**Solutions – Exam10II2019**

# Problem 1. HTML Grade

function calculateGrade(examPoints, homeworkPoints, totalHomework){

let totalExamPoits = (examPoints \* 100 /400) \* 0.9 ;

let totalHomeworkPoits = (homeworkPoints \* 100 / totalHomework) \* 0.1;

let totalPoints = totalExamPoits + totalHomeworkPoits;

let grade = 3 + 2 \* (totalPoints - 100 / 5) / (100 / 2);

if (examPoints === 400){

grade = 6;

}

if(grade < 3){

grade = 2;

} else if (grade >= 6){

grade = 6;

}

let gradeString = grade.toFixed(2);

console.log(gradeString)

//console.log(grade)

return gradeString;

}

calculateGrade(300, 10, 10)

calculateGrade(200, 5, 5)

calculateGrade(290, 5, 5)

# Problem 2. Table Filter

function showWantedData(matrixTable, command){

let commandArray = command.split(/\s+/);

//let commandArray = command.split(/[ ]+/);

//let commandArray = command.split(' ');

switch(commandArray[0]){

case 'hide': hide(matrixTable, commandArray); break;

case 'sort': sortTable(matrixTable, commandArray); break;

case 'filter': filterTable(matrixTable, commandArray); break;

}

function hide(table, commandArray){

let indexToDelete = table[0].indexOf(commandArray[1]);

for(let row of table){

row = row.splice(indexToDelete, 1);

}

printTable(table);

}

function sortTable(table, commandArray){

let searchedHeadingIndex = table[0].indexOf(commandArray[1]);

let newTable = [];

let valueToSort = [];

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

if(i !== 0){

valueToSort.push(table[i][searchedHeadingIndex]);

} else {

newTable.push(table[i]);

}

}

valueToSort.sort();

for(let sortingValue of valueToSort){

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

if(i !== 0){

table[i].forEach((v, index) => {

if(index === searchedHeadingIndex && v === sortingValue){

newTable.push(table[i]);

}

})

}

}

}

printTable(newTable);

}

function filterTable(table, commandArray){

let searchedHeadingIndex = table[0].indexOf(commandArray[1]);

let newTable = [];

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

if(i !== 0){

table[i].forEach((v, index) => {

if(index === searchedHeadingIndex && v === commandArray[2]){

newTable.push(table[i]);

}

})

} else {

newTable.push(table[i]);

}

}

printTable(newTable);

}

function printTable (table){

for(let row of table){

console.log(row.join(' | '));

}

}

}

showWantedData([['name', 'age', 'grade'], ['Peter', '25', '5.00'], ['George', '34', '6.00'], ['Marry', '28', '5.49']], 'sort name')

showWantedData([['name', 'age', 'grade'], ['Peter', '25', '5.00'], ['George', '34', '6.00'], ['Marry', '28', '5.49']], 'hide age')

showWantedData([['firstName', 'age', 'grade', 'course'], ['Peter', '25', '5.00', 'JS-Core'], ['George', '34', '6.00', 'Tech'], ['Marry', '28', '5.49', 'Ruby']], 'filter firstName Marry')

**Problem 3. HTML Strict Mode**

function extractText(stringArray) {

let textArray = [];

let startTagPattern = /^<\w+>/;

let endTagPattern = /<\/\w+>$/;

let linePattern = /^(<\w+>.+<\/\w+>)$/;

for (let line of stringArray) {

if (!linePattern.test(line)) {

continue;

}

let [startTag, endTag] = [line.match(startTagPattern), line.match(endTagPattern)];

if(startTag[0] !== endTag[0].replace('/', '')){

continue;

}

line = line.replace(/<\w+>/g, '');

line = line.replace(/<\/\w+>/g, '');

textArray.push(line);

}

let text = textArray.join(' ');

//console.log(text);

return text;

}

// function extractText(stringArray){

// console.log(stringArray.filter(s => /^<(\w+)>.+<(\/\1)>$/.test(s)).map(s => s.replace(/<\/?\w+>/g, '')).join(' '));

// }

extractText(['<h1><span>Hello World!</span></h1>', '<p>I am Peter.'])

extractText(['<div><p>This</p> is</div>', '<div><a>perfectly</a></div>', '<divs><p>valid</p></divs>', '<div><p>This</div></p>', '<div><p>is not</p><div>'])

100/100 in judge

function extractText(stringArray){

console.log(stringArray.filter(s => /^<(\w+)>.+<(\/\1)>$/.test(s)).map(s => s.replace(/<\/?\w+>/g, '')).join(' '));

}

90/100 in Judge

function extractText(stringArray) {

let textArray = [];

let validStartTags = [];

let validEndTags = [];

let startTagPattern = /<\w+>/;

let endTagPattern = /<\/\w+>/;

let linePattern = /^(<\w+>.+<\/\w+>)$/;

for (let line of stringArray) {

if (!linePattern.test(line)) {

continue;

}

// let startTagCounter = 0;

// let startTagMatch;

// while (startTagMatch = startTagPattern.exec(line)) {

// startTagCounter++;

// //validStartTags.push()

// }

// let endTagCounter = 0;

// let endTagMatch;

// while (endTagMatch = endTagPattern.exec(line)) {

// endTagCounter++;

// }

// if (startTagCounter === endTagCounter){

// line = line.replace(/<\w+>/g, '');

// line = line.replace(/<\/\w+>/g, '');

// textArray.push(line);

// }

line = line.replace(/<\w+>/g, '');

line = line.replace(/<\/\w+>/g, '');

textArray.push(line);

}

let text = textArray.join(' ');

console.log(text);

return text;

}

extractText(['<h1><span>Hello World!</span></h1>', '<p>I am Peter.'])

extractText(['<div><p>This</p> is</div>', '<div><a>perfectly</a></div>', '<divs><p>valid</p></divs>', '<div><p>This</div></p>', '<div><p>is not</p><div>'])

**//100/100 in Judge**

**function** solve(input) {

    let final = [];

**for** (let string of input) {

**if** (string.match(/^([<])([\w]+[>])+(.+)(\1[/]\2)$/g)){

            final.push(string.replace(*/(<([^>]+)>)/ig*, ""))

        }

    }

    console.log(final.join(' '));

}

solve(['<div><p>This</p> is</div>',

    '<div><a>perfectly</a></div>',

    '<divs><p>valid</p></divs>',

    '<div><p>This</div></p>',

    '<div><p>is not</p><div>']

);

function solve(arr) {

    console.log(arr.filter(x => /^<(\w+)>(.+)<(\/\1)>$/.test(x)).map(x => x.replace(/<\/\*\w+>/gi, '')).join(' '));

}

function solve(arr) {

    let stringToPrint = []

    let [startReg, endReg] = [/^<[a-zA-Z0-9\_]+>/, /<\/[a-zA-Z0-9\_]+>$/]

    for (let line of arr) {

        let check = startReg.test(line) && endReg.test(line);

        let [start, end] = [line.match(startReg), line.match(endReg)];

        if (check && start[0] === end[0].replace(/\//, '')) {

            stringToPrint.push(line.replace(/<\/\*[a-zA-Z0-9\_]+>/gi, ''));

        }

    }

    console.log(stringToPrint.join(' '))

}

function solve(arr) {

console.log(arr.filter(x => /^<(\w+)>(.+)<(\/\1)>$/.test(x)).map(x => x.replace(/<\/\*\w+>/gi, '')).join(' '));

}

# Problem 4. E-Venetka.bg

function showMostProfitableObjects(objectArray){

let towns = {};

let outputText = '';

for(let vineteData of objectArray){

if(!towns.hasOwnProperty(vineteData.town)){

towns[vineteData.town] = {[vineteData.model]: [[vineteData.price], [vineteData.regNumber]]};

} else{

if(!towns[vineteData.town].hasOwnProperty(vineteData.model)){

towns[vineteData.town][vineteData.model] = [[vineteData.price], [vineteData.regNumber]]

} else{

towns[vineteData.town][vineteData.model].map((m, i) => i === 0 ? m.push(vineteData.price) : m.push(vineteData.regNumber));

}

}

}

let mostProfitableTown = Object.keys(towns).sort((a, b) => sumModelProfit(b) - sumModelProfit(a) || takeVineteSoldCount(b) - takeVineteSoldCount(a) || a.localeCompare(b))[0];

outputText += `${mostProfitableTown} is most profitable - ${sumModelProfit(mostProfitableTown)} BGN\n`;

let mostDrivenModel = Object.keys(towns[mostProfitableTown]).sort((a, b) => towns[mostProfitableTown][b][0].length - towns[mostProfitableTown][a][0].length || sortVinetePrices(towns[mostProfitableTown][b][0])[0] - sortVinetePrices(towns[mostProfitableTown][a][0])[0] || a.localeCompare(b))[0];

outputText += `Most driven model: ${mostDrivenModel}\n`;

let townsWithMostDrivenModel = Object.keys(towns).filter(t => towns[t].hasOwnProperty(mostDrivenModel)).sort();

for(let townWithMostDrivenModel of townsWithMostDrivenModel){

outputText += `${townWithMostDrivenModel}: ${(towns[townWithMostDrivenModel][mostDrivenModel][1].sort()).join(', ')}\n`;

