INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	*Fundamentos de la programación II*				
TÍTULO DE LA PRÁCTICA:	*Combinando Arreglos Estándar y ArrayList*				
NÚMERO DE PRÁCTICA:	*07*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*02*
FECHA DE PRESENTACIÓN	*15/11/2024*	**HORA DE PRESENTACIÓN**	*19/59/00*		
INTEGRANTE (s) *José León Enrique Hatches Curo*				**NOTA (0-20)**	*Nota colocada por el docente*
DOCENTE(s): *Ing. Lino Pinto Oppe*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS RESUELTOS:

El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.
El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado.

Repositorio en GitHub:

https://github.com/LeonHatches/FP2-Laboratorios/tree/main/Laboratorio07

VIDEOJUEGO 4:

Tendrá 2 Ejércitos (utilizar la estructura de datos más adecuada). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados y sus puntos de vida (usar caracteres como | _ y otros y distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 Marco Aedo López 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacer el programa iterativo.

- *Primero, este es el main con todos los métodos y un ganador por la vida total del ejército; iterativo.*

```java
public static void main (String [] args)
{
    int decision = 1;

    do
    {
        ArrayList <Soldado> ejercito1 = new ArrayList <Soldado>();
        ArrayList <Soldado> ejercito2 = new ArrayList <Soldado>();
        Soldado [][]          tablero = new Soldado [10][10];


        // NOTA IMPORTANTE: Los ejercitos se ubicaran al final como STRINGS, 1 y 2 | Luego la vida (HP)
        ejercito1 = crear (tablero, "1");
        ejercito2 = crear (tablero, "2");
        mostrarTabla    (tablero);


        // Mostrar los de mayor vida segun el ejercito
        mostrarMayorVida    (tablero, "1");
        mostrarMayorVida    (tablero, "2");

        // Vida promedio de cada ejercito
        double vida1 = mostrarPromedioVida (tablero, "1");
        double vida2 = mostrarPromedioVida (tablero, "2");

        // MOSTRAR VIDA DE TODOS LOS SOLDADOS
        System.out.println ("\n| VIDA TOTAL DEL EJERCITO 1 |");
        System.out.println ("La vida total es: " + vida1 );

        System.out.println ("\n| VIDA TOTAL DEL EJERCITO 2 |");
        System.out.println ("La vida total es: " + vida2 );

        // ArrayList estandar para ordenamientos
        System.out.println("\n| SOLDADOS - EJERCITO 1 - ORDEN DE CREACION |");
        mostrar (ejercito1);

        System.out.println("\n| SOLDADOS - EJERCITO 2 - ORDEN DE CREACION |");
        mostrar (ejercito2);

        // RANKING DEL EJERCITO 1
        System.out.println("\n| SOLDADOS - EJERCITO 1 - RANKING DE VIDA MAYOR A MENOR |");
        mostrarRankingMayor (ejercito1);
        System.out.println("\n| SOLDADOS - EJERCITO 1 - RANKING DE VIDA MENOR A MAYOR |");
        mostrarRankingMenor (ejercito1);

        // RANKING DEL EJERCITO 2
        System.out.println("\n| SOLDADOS - EJERCITO 2 - RANKING DE VIDA MAYOR A MENOR |");
        mostrarRankingMayor (ejercito2);
        System.out.println("\n| SOLDADOS - EJERCITO 2 - RANKING DE VIDA MENOR A MAYOR |");
        mostrarRankingMenor (ejercito2);

        // GANADOR
        if (vida1 == vida2)
            System.out.println("\n| EMPATE POR VIDA DEL EJERCITO IGUALES |");

        else if (vida1 > vida2)
            System.out.println("\n| EJERCITO 1 - GANADOR POR MAYOR VIDA DEL EJERCITO |");

        else
            System.out.println("\n| EJERCITO 2 - GANADOR POR MAYOR VIDA DEL EJERCITO |");

        decision = iteracion();
    }
    while (decision == 1);
}
```

