/\*\*\*\*\*\*\*\*\*\*\*\*\* variables \*\*\*\*\*\*\*\*\*\*\*\*\*/

var add = document.getElementById("add");

var search = document.getElementById("search");

var input = document.getElementById("input");

var speed = document.getElementById("speedChange");

var undo = document.getElementById("undo");

var redo = document.getElementById("redo");

var root = null;

var time = 1000;

var rootTopPosition = 80;

let search\_interval;

const SEARCH\_INTERVAL\_TIME = 800;

/////

var LRNArray = [];

var LNRArray = [];

var NLRArray = [];

class Stack {

#items = [];

push = (element) => this.#items.push(element);

pop = () => this.#items.pop();

isempty = () => this.#items.length === 0;

empty = () => (this.#items.length = 0);

size = () => this.#items.length;

}

const undoStack = new Stack();

const redoStack = new Stack();

/////

function LRN(a) {

if (a.right == null && a.left == null) {

LRNArray.push(a.n.innerHTML);

return;

}

if (a.left != null) {

LRN(a.left);

}

if (a.right != null) {

LRN(a.right);

}

LRNArray.push(a.n.innerHTML);

return;

}

function LNR(a) {

if (a.right == null && a.left == null) {

LNRArray.push(a.n.innerHTML);

return;

}

if (a.left != null) {

LNR(a.left);

}

LNRArray.push(a.n.innerHTML);

if (a.right != null) {

LNR(a.right);

}

return;

}

function NLR(a) {

NLRArray.push(a.n.innerHTML);

if (a.right == null && a.left == null) {

return;

}

if (a.left != null) {

NLR(a.left);

}

if (a.right != null) {

NLR(a.right);

}

return;

}

//////////////////////////////////

document.getElementById("LRN").onclick = function () {

LRN(root);

alert("Kết quả duyệt cây LRN của bạn là:");

alert(LRNArray);

LRNArray = [];

};

document.getElementById("LNR").onclick = function () {

LNR(root);

alert("Kết quả duyệt cây LNR của bạn là:");

alert(LNRArray);

LNRArray = [];

};

document.getElementById("NLR").onclick = function () {

NLR(root);

alert("Kết quả duyệt cây NLR của bạn là:");

alert(NLRArray);

NLRArray = [];

};

/////////////////////////////////////

///////////////////////////////////////////////

document.onkeypress = function (e) {

// console.log(e.keyCode, e.charCode);

if (e.keyCode == 13 || e.charCode == 105 || e.charCode == 73) {

Add();

input.value = "";

} else if (e.charCode == 115 || e.charCode == 83) {

callSearch();

input.value = "";

} else if (e.charCode == 100 || e.charCode == 68) {

callDelete();

input.value = "";

}

};

speed.onchange = function () {

speedChange();

};

var count = 0;

add.onclick = function () {

Add();

count = input.value;

checkUndo = false;

checkRedo = false;

};

search.onclick = function () {

callSearch();

};

Deleter.onclick = function () {

callDelete();

};

undo.onclick = function () {

callDeleteUndo();

};

redo.onclick = function () {

AddRedo();

};

/////////////////////////////////////////////////////////////

function callDeleteUndo() {

if (!undoStack.isempty()) {

const val = undoStack.pop();

redoStack.push(val);

root = Delete(val, root);

setTimeout(function () {

// Draw AVL-tree

mainColor(root);

Reallocate(root, window.innerWidth / 2, rootTopPosition);

var temp = mostLeft(root);

if (val < 0) {

setPosition(root, -1 \* parseInt(temp.node.left));

}

}, time);

} else {

alert("Không thể redo được nữa");

}

}

function AddRedo() {

if (!redoStack.isempty()) {

const val = redoStack.pop();

undoStack.push(val);

if (!root) {

root = new Node(val, window.innerWidth / 2, rootTopPosition);

return;

// Add root at first

} else {

root = insert(val, root, root.node.left, root.node.top);

// Insert node to AVL-tree

}

setTimeout(function () {

// Draw AVL-tree

Reallocate(root, window.innerWidth / 2, rootTopPosition);

var temp = mostLeft(root);

if (parseInt(temp.node.left) < 0) {

setPosition(root, -1 \* parseInt(temp.node.left));

}

mainColor(root);

}, time);

} else {

alert("Không thể redo được nữa");

}

}

//////////////////////////////////////////////////////////////////////////

function callSearch() {

if (input.value == "") {

alert("Bạn cần phải nhập giá trị cần tìm kiếm!");

return;

}

Search(parseInt(input.value), root);

setTimeout(function () {

mainColor(root);

}, 3 \* time);

}

function callDelete() {

if (input.value == "") {

alert("Bạn cần phải nhập giá trị cần xóa!");

return;

}

root = Delete(parseInt(input.value), root);

setTimeout(function () {

// Draw AVL-tree

mainColor(root);

Reallocate(root, window.innerWidth / 2, rootTopPosition);

var temp = mostLeft(root);

if (parseInt(temp.node.left) < 0) {

setPosition(root, -1 \* parseInt(temp.node.left));

}

}, time);

}

function Add() {

if (input.value == "") {

alert("Bạn cần phải nhập giá trị cần thêm!");

return;

}

undoStack.push(parseInt(input.value));

redoStack.empty();

// mainColor(root);

if (!root) {

root = new Node(

parseInt(input.value),

window.innerWidth / 2,

rootTopPosition

);

return;

// Add root at first

} else {

root = insert(parseInt(input.value), root, root.node.left, root.node.top);

// Insert node to AVL-tree

}

setTimeout(function () {

// Draw AVL-tree

Reallocate(root, window.innerWidth / 2, rootTopPosition);

var temp = mostLeft(root);

if (parseInt(temp.node.left) < 0) {

setPosition(root, -1 \* parseInt(temp.node.left));

}

mainColor(root);

}, time);

}

function insert(val, node, x, y) {

if (!node) {

return new Node(val, x, y);

}

if (val < node.n.innerHTML) {

node.node.backgroundColor = "yellow";

node.left = insert(

val,

node.left,

parseInt(node.node.left) - 50,

parseInt(node.node.top) + 50

);

// Insert node to left

} else if (val > node.n.innerHTML) {

node.node.backgroundColor = "yellow";

node.right = insert(

val,

node.right,

parseInt(node.node.left) + 50,

parseInt(node.node.top) + 50

);

// Insert node to right

} else {

return node;

// Duplicate node

}

node.h = 1 + Math.max(Height(node.left), Height(node.right));

// Increase height of node

var balance = GetBalance(node);

// Get balance for AVL-tree

// left left rotation

if (balance > 1 && val < node.left.n.innerHTML) {

node = rotateToRight(node);

return node;

}

// right right rotation

if (balance < -1 && val > node.right.n.innerHTML) {

node = rotateToLeft(node);

return node;

}

// Left Right Case

if (balance > 1 && val > node.left.n.innerHTML) {

node.left = rotateToLeft(node.left);

node = rotateToRight(node);

return node;

}

// Right Left Case

if (balance < -1 && val < node.right.n.innerHTML) {

node.right = rotateToRight(node.right);

node = rotateToLeft(node);

return node;

}

return node;

}

function Height(node) {

// Return Height of node

if (!node) {

return -1;

}

return node.h;

}

function GetBalance(node) {

//Check balance of AVL-tree return distance different

if (!node) {

return 0;

}

return Height(node.left) - Height(node.right);

}

function rotateToRight(node) {

var n = node.left;

var nr = n.right;

//Perform rotation

n.right = node;

node.left = nr;

//Update heights

node.h = 1 + Math.max(Height(node.left), Height(node.right));

n.h = 1 + Math.max(Height(n.left), Height(n.right));

return n;

/\*

T1, T2, T3 and T4 are subtrees.

z y

/ \ / \

y T4 Right Rotate (z) x z

/ \ - - - - - - - - -> / \ / \

x T3 T1 T2 T3 T4

/ \

T1 T2

\*/

}

function rotateToLeft(node) {

var newP = node.right;

var temp = newP.left;

//Perform ratation

newP.left = node;

node.right = temp;

//Update heights

node.h = 1 + Math.max(Height(node.left), Height(node.right));

newP.h = 1 + Math.max(Height(newP.left), Height(newP.right));

return newP;

/\*

z y

/ \ / \

T1 y Left Rotate(z) z x

/ \ - - - - - - - -> / \ / \

T2 x T1 T2 T3 T4

/ \

T3 T4

\*/

}

function Reallocate(node, x, y) {

// Get location for node and line

if (!node) return;

var temp = Math.pow(2, node.h - 1) \* 50;

if (node.linel) {

document.body.removeChild(node.linel);

node.linel = null;

}

if (node.liner) {

document.body.removeChild(node.liner);

node.liner = null;

}

if (node.left) {

node.linel = getLine(x, y, x - temp, y + 100, 1);

}

if (node.right) {

node.liner = getLine(x, y, x + temp, y + 100, -1);

}

node.node.left = x + "px";

node.node.top = y + "px";

// node.node.backgroundColor = "red";

Reallocate(node.left, x - temp, y + 100);

Reallocate(node.right, x + temp, y + 100);

