insertNode

tree.insertNode(tree.root, 1, 3);

tree.insertNode(tree.root, 2, 4);

tree.insertNode(tree.root, 2, 5);

tree.insertNode(tree.root, 5, 6);

tree.insertNode(tree.root, 5, 7);

tree.insertNode(tree.root, 5, 8);

// Test the methods

System.out.println("Tree size: " + tree.size());//5pt for testing size() function

System.out.println("Is 5 present? " + tree.isPresent(tree.root, 5)); //5pt for testing isPresent

System.out.println("Is 10 present? " + tree.isPresent(tree.root, 10));

}

}