}

outputText = outputText.substring(0, outputText.length - 1);

console.log(outputText);

function sumModelProfit(town){

let modelProfitSum = Object.keys(towns[town]).map(m => towns[town][m][0].reduce((a, b) => a + b)).reduce((a, b) => a + b);

return modelProfitSum;

}

function takeVineteSoldCount(town){

let vineteSoldCount = Object.keys(towns[town]).reduce((a, b) => towns[town][a][0].length + towns[town][b][0].length);

return vineteSoldCount;

}

function sortVinetePrices(vinetePrices){

return vinetePrices.sort((a, b) => b - a);

}

}

showMostProfitableObjects([ { model: 'BMW', regNumber: 'B1234SM', town: 'Varna', price: 2}, { model: 'BMW', regNumber: 'C5959CZ', town: 'Sofia', price: 8}, { model: 'Tesla', regNumber: 'NIKOLA', town: 'Burgas', price: 9}, { model: 'BMW', regNumber: 'A3423SM', town: 'Varna', price: 3}, { model: 'Lada', regNumber: 'SJSCA', town: 'Sofia', price: 3} ])

**function** solve(arr) {

**function** getModel(arr, town) {

        let townModels = [];

**for** (let modelRow of arr) {

**if** (modelRow.town === town) {

                let model = townModels.find(t => t.name === town);

**if** (model) {

                    model.counter++;

                } **else** {

                    let newModel = {

                        model: modelRow.model,

                        counter: 1,

                        price: modelRow.price

                    }

                    townModels.push(newModel);

                }

            }

        }

        let sorted = townModels.sort((a, b) => b.counter - a.counter || b.price - a.price || a.model.localeCompare(b.model));

**return** sorted[0];

    }

**function** getTown(arr) {

        let townsAndTotal = [];

**for** (let purchase of arr) {

            let town = townsAndTotal.find(t => t.name === purchase.town);

**if** (town) {

                town.profit += purchase.price;

                town.vignettesCount++;

            } **else** {

                let town = {

                    name: purchase.town,

                    vignettesCount: 1,

                    profit: purchase.price

                }

                townsAndTotal.push(town);

            }

        }

        let sorted = townsAndTotal.sort((a, b) => b.profit - a.profit || b.vignettesCount - a.vignettesCount ||a.name.localeCompare(b.name));

**return** sorted[0];

    }

**function** getTownsWithModel(arr, model) {

        let towns = [];

**for** (let townArg of arr) {

**if** (townArg.model === model) {

                let town = towns.find(t => t.town === townArg.town);

**if** (town) {

                    town.regs.push(townArg.regNumber);

                } **else** {

                    let town = {

                        town: townArg.town,

                        regs: []

                    };

                    town.regs.push(townArg.regNumber);

                    towns.push(town);

                }

            }

        }

        let sorted = towns.sort((a, b) =>  a.town.localeCompare(b.town));

**return** sorted;

    }

    let town = getTown(arr);

    let model = getModel(arr, town.name.toString());

    let towns = getTownsWithModel(arr, model.model);

    console.log(`${town.name} is most profitable - ${town.profit} BGN`);

    console.log(`Most driven model: ${model.model}`);

**for** (let town of towns) {

        console.log(town.town + ': ' + town.regs.sort((a, b) => a.localeCompare(b)).join(', '));

    }

}

solve([ { model: 'BMW', regNumber: 'B1234SM', town: 'Varna', price: 2},

    { model: 'BMW', regNumber: 'C5959CZ', town: 'Sofia', price: 8},

    { model: 'Tesla', regNumber: 'NIKOLA', town: 'Burgas', price: 9},

    { model: 'BMW', regNumber: 'A3423SM', town: 'Varna', price: 3},

    { model: 'Lada', regNumber: 'SJSCA', town: 'Sofia', price: 3} ]

