**Q6: AVL Tree for Stock Price Management**: Use an **AVL tree** to maintain stock prices. Ensure that after each insertion, the tree remains balanced by performing rotations.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AVL Tree Stock Price Manager</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

h1 {

text-align: center;

}

#treeOutput {

margin-top: 20px;

white-space: pre-wrap;

font-family: monospace;

background: #f9f9f9;

padding: 10px;

border: 1px solid #ddd;

}

input, button {

margin: 5px 0;

padding: 10px;

font-size: 16px;

}

</style>

</head>

<body>

<h1>AVL Tree Stock Price Management</h1>

<label for="stockPrice">Enter Stock Price:</label>

<input type="number" id="stockPrice" placeholder="Enter price..." />

<button onclick="addStockPrice()">Add Price</button>

<h3>Tree Visualization:</h3>

<div id="treeOutput">Tree will appear here...</div>

<script>

class Node {

constructor(value) {

this.value = value;

this.height = 1;

this.left = null;

this.right = null;

}

}

class AVLTree {

constructor() {

this.root = null;

}

getHeight(node) {

return node ? node.height : 0;

}

getBalance(node) {

return node ? this.getHeight(node.left) - this.getHeight(node.right) : 0;

}

rotateRight(y) {

let x = y.left;

let T = x.right;

x.right = y;

y.left = T;

y.height = Math.max(this.getHeight(y.left), this.getHeight(y.right)) + 1;

x.height = Math.max(this.getHeight(x.left), this.getHeight(x.right)) + 1;

return x;

}

rotateLeft(x) {

let y = x.right;

let T = y.left;

y.left = x;

x.right = T;

x.height = Math.max(this.getHeight(x.left), this.getHeight(x.right)) + 1;

y.height = Math.max(this.getHeight(y.left), this.getHeight(y.right)) + 1;

return y;

}

insert(node, value) {

if (!node) return new Node(value);

if (value < node.value) node.left = this.insert(node.left, value);

else if (value > node.value) node.right = this.insert(node.right, value);

else return node;

node.height = Math.max(this.getHeight(node.left), this.getHeight(node.right)) + 1;

let balance = this.getBalance(node);

if (balance > 1 && value < node.left.value) return this.rotateRight(node);

if (balance < -1 && value > node.right.value) return this.rotateLeft(node);

if (balance > 1 && value > node.left.value) {

node.left = this.rotateLeft(node.left);

return this.rotateRight(node);

}

if (balance < -1 && value < node.right.value) {

node.right = this.rotateRight(node.right);

return this.rotateLeft(node);

}

return node;

}

add(value) {

this.root = this.insert(this.root, value);

}

print(node, space = 0, result = []) {

if (node) {

space += 5;

this.print(node.right, space, result);

result.push(" ".repeat(space) + node.value);

this.print(node.left, space, result);

}

return result;

}

}

const tree = new AVLTree();

function addStockPrice() {

const priceInput = document.getElementById("stockPrice");

const price = parseInt(priceInput.value);

if (!price) {

alert("Please enter a valid stock price!");

return;

}

tree.add(price);

priceInput.value = "";

const treeOutput = document.getElementById("treeOutput");

const treeStructure = tree.print(tree.root).join("\n");

treeOutput.textContent = treeStructure || "Tree is empty.";

}

</script>

</body>

</html>