}

function mostLeft(node) {

// Get node.left last

var cur = node;

while (cur.left) {

cur = cur.left;

}

return cur;

}

function setPosition(node, shifting) {

// Set Position to draw

if (!node) {

return;

}

setPosition(node.left, shifting);

setPosition(node.right, shifting);

node.node.left = parseInt(node.node.left) + shifting + "px";

if (node.linel) {

node.linel.style.left = parseInt(node.linel.style.left) + shifting + "px";

}

if (node.liner) {

node.liner.style.left = parseInt(node.liner.style.left) + shifting + "px";

}

}

function Search(val, node) {

// Search value in AVL-tree

search\_interval = setTimeout(() => {

if (!node) {

alert("Not found :V");

clearInterval(search\_interval);

return;

} else if (node.n.innerHTML == val) {

node.node.backgroundColor = "green";

return;

} else if (node.n.innerHTML < val) {

node.node.backgroundColor = "yellow";

Search(val, node.right);

// node.node.backgroundColor = "red";

} else if (node.n.innerHTML > val) {

node.node.backgroundColor = "yellow";

Search(val, node.left);

// node.node.backgroundColor = "red";

}

}, SEARCH\_INTERVAL\_TIME);

}

function Delete(val, node) {

if (!node) {

return node;

}

node.node.backgroundColor = "yellow";

// Recursion find value to delete

if (val < node.n.innerHTML) {

node.left = Delete(val, node.left);

} else if (val > node.n.innerHTML) {

node.right = Delete(val, node.right);

}

// Had found value to delete

else if (val == node.n.innerHTML) {

// Remove node and line

if (!node.left) {

// Check node left child is null

var temp = node;

node = node.right;

document.body.removeChild(temp.n);

if (node) {

// Check node right child not null

document.body.removeChild(temp.liner);

// Remove line at right

}

console.log(delete temp);

temp = null;

return node;

} else if (!node.right) {

var temp = node;

node = node.left;

document.body.removeChild(temp.n);

document.body.removeChild(temp.linel);

temp = null;

console.log(delete temp);

return node;

} // Node is parent

else {

var temp = mostLeft(node.right);

node.n.innerHTML = temp.n.innerHTML;

node.right = Delete(parseInt(temp.n.innerHTML), node.right);

}

}

node.h = 1 + Math.max(Height(node.left), Height(node.right));

var balance = GetBalance(node);

// Left Left Case

if (balance > 1 && GetBalance(node.left) >= 0) {

node = rotateToRight(node);

return node;

}

// Right Right Case

if (balance < -1 && GetBalance(node.right) <= 0) {

node = rotateToLeft(node);

return node;

}

// Left Right Case

if (balance > 1 && GetBalance(node.left) < 0) {

node.left = rotateToLeft(node.left);

node = rotateToRight(node);

return node;

}

// Right Left Case

if (balance < -1 && GetBalance(node.right) > 0) {

node.right = rotateToRight(node.right);

node = rotateToLeft(node);

return node;

}

return node;

}

function Node(val, x, y) {

// Initiation Node struct

this.left = null;

this.right = null;

this.h = 0;

this.n = document.createElement("div");

this.n.innerHTML = val;

this.n.className = "node";

this.node = this.n.style;

this.node.top = y + "px";

this.node.left = x + "px";

this.linel = null;

this.liner = null;

document.body.appendChild(this.n);

return this;

}

function sleep(ms) {

var curT = new Date().getTime();

var duration = curT + ms;

while (curT < duration) {

curT = new Date().getTime();

}

}

function getLength(x1, y1, x2, y2) {

// Return length of line (hypotenuse)

var x = Math.pow(y1 - y2, 2);

var y = Math.pow(x1 - x2, 2);

return Math.sqrt(x + y);

}

function getAngle(x1, x2, dist) {

// Get sin to calculate degree in rad

var a = Math.abs(x1 - x2);

return Math.asin(a / dist);

}

function getLine(x1, y1, x2, y2, fact) {

// Draw line

var line = document.createElement("div");

line.className = "line";

line.style.top = y1 + 25 + "px";

line.style.left = x1 + 25 + "px";

var length = getLength(x1, y1, x2, y2);

line.style.height = length + "px";

line.style.transform = "rotate(" + fact \* getAngle(x1, x2, length) + "rad)";

document.body.appendChild(line);

return line;

}

function mainColor(node) {

// Main color

if (!node) return;

node.node.backgroundColor = "red";

mainColor(node.left);

mainColor(node.right);

}

function speedChange() {

\_time = document.getElementById("speedChange").value;

time = (\_time / 10) \* 1000;

console.log(time);

}