**JavaScript Algorithms and Data Structures Projects**

**1. Palindrome Checker**

function palindrome(str) {

// Good luck!

var regex = /[\W\_]/g;

var newStr = str.toLowerCase().replace(regex, '');

var reverseStr = newStr.split('').reverse().join('');

return reverseStr === newStr;

}

palindrome("eye"); should return true

**2. Roman Numeral Converter**

function convertToRoman(num) {

var decimal = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1];

var roman = ['M', 'CM', 'D', 'CD', 'C', 'XC', 'L', 'XL', 'X', 'IX', 'V', 'IV', 'I'];

var result = '';

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

while(num%decimal[i] < num) {

result += roman[i];

num -= decimal[i];

}

}

return result;

}

convertToRoman(36);

convertToRoman(3999)should return "MMMCMXCIX"

**3. Caesar’s Cipher**

function rot13(str) { // LBH QVQ VG!

var result = [];

var currentCode;

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

currentCode = str.charCodeAt(i);

//capital letters

if(currentCode < 65 || currentCode > 90) {

result.push(str.charAt(i));

} else if (currentCode > 77) {

result.push(String.fromCharCode(currentCode - 13));

} else {

result.push(String.fromCharCode(currentCode + 13));

}

}

return result.join('');

}

// Change the inputs below to test

rot13("SERR PBQR PNZC"); should decode to FREE CODE CAMP

rot13("GUR DHVPX OEBJA SBK WHZCF BIRE GUR YNML QBT.")should decode to THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.

4. Telephone Number Validator

function telephoneCheck(str) {

var regex = /^(1\s?)?(\(\d{3}\)|\d{3})[\s\-]?\d{3}[\s\-]?\d{4}$/;

return regex.test(str);

}

telephoneCheck("555-555-5555"); should return true.

telephoneCheck("555-5555")should return false.

telephoneCheck("2 757 622-7382")should return false.

5. Cash Register

var denomination = [

{ denom: 'ONE HUNDRED', value: 100.00},

{ denom: 'TWENTY', value: 20.00},

{ denom: 'TEN', value: 10.00},

{ denom: 'FIVE', value: 5.00},

{ denom: 'ONE', value: 1.00},

{ denom: 'QUARTER', value: 0.25},

{ denom: 'DIME', value: 0.10},

{ denom: 'NICKEL', value: 0.05},

{ denom: 'PENNY', value: 0.01}

];

function checkCashRegister(price, cash, cid) {

var result = { status: null, change: [] };

var change = cash - price;

var register = cid.reduce(function(acc, currentValue) {

acc.total += currentValue[1];

acc[currentValue[0]] = currentValue[1];

return acc;

}, {total: 0});

if(register.total === change) {

result.status = "CLOSED";

result.change = cid;

return result;

}

if(register.total < change) {

result.status = "INSUFFICIENT\_FUNDS";

return result;

}

var change\_arr = denomination.reduce(function(acc, currentValue) {

var value = 0;

while (register[currentValue.denom] > 0 && change >= currentValue.value) {

change -= currentValue.value;

register[currentValue.denom] -= currentValue.value;

value += currentValue.value;

change = Math.round(change \* 100) / 100;

}

// Add denomination to the output

if (value > 0) {

acc.push([currentValue.denom, value]);

}

return acc; // Return the current change\_arr

}, []); // Initial value of empty array for reduce

// Return "Insufficient Funds" if there are no elements in change\_arr or have leftover change,

if (change\_arr.length < 1 || change > 0) {

result.status = "INSUFFICIENT\_FUNDS";

return result;

}

// Here is your change, ma'am.

result.status = "OPEN";

result.change = change\_arr;

return result;

}

// Example cash-in-drawer array:

// [["PENNY", 1.01],

// ["NICKEL", 2.05],

// ["DIME", 3.1],

// ["QUARTER", 4.25],

// ["ONE", 90],

// ["FIVE", 55],

// ["TEN", 20],

// ["TWENTY", 60],

// ["ONE HUNDRED", 100]]

checkCashRegister(19.5, 20, [["PENNY", 1.01], ["NICKEL", 2.05], ["DIME", 3.1], ["QUARTER", 4.25], ["ONE", 90], ["FIVE", 55], ["TEN", 20], ["TWENTY", 60], ["ONE HUNDRED", 100]]);

should return {status: "OPEN", change: [["QUARTER", 0.5]]}