- *En este método se muestra la creación y verificación del objeto Soldado con arreglo bidimensional, además la inicialización de un ArrayList para los ordenamientos.*

```
public static ArrayList <Soldado> crear (Soldado [][] tablero, String e)
{
    int                aleatorio = posicionRandom()+1, fila, columna;
    ArrayList <Soldado> ejercito = new ArrayList <Soldado>();

    for (int i = 0 ; i < aleatorio ; i++)
    {
        // VERIFICA POSICION LIBRE
            do
            {
                fila    = posicionRandom();
                columna = posicionRandom();
            }
            while ( tablero[fila][columna] != null );

        // CREA E INICIALIZA DATOS DEL SOLDADO
        tablero[fila][columna] = new Soldado();
        inicializar (tablero, fila, columna, i, e);

        // ARRAYLIST ESTANDAR PARA TRABAJAR CON ORDENAMIENTO
        ejercito.add( tablero[fila][columna] );
    }
    return ejercito;
}
```

- *Luego, este método inicializa los objetos con un nombre, vida, ejército, fila y columna según la tabla del enunciado.*

```
public static void inicializar (Soldado [][] tablero, int f, int c, int cont, String e)
{
    String columnaLetras [] = {"A","B","C","D","E","F","G","H","I","J"};

    tablero[f][c].setNombre  ("Soldado"+cont+"X"+e);
    tablero[f][c].setVida    ( vida() );
    tablero[f][c].setFila    (f+1);
    tablero[f][c].setColumna (columnaLetras[c]);
    tablero[f][c].setEjercito(e);
}
```

- *Luego, este método muestra la tabla con los objetos (solo nombre y vida).*

```
public static void mostrarTabla (Soldado [][] tablero)
{
    String letras [] = {"A","B","C","D","E","F","G","H","I","J"};
    System.out.print ("          ");

    // MUESTRA LAS COLUMNAS
    for (int i = 0 ; i < letras.length ; i++)
    {
        System.out.print (letras[i]+"                  ");
    }
    System.out.println(" ");

    for (int i = 0 ; i < tablero.length ; i++)
    {
        // MUESTRA LAS FILAS
        System.out.print (i+1);
        for (int j = 0 ; j < tablero[i].length ; j++)
        {
            if (tablero[i][j] != null)
                System.out.print (" "+tablero[i][j].getNombre()+" HP: "+tablero[i][j].getVida()+" |");

            else
                System.out.print ("                  |");
        }

        // MUSTRA LAS LINEAS HORIZONTALES
        System.out.println();

        for (int n = 0 ; n < 191 ; n++)
            System.out.print("-");

        System.out.println();
    }
}
```

- *Después, este método muestra al soldado con mayor vida por ejército.*

```
// Uso de metodos con el Arreglo bidimensional para la demostracion de conocimiento en este mismo.
public static void mostrarMayorVida (Soldado [][] tablero, String e)
{
    int fila = 0, columna = 0;

    // LOS MAYORES VALORES INICIALES
    for (int i = 0 ; i < tablero.length ; i++) {
        for (int j = 0 ; j < tablero[i].length ; j++)
        {
                if ( !(tablero[i][j] == null) && tablero[i][j].getEjercito().equals(e) )
                { fila = i; columna = j; break; }
        }
    }

   // BUSQUEDA DEL SOLDADO CON MAYOR VIDA
   for (int i = 0 ; i < tablero.length ; i++)
   {
        for (int j = 0 ; j < tablero[i].length ; j++){
            if ( !(tablero[i][j] == null) && tablero[i][j].getEjercito().equals(e) ) {
                if ( tablero[fila][columna].getVida() < tablero[i][j].getVida() )
                {
                    fila    = i;
                    columna = j;
                }
            }
        }
    }

    // MOSTRAR
    System.out.println ("\n\t| SOLDADO CON MAYOR VIDA DEL EJERCITO "+e+" |\n");
    System.out.println (tablero[fila][columna]+"\n");
}
```

- *Luego, este método hace el promedio y devuelve la vida total del ejército.*

```
public static double mostrarPromedioVida (Soldado [][] tablero, String e)
{
    double suma = 0, cont = 0;

    // PROMEDIO DE VIDA
    for (int i = 0 ; i < tablero.length ; i++)
    {
         for (int j = 0 ; j < tablero[i].length ; j++){
             if ( !(tablero[i][j] == null) && tablero[i][j].getEjercito().equals(e) )
             {
                  suma += tablero[i][j].getVida();
                  cont++;
             }
         }
    }

     // MOSTRAR
     System.out.println ("\n| PROMEDIO DE VIDA DEL EJERCITO "+e+" |");
     System.out.println ("El promedio de vida es: " