let price = 19.5; //1.87 3.26 19.5

let cid = [

  ["PENNY", 1.01],

  ["NICKEL", 2.05],

  ["DIME", 3.1],

  ["QUARTER", 4.25],

  ["ONE", 90],

  ["FIVE", 55],

  ["TEN", 20],

  ["TWENTY", 60],

  ["ONE HUNDRED", 100]

];

/\*

let cid = [

  ["PENNY", 0.5],

  ["NICKEL", 0],

  ["DIME", 0],

  ["QUARTER", 0],

  ["ONE", 0],

  ["FIVE", 0],

  ["TEN", 0],

  ["TWENTY", 0],

  ["ONE HUNDRED", 0]];

\*/

// Object for 2nd method on calcChange()

/\*

const currency = {

  "PENNY": 0.01,

  "NICKEL": 0.05,

  "DIME": 0.1,

  "QUARTER": 0.25,

  "ONE": 1,

  "FIVE": 5,

  "TEN": 10,

  "TWENTY": 20,

  "ONE HUNDRED": 100,

}

\*/

const currency = [  ["PENNY", 0.01],

  ["NICKEL", 0.05],

[  "DIME", 0.1],

[  "QUARTER", 0.25],

[  "ONE", 1],

[  "FIVE", 5],

[  "TEN", 10],

[  "TWENTY", 20],

[  "ONE HUNDRED", 100]

] ;

/\*Creating variables------------------------------\*/

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

const purchaseBtn = document.getElementById("purchase-btn");

const priceTag = document.getElementById("price-tag");

const changeDueTag = document.getElementById("change-due");

const cashRegisterChange = document.getElementById("cash-register-change");

/\*Functions------------------------------\*/

const capitalize = (str) => str = str.charAt(0).toUpperCase() + str.substr(1).toLowerCase();

//---------------------

const totalCid = (cid) => {

  return parseFloat(cid.reduce((acc, el) => acc + el[1],0).toFixed(2));

}

//---------------------

const updateChangeCid = (arr) => {

  cashRegisterChange.innerHTML = `<br><b>Change in drawer:<b> <br>`

  for (let i in arr)

cashRegisterChange.innerHTML += `

${capitalize(arr[i][0])}: $${parseFloat(arr[i][1].toFixed(2))} <br>

`

cashRegisterChange.innerHTML += `<br><p class="totalCid">Total ($) ${totalCid(cid)}</p>` ;

}

/\*Showing total price and changes------------------------------\*/

priceTag.innerHTML = `Total: <b>$${price}<b>`

//---------------------

const cleaningInput = (input) =>

   {input = input.toString().replace(/e+/g, "");

   return Number(input)};

updateChangeCid(cid);

//---------------------

const updateChangeResultStatus = (arr) => {

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

      changeDueTag.innerHTML += `<p>${arr[i][0]}: $${parseFloat(arr[i][1].toFixed(2))}</p>`

  }

}

//---------------------

const updateStatus = (status) => {

  if (status === "open" ) {

    changeDueTag.innerHTML = `<p>Status: <span class="open status">OPEN</span></p>`

  }

 else if (status === "closed" ) {

    changeDueTag.innerHTML = `<p>Status: <span class="closed status">CLOSED</span></p>`

  }

  else if (status === "insufficient" ) {

    changeDueTag.innerHTML = `<p>Status: <span class="insufficient status">INSUFFICIENT\_FUNDS</span></p>`

  }

 else {

    changeDueTag.innerHTML = ``;

  }

}

//---------------------

const calculateCid = (cid, changeResult) => {

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

        const currencyName = changeResult[i][0];

        const amountToSubtract = changeResult[i][1];

        const currencyIndex = cid.findIndex((entry) => entry[0] === currencyName);

        if (currencyIndex !== -1) {

            cid[currencyIndex][1] -= amountToSubtract;

        }

    }

    return cid;

};

//---------------------

const calcChange = (changeToGive, cid, currency) => {

  if (totalCid(cid) < changeToGive) {

    updateStatus("insufficient");

console.log(totalCid(cid));

console.log(changeToGive);

  ;

} else if (totalCid(cid) === changeToGive) {

   updateStatus("closed");

   updateChangeResultStatus(cid);

   cid = calculateCid(cid, cid);

   updateChangeCid(cid);

}

 else {

    let changeResult = [];

   for (let i = cid.length - 1; i>=0; i-- ) {

  if (changeToGive > currency[i][1]) {

    let bills = Math.trunc(changeToGive / currency[i][1]);

    while(bills > 0) {

      let amountNec = parseFloat((bills \* currency[i][1]).toFixed(2));

      if (cid[i][1] >= amountNec) {

        changeResult.push([currency[i][0], amountNec]);

        changeToGive = parseFloat((changeToGive - amountNec).toFixed(2));

        break;

      }

      else {

        bills--;

      }

    }

  }

  //2nd method for this function

    /\*

const nameOfCurrency = cid[i][0];

    const valueOfCurrency = currency[nameOfCurrency];

    const availableAmount = cid[i][1];

    let balanceToGive = (availableAmount / valueOfCurrency).toFixed(0);

    let returnedAmount = 0;

    while (changeToGive >= valueOfCurrency && balanceToGive > 0) {

      changeToGive -= valueOfCurrency;

      changeToGive = changeToGive.toFixed(2);

      balanceToGive--;

      returnedAmount += valueOfCurrency;

    }

    if (returnedAmount > 0) {

      changeResult.push([nameOfCurrency, returnedAmount]);

    }

    if (changeToGive > 0) {

    updateStatus("insufficient");

  }\*/

    if (changeToGive !== 0) {

    updateStatus("insufficient");

  } else {

    cid = calculateCid(cid, changeResult);

    updateChangeCid(cid);

    updateStatus("open");

    updateChangeResultStatus(changeResult);

  }

 }

}

return(cid)

}

//---------------------

const checkChangeDue = (cash) => {

  cash = cleaningInput(cash);

  const changeToGive = parseFloat((cash - price).toFixed(2)); //Needs cleaning

  if (changeToGive < 0) {

    alert("Customer does not have enough money to purchase the item")

  }

   else if (changeToGive === 0) {

    changeDueTag.innerHTML = "No change due - customer paid with exact cash"

  }

  else {

   calcChange(changeToGive, cid, currency);

  }

}

/\*INPUT------------------------------\*/

/\* ADD a max property here for the HTML input with the value set as the sum of the cash register availability\*/

cash.setAttribute("max", totalCid(cid) + price)

/\*EVENTS------------------------------\*/

purchaseBtn.addEventListener("click", () => {

  checkChangeDue(cash.value);

})

cash.addEventListener("keydown", (e) => {

  if (e.key==="Enter") {

    checkChangeDue(cash.value);

  }

})