**CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS (CUCEI)**

**DIVISIÓN DE ELECTRÓNICA Y COMPUTACIÓN**

**DEPARTAMENTO DE CIENCIAS COMPUTACIONALES**

**Carrera:** Ingeniería en Computación

**Nombre Materia:** Seminario de Solución de Problemas de Sistemas Operativos

**Profesor:** Valdés López Julio Esteban

**SECCIÓN:** D08

**Nombre alumno:** López Arellano Ricardo David

**CODIGO**: 217136143


# Práctica 7

**Fecha de entrega: 12/05/2023**

# CONTENIDO

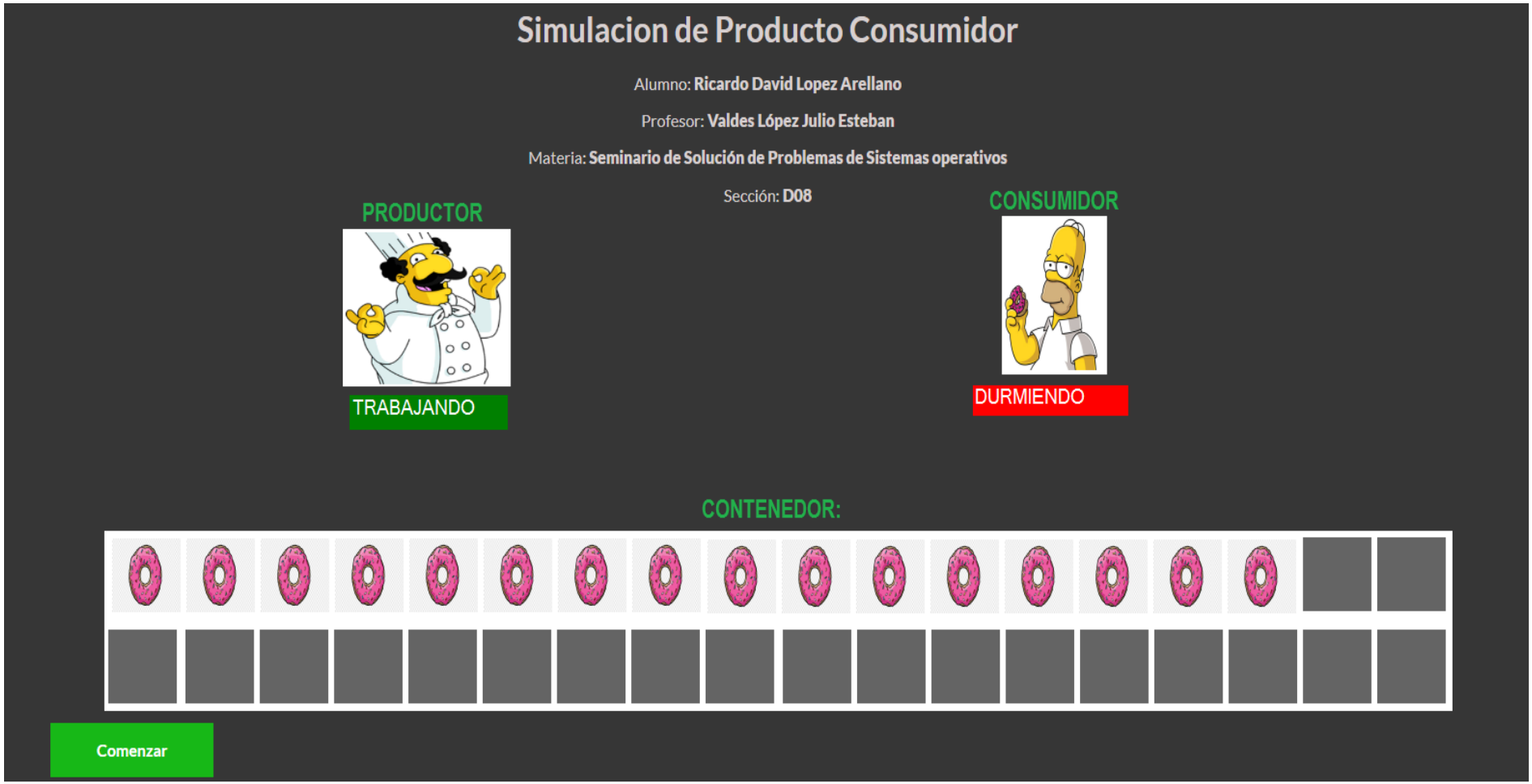
# INTRODUCCIÓN:

Todos los algoritmos se pueden ejecutar como un proceso por lotes. Es decir, que se pueden ejecutar utilizando no sólo un único conjunto de insumos, sino varios de ellos y ejecutar el algoritmo tantas veces sea necesario. Esto es útil al procesar grandes cantidades de datos, ya que no es necesario poner en marcha el algoritmo muchas veces desde la caja de herramientas.

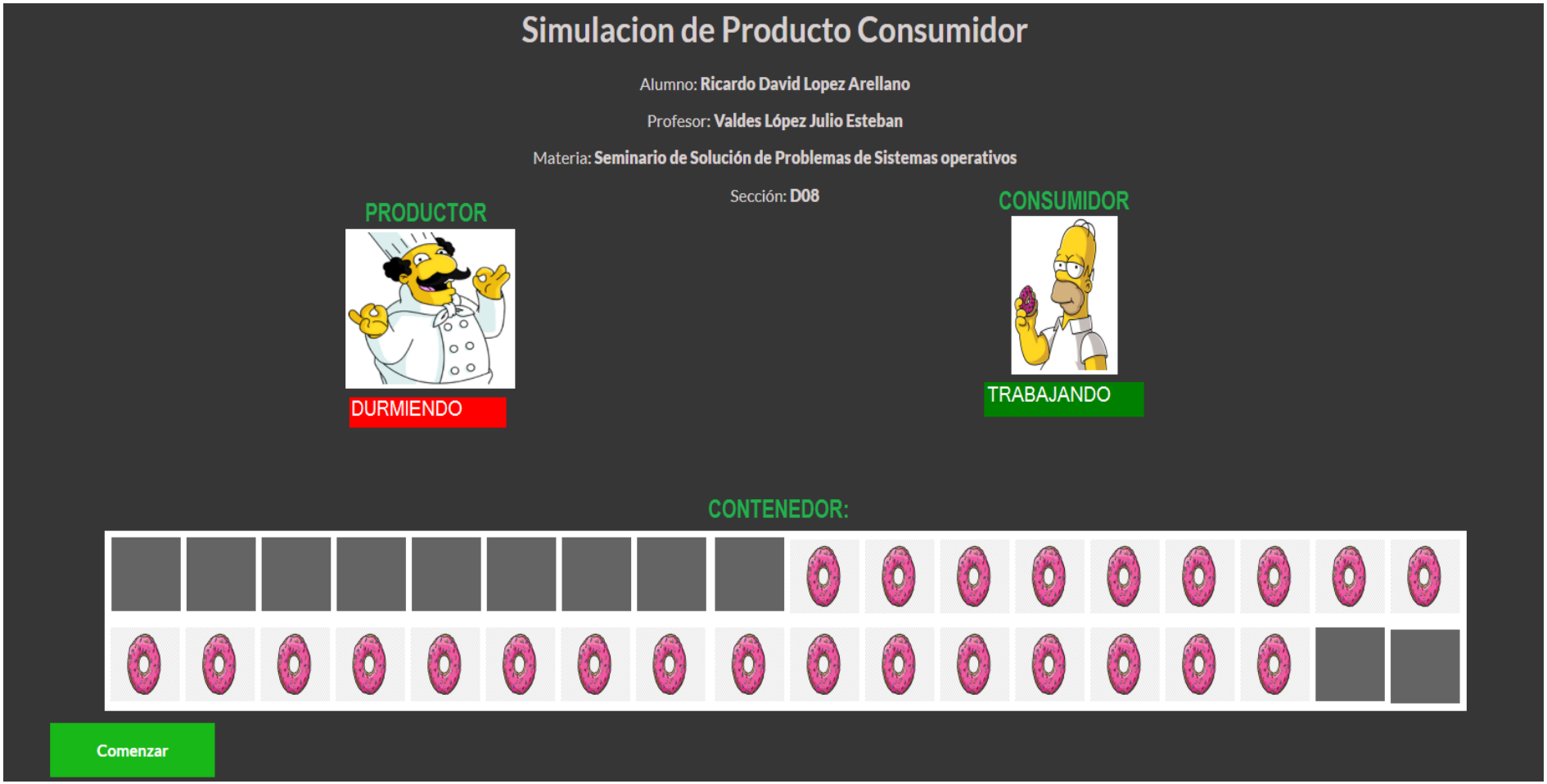
# CAPTURAS DE PANTALLA

Esta es la pantalla principal de la práctica:

Le damos en comenzar y el productor comenzará a producir donas mientras el consumidor duerme:



Después el consumidor comenzará a “comer” las donas mientras el productor duerme:



Si el contenedor se encuentra lleno el productor se quedará intentado hacer más donas y aparecerá un recuadro que dice “intentando”:

Si el contenedor se encuentra vacío el consumidor se quedará intentado comer más donas y aparecerá un recuadro que dice “intentando”:

# CÓDIGO FUENTE:

## Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="css/normalize.css">
    <link rel="stylesheet" href="css/styles.css">

    <title>Simulacion de procesamiento por lotes</title>

</head>
<body>
    <header>
        <h1>P.2: Simular el comportamiento de la Multiprogramación en un
proceso por lotes</h1>
        <p>Alumno: <span>Ricardo David López Arellano</span></p>
        <p>Profesor: <span>Valdes López Julio Esteban</span></p>
        <p>Materia: <span>Seminario de Solución de Problemas de Sistemas
operativos</span></p>
        <p>Sección: <span>D08</span></p>
    </header>
    <div class="Nproceso">

        <form class="formulario" action="">
            <fieldset>
                <center><legend>NÚMERO DE PROCESOS:  </legend></center>
                <label for="procesos">Ingrese los procesos que requiera:
</label>