);

function Solve(arr) {

var temp = [];

var cars = [];

for (var i = 0; i < arr.length; i++) {

if (temp.some(x => x.name == arr[i].town)) {

temp.find(x => x.name == arr[i].town).number++;

temp.find(x => x.name == arr[i].town).price += arr[i].price;

} else {

temp.push({

name: arr[i].town,

number: 1,

price: arr[i].price

});

}

if (cars.some(x => x.model == arr[i].model)) {

cars.find(x => x.model == arr[i].model).count++;

if (cars.find(x => x.model == arr[i].model).price < arr[i].price) {

cars.find(x => x.model == arr[i].model).price = arr[i].price;

}

} else {

cars.push({

model: arr[i].model,

reg: arr[i].regNumber,

count: 1,

price: arr[i].price

});

}

}

temp.sort(function (a, b) {

var p1 = a.price;

var p2 = b.price;

var n1 = a.number;

var n2 = b.number;

var t1 = a.name;

var t2 = b.name;

if (p1 < p2) return 1;

if (p1 > p2) return -1;

if (n1 < n2) return 1;

if (n1 > n2) return -1;

if (t1 < t2) return -1;

if (t1 > t2) return 1;

});

cars.sort(function (a, b) {

var c1 = a.count;

var c2 = b.count;

var p1 = a.price;

var p2 = b.price;

var n1 = a.model;

var n2 = b.model;

if (c1 < c2) return 1;

if (c1 > c2) return -1;

if (p1 < p2) return 1;

if (p1 > p2) return -1;

if (n1 < n2) return -1;

if (n1 > n2) return 1;

});

console.log(temp[0].name + ' is most profitable - ' + temp[0].price + ' BGN');

console.log('Most driven model: ' + cars[0].model);

var cities = [];

for (var i = 0; i < arr.length; i++) {

if (arr[i].model == cars[0].model) {

if (cities.some(x => x.name == arr[i].town)) {

cities.find(x => x.name == arr[i].town).reg.push(arr[i].regNumber);

} else {

cities.push({

name: arr[i].town,

reg: [arr[i].regNumber]

});

}

}

}

cities.sort(function (a, b) {

var n1 = a.name;

var n2 = b.name;

if (n1 < n2) return -1;

if (n1 > n2) return 1;

});

for (var i = 0; i < cities.length; i++) {

console.log(cities[i].name + ': ' + cities[i].reg.join(', '));

}

}

function solve(arr) {

let townObj = {};

arr.forEach(x => !townObj.hasOwnProperty(x.town) ? townObj[x.town] = {[x.model]: [[x.price], [x.regNumber]]}

: (!townObj[x.town].hasOwnProperty(x.model) ? townObj[x.town][x.model] = [[x.price], [x.regNumber]]

: townObj[x.town][x.model].map((y, i) => i === 0 ? y.push(x.price) : y.push(x.regNumber))))

let maxProfitTown = Object.keys(townObj).sort((a, b) => modelProfitSum(b) - modelProfitSum(a)

|| vigneteSold(b) - vigneteSold(a) || a.localeCompare(b))[0];

console.log(`${maxProfitTown} is most profitable - ${modelProfitSum(maxProfitTown)} BGN`)

let maxDrivvenModel = Object.keys(townObj[maxProfitTown])

.sort((a, b) => townObj[maxProfitTown][b][0].length - townObj[maxProfitTown][a][0].length

|| sortNum(townObj[maxProfitTown][b][0])[0] - sortNum(townObj[maxProfitTown][a][0])[0]

|| a.localeCompare(b))[0];

console.log(`Most driven model: ${maxDrivvenModel}`)

sortString(filterTownsForModel(maxDrivvenModel)).forEach(x => console.log(`${x}: ${sortString(townObj[x][maxDrivvenModel][1]).join(', ')}`));

function sortNum(arr) {

return arr.sort((x, y) => y - x);

}

function sortString(arr) {

return arr.sort((x, y) => x.localeCompare(y));

}

function modelProfitSum(town) {

return Object.keys(townObj[town]).map(x => townObj[town][x][0].reduce((a, b) => a + b)).reduce((a, b) => a + b);

}

function vigneteSold(town) {

return Object.keys(townObj[town]).reduce((a, b) => townObj[town][a][0].length + townObj[town][b][0].length);

}

function filterTownsForModel(carModel) {

return Object.keys(townObj).filter(x => townObj[x].hasOwnProperty(carModel));

}

}