**Assignment23**

**1.**

function calculateDepth(preorder) {

let depth = 0;

let stack = [];

for (let i = 0; i < preorder.length; i++) {

if (preorder[i] === 'n') {

depth++;

stack.push(depth);

} else if (preorder[i] === 'l') {

if (stack.length > 0) {

depth = stack.pop();

}

}

}

return depth;

}

// Example:

const preorder1 = 'nlnll';

console.log(calculateDepth(preorder1));

Output: 2

const preorder2 = 'nlnnlll';

console.log(calculateDepth(preorder2));

Output: 3

**2.**

class Node {

constructor(value) {

this.val = value;

this.left = null;

this.right = null;

}

}

function printLeftView(root) {

if (root === null) {

return;

}

const queue = [];

queue.push(root);

while (queue.length > 0) {

const levelSize = queue.length;

for (let i = 0; i < levelSize; i++) {

const node = queue.shift();

if (i === 0) {

console.log(node.val);

}

if (node.left !== null) {

queue.push(node.left);

}

if (node.right !== null) {

queue.push(node.right);

}

}

}

}

// Example:

const root1 = new Node(4);

root1.left = new Node(5);

root1.right = new Node(2);

root1.right.left = new Node(3);

root1.right.right = new Node(1);

root1.right.left.left = new Node(6);

root1.right.left.right = new Node(7);

printLeftView(root1);

Output: 4 5 3 6

const root2 = new Node(1);

root2.left = new Node(2);

root2.right = new Node(3);

root2.left.right = new Node(4);

root2.left.right.right = new Node(5);

root2.left.right.right.right = new Node(6);

printLeftView(root2);

Output: 1 2 4 5 6

**3.**

class Node {

constructor(value) {

this.val = value;

this.left = null;

this.right = null;

}

}

function printRightView(root) {

if (root === null) {

return;

}

const queue = [];

queue.push(root);

while (queue.length > 0) {

const levelSize = queue.length;

for (let i = 0; i < levelSize; i++) {

const node = queue.shift();

if (i === levelSize - 1) {

console.log(node.val);

}

if (node.left !== null) {

queue.push(node.left);

}

if (node.right !== null) {

queue.push(node.right);

}

}

}

}

// Example:

const root1 = new Node(1);

root1.left = new Node(2);

root1.right = new Node(3);

root1.left.left = new Node(4);

root1.left.right = new Node(5);

root1.right.left = new Node(6);

root1.right.right = new Node(7);

root1.right.left.right = new Node(8);

printRightView(root1);

Output: 1 3 7 8

const root2 = new Node(1);

root2.left = new Node(8);

root2.left.left = new Node(7);

printRightView(root2);

Output: 1 8 7

4.

class Node {

constructor(value) {

this.val = value;

this.left = null;

this.right = null;

this.hd = 0;

}

}

function printBottomView(root) {

if (root === null) {

return;

}

const map = new Map();

const queue = [];

queue.push(root);

while (queue.length > 0) {

const node = queue.shift();

const hd = node.hd;

map.set(hd, node.val);

if (node.left !== null) {

node.left.hd = hd - 1;

queue.push(node.left);

}

if (node.right !== null) {

node.right.hd = hd + 1;

queue.push(node.right);

}

}

for (let [key, value] of map) {

console.log(value);

}

}

// Example:

const root1 = new Node(20);

root1.left = new Node(8);

root1.right = new Node(22);

root1.left.left = new Node(5);

root1.left.right = new Node(3);

root1.right.right = new Node(25);

root1.left.right.left = new Node(10);

root1.left.right.right = new Node(14);

printBottomView(root1);

Output: 5 10 3 14 25

const root2 = new Node(20);

root2.left = new Node(8);

root2.right = new Node(22);

root2.left.left = new Node(5);

root2.left.right = new Node(3);

root2.right.left = new Node(4);

root2.right.right = new Node(25);

root2.left.right.left = new Node(10);

root2.left.right.right = new Node(14);

printBottomView(root2);

Output: 5 10 4 14 25