                <input type="num" id="procesos" required>
            </fieldset>
        </form>
        <center><input class='boton' type="submit" id="ingresar"
value="Continuar"></center>
        <div id="errores" class='alerta'>
        </div>
    </div>
    <div class="contenedor">
        <div>
            <p id="lotePendientes"></p>
            <!--Lote en Proceso-->
            <h2>Lote en Proceso</h2>
            <table class="tabla">
```

```
                <thead>
                    <th>Lote</th>
                    <th>ID</th>
                    <th>Tiempo Máximo Estimado</th>
                </thead>
                <tbody id="loteProceso">

                </tbody>
            </table>
        </div>
        <!-- Proceso en Ejecucion -->
        <div >
            <p class='alerta'></p>
            <h2>Proceso en Ejecución</h2>
            <div id="procesoEjecucion" class="proceso">

            </div>
        </div>
        <div>
            <h2>Procesos Terminados</h2>
            <table class="tabla">
                <thead>
                    <th>Lote</th>
                    <th>ID</th>
                    <th>Operacion</th>
                    <th>Resultado</th>
                </thead>
                <tbody id="procesoTerminado">

                </tbody>
            </table>
        </div>
    </div>
    <!-- <p>Press inside this IFrame first to focus it, then try pressing
keys on the keyboard.</p>
    <p id="log"></p> -->
    <footer>
        <p id="tiempoTotal" class="ttotal"></p>
    </footer>

    <!-- <script type="module" src="js/app.js"></script> -->
    <script src="node_modules/advanced-timer/src/main.js"></script>
    <script src="js/Proceso.js"> </script>
    <script src="js/index.js"> </script>
</body>
</html>
```

## Normalize.css:

```
/*LOPEZ ARELLANO RICARDO DAVID*/

 html {
    line-height: 1.15; /* 1 */
    -webkit-text-size-adjust: 100%; /* 2 */
  }

  body {
    margin: 0;
  }

  main {
    display: block;
  }

  h1 {
    font-size: 2em;
    margin: 0.67em 0;
  }

  hr {
    box-sizing: content-box; /* 1 */
    height: 0; /* 1 */
    overflow: visible; /* 2 */
  }

  pre {
    font-family: monospace, monospace; /* 1 */
    font-size: 1em; /* 2 */
  }

  a {
    background-color: transparent;
  }

  abbr[title] {
    border-bottom: none; /* 1 */
    text-decoration: underline; /* 2 */
    text-decoration: underline dotted; /* 2 */
  }

  b,
  strong {
    font-weight: bolder;
  }

  code,
```

```
kbd,
samp {
  font-family: monospace, monospace; /* 1 */
  font-size: 1em; /* 2 */
}

small {
  font-size: 80%;
}

sub,
sup {
  font-size: 75%;
  line-height: 0;
  position: relative;
  vertical-align: baseline;
}

sub {
  bottom: -0.25em;
}

sup {
  top: -0.5em;
}

img {
  border-style: none;
}

button,
input,
optgroup,
select,
textarea {
  font-family: inherit; /* 1 */
  font-size: 100%; /* 1 */
  line-height: 1.15; /* 1 */
  margin: 0; /* 2 */
}

button,
input { /* 1 */
  overflow: visible;
}

button,
select { /* 1 */
  text-transform: none;
```

```
}

button,
[type="button"],
[type="reset"],
[type="submit"] {
  -webkit-appearance: button;
}

button::-moz-focus-inner,
[type="button"]::-moz-focus-inner,
[type="reset"]::-moz-focus-inner,
[type="submit"]::-moz-focus-inner {
  border-style: none;
  padding: 0;
}

button:-moz-focusring,
[type="button"]:-moz-focusring,
[type="reset"]:-moz-focusring,
[type="submit"]:-moz-focusring {
  outline: 1px dotted ButtonText;
}

fieldset {
  padding: 0.35em 0.75em 0.625em;
}

legend {
  box-sizing: border-box; /* 1 */
  color: inherit; /* 2 */
  display: table; /* 1 */
  max-width: 100%; /* 1 */
  padding: 0; /* 3 */
  white-space: normal; /* 1 */
}

progress {
  vertical-align: baseline;
}

textarea {
  overflow: auto;
}

[type="checkbox"],
[type="radio"] {
  box-sizing: border-box; /* 1 */
  padding: 0; /* 2 */
```

```
  }

  [type="number"]::-webkit-inner-spin-button,
  [type="number"]::-webkit-outer-spin-button {
    height: auto;
  }

  [type="search"] {
    -webkit-appearance: textfield; /* 1 */
    outline-offset: -2px; /* 2 */
  }

  [type="search"]::-webkit-search-decoration {
    -webkit-appearance: none;
  }

  ::-webkit-file-upload-button {
    -webkit-appearance: button; /* 1 */
    font: inherit; /* 2 */
  }

  details {
    display: block;
  }

  summary {
    display: list-item;
  }

  template {
    display: none;
  }

  [hidden] {
    display: none;
  }
```

## Styles.css:

```
/*LOPEZ ARELLANO RICARDO DAVID*/
@import
url("https://fonts.googleapis.com/css?family=Lato:300,400,700,900");
html{
    line-height:1.15;
    -webkit-text-size-adjust:100%
}

header{
    background-color: rgb(54, 54, 54);
    padding: 1rem 0 3rem 0;
```

```
    text-align: center;
}

body{
    margin: 0;
    font-family: 'Lato', sans-serif;
    background-color: rgb(147, 232, 253);
}

h1, h2, p, span{
    text-align: center;
    color: rgb(0, 0, 0);
}

span{
    font-weight: 900;
}

.Nproceso{
    width: 95%;
    max-width: 120rem;
    margin: 0 auto;
}

.formulario{
    color: rgb(0, 0, 0);
}

.formulario legend{
    font-size: 1.2rem;
    font-weight: 700;
}

.formulario label{
    margin-top: 1rem;
    font-weight: 700;
    text-transform: uppercase;
    display: block;
}

.formulario input:not([type="submit"]){
    padding: .5rem;
    display: block;
    width: 99%;
    background-color: white;
    border: 2px solid black;
    border-radius: .5rem;
}
```

```css
.boton{
    background-color: rgb(0, 0, 0);
    color: white;
    text-decoration: none;
    font-weight: 700;
    padding: 1rem 3rem;
    text-align: center;
    margin-top: .5rem;
    display: inline-block;
    border: none;
    cursor: pointer;
}

.boton:hover{
    background-color: rgba(0, 0, 0, 0.87);
}

.contenedor{
    display:grid;
    gap: 3rem;
    grid-template-columns: repeat(3, 1fr);
}

.tabla {
    width: 80%;
    border-spacing: 0;
    margin: 0 auto;
    color: rgb(0, 0, 0);
}

.tabla thead{
    background-color: rgba(0, 0, 0, 0.87);
}

.tabla td{
    margin-left: 1rem;
    text-align: center;
}

.ttotal{

    margin-top: 2rem;
    font-size: 1.2rem;
    text-align: center;
}

.alerta{
    width: auto;
    background-color: rgb(139, 11, 11);
```

```
}
footer{
    padding: 1rem 0 3rem 0;
    background-color: rgb(147, 232, 253);
}
```

## Index.js:

```
//LOPEZ ARELLANO RICARDO DAVID

// Cronómetros
const GLOBAL_TIMER = new Timer(1000);
const PROCESS_TIMER = new Timer(1000);

// Lista de procesos
let lotes= [];

// Detener proceso en ejecución
let abortController;
let abortSignal;

let pause;
let numeroLote = 0;
let numeroProcesos = 0;

let loteActual;
let proceso;


document.addEventListener('DOMContentLoaded', function(){
    addEventListener();
});

function addEventListener(){
    const botonIngresar =  document.querySelector('#ingresar');
    botonIngresar.addEventListener('click', ObtenerNumeroProcesos);

    document.addEventListener('keydown', (e)=>{
        const alerta = document.querySelector('.alerta');
        switch(e.key){
            case 'e':
                console.log('error');
                if(!pause){
                    abortController.abort();
                    proceso.resultado = 'ERROR';
                    loteActual.shift();
                    ProcesosTerminados();
                }
```

```
                break;
            case 'i':
                    alerta.innerHTML = `PROCESO INTERRUPIDO`;
                    console.log('interrupcion');
                    if(!pause && loteActual.length > 1){
                        abortController.abort();
                        loteActual.push(loteActual.shift());
                        actualizarTabla();
                    }
                break;
            case 'p':
                    alerta.innerHTML = `PAUSE`;
                    console.log('Pause');
                    pause = true;
                    GLOBAL_TIMER.pause();
                    PROCESS_TIMER.pause();
                break;
            case 'c':
                    alerta.innerHTML = ` `;
                    console.log('Continua');
                    pause = false;
                    GLOBAL_TIMER.resume();
                    PROCESS_TIMER.resume();
                break;

        }
    });
}

function ObtenerNumeroProcesos(){
    const procesos = Number(document.querySelector('#procesos').value);
    //Valida si es un numero mayor a 0
    if(procesos){
       ingresarProceso(procesos);
    }else{
        alert('Debe ser un numero mayor a 0');
    }
}

function crearProcesos(id){
    const nombres = ['David', 'Karla', 'Esteban', 'Julio',
'Ricardo','Carlos','Juan','Benito'];
    const nombre = nombres[randomNumber(0, 8)];
    const tiempo = randomNumber(7, 18);
    const operacion = generarOperacion();

    const errores = validar(operacion);

    if(isEmpty(errores)){
```

```
        const proceso =  new Proceso(id, nombre, tiempo, operacion);
        proceso.RealizarOperacion();
        return proceso;
    }
    else{
        crearProcesos(id);
    }
}

function ingresarProceso(procesos){
    let lote = [];
    for(let i = 0; i<procesos; i++){
        const proceso = crearProcesos(i+1);
        lote.push(proceso);
    }
    dividirLotes(lote);
    run();
}

function generarOperacion(){
    let operacion;
    const operado1 = randomNumber();
    const operado2 = randomNumber();
    switch(randomNumber(1, 6)){
        case 1:
            operacion = '+'
            break;
        case 2:
            operacion = '-'
            break;
        case 3:
            operacion = '/'
            break;
        case 4:
            operacion = '*'
            break;
        case 5:
            operacion = '%'
            break;
        case 6:
            operacion = 'x'
            break;
    }
    return `${operado1}${operacion}${operado2}`;
}

async function run(){
    abortController = new AbortController();
    GLOBAL_TIMER
```

```
        .action(t => {
            let m = Math.floor(t.currentCycle / 60);
            let s = Math.floor(t.currentCycle % 60);
            document.getElementById('tiempoTotal').innerHTML = `Tiempo
Global: <span>${checkTime(m)} : ${checkTime(s)}</span> minutos`;
        })
        .start();
    await ejecutar();

    console.log('termine');

    GLOBAL_TIMER.destroy();
    PROCESS_TIMER.destroy();
}

async function ejecutar(){
    const tablaProceso = document.querySelector('#loteProceso');
    //Lotes
    // for(const lote of lotes){
    while(lotes.length){
        loteActual = lotes.shift();
        document.getElementById('lotePendientes').innerHTML = `Lotes
Pendientes: ${lotes.length}`;
        numeroLote++;
        // for(proceso of lote){
        while(loteActual.length){
            actualizarTabla();
            abortController = new AbortController();
            abortSignal = abortController.signal;
            proceso = loteActual[0];
            actualizarPoceso();
            try{
                await procesoEjecucion(abortSignal);
                ProcesosTerminados();
            }catch(err){
                console.warn(err);
                continue
            }finally{
                PROCESS_TIMER.reset();
                delete abortController;
            }
           loteActual.shift();
        }
        tablaProceso.innerHTML = ``;
    }
}

async function procesoEjecucion(signal){
    const tiempos = document.querySelectorAll('#tiempo');
```

```
    let i = 0;
    return new Promise((resolve, reject)=>{
        PROCESS_TIMER
            .action(t => {
                tiempos[0].innerHTML = `Tiempo Trascurrido: <span>${i}
segundos</span>`;
                tiempos[1].innerHTML = `Tiempo Restante:
<span>${proceso.tiempoEstimado - i} segundos </span>`;
                ++i;
            })
            .repeat((proceso.tiempoEstimado - i))
            .done(resolve)
            .start();

        signal.addEventListener('abort', (error) => {
            PROCESS_TIMER.reset();
            PROCESS_TIMER.stop();
            reject(error);
        });
    });
}

function actualizarTabla(){
    const tablaProceso = document.querySelector('#loteProceso');
    tablaProceso.innerHTML = ` `;
    for(const process of loteActual){
        const row = document.createElement('TR');
        row.innerHTML = `
            <td>${numeroLote}</td>
            <td>${process.id}</td>
            <td>${process.tiempoEstimado}</td>
        `;
        tablaProceso.appendChild(row);
    }
}

function ProcesosTerminados(){
    const alert = document.querySelector('.alerta');
    const tbody  = document.querySelector('#procesoTerminado');
    const row = document.createElement('TR');
    alert.innerHTML = ` `;
    row.innerHTML = `
        <td>${numeroLote}</td>
        <td>${proceso.id}</td>
        <td>${proceso.operacion}</td>
        <td>${proceso.resultado}</td>
    `;
    tbody.appendChild(row);
}
```

```
function actualizarPoceso(){
    const divProceso = document.querySelector('#procesoEjecucion');
     divProceso.innerHTML = `
        <p>ID: <span>${proceso.id}</span></p>
        <p>Nombre: <span>${proceso.nombre}</span></p>
        <p>Operacion: <span>${proceso.operacion}</span></p>
        <p>Tiempo Max. Estimado:
<span>${proceso.tiempoEstimado}</span></p>
        <p id = "tiempo"></p>
        <p id = "tiempo"></p>
     `;
}

function dividirLotes(lote){
    for (let i = 0; i < lote.length; i+=3){
        const pedazo = lote.slice(i, i + 3);
        lotes.push(pedazo);
    }
}

function IsValidDivision(operacion){ //2/2
    const operador = operacion.match('[+-/%\*x]{1,1}') //Obtiene
operacion +, -, *, 7, %
    if( operador[0] ==='/' || operador [0]==='%' ){
        const operando = validarOperando(operacion, operador);
        if(operando === '0'){
           return false;
        }
    }
    return true
}

//Obtiene los dos operandos y retorna el segundo para despues validar si
es 0
function validarOperando (operacion, operando){
    const operandos =  operacion.split(operando);
    return operandos[1];
}

function validar(operacion){
    let errores = []
    if(!IsValidDivision(operacion))
        errores.push('El formato de Division o Residuo no es valido.');
    return errores;
}

function isEmpty(array){
    return array.length === 0;
```

```
}

function randomNumber(min = 0, max=100){
    return Math.floor(Math.random() * (max - min)) + min;
}

const checkTime = (int) => {
    if (int < 10) { int = '0' + int };
    return int;
}
```

## Proceso.js:

```
// Declaramos la clase
function Proceso(id, nombre, tiempoEstimado, operacion){
    //Operacion resultado operacion y resultado

    this.resultado;
    this.operador;
    this.operandos = [];

    //Atributos
    this.id = id;
    this.nombre = nombre;
    this.tiempoEstimado = tiempoEstimado;
    this.operacion = operacion;


    this.ToString = function(){
        return `Datos del Proceso: ${this.id}, ${this.nombre},
${this.tiempoEstimado}, ${this.operacion} = ${this.resultado}`;
    }

    this.ObtenerOperador = function(){
        this.operador = this.operacion.match('[+-/%\*x]{1,1}');
    }

    this.ObtenerOperandos = function(){
        const operando =  this.operacion.split(this.operador);
        for(let i=0 ; i<2; i++){
            this.operandos[i] = parseInt(operando[i]);
        }
    }

    this.RealizarOperacion =  function(){
        this.ObtenerOperador();
        this.ObtenerOperandos();
```

```
        switch(this.operador[0]){
            case '+':
                    this.resultado = this.operandos[0] +
this.operandos[1];
                break;
            case '-':
                    this.resultado = this.operandos[0] -
this.operandos[1];
                break;
            case '/':
                try{
                    this.resultado = Math.round((this.operandos[0] /
this.operandos[1]) * 100) /100;
                }catch(error){
                    alert('Agregaste una division entre 0')
                    console.log(error);
                }
                break;
            case '*':
                    this.resultado = this.operandos[0] *
this.operandos[1];
                break;
            case 'x':
                    this.resultado = this.operandos[0] *
this.operandos[1];
                break;
            case '%':
                    this.resultado =  Math.round((this.operandos[0] %
this.operandos[1]) * 100) /100
                break;
        }
    }
}
```

## Prueba.js:

```
const log = document.getElementById('log');

document.addEventListener('keypress', logKey);

function logKey(e) {
  log.textContent += ` ${e.code}`;
}
```

# CONCLUSIÓN:

En conclusión a esta práctica puedo decir que no fue tan complicada ya que es la continuación de una práctica anterior y solo tuve que implementarle nuevas teclas y la función para que vaya agregando procesos conforme le vayamos indicando al programa.

Lo que podría ser lo más complicado de la práctica es hacer lo de los contadores ya que todo el tiempo este uno detrás del otro consecutivamente sin que paren, de ahí en más es como hacer una calculadora que te va midiendo el tiempo en que ejecuta sus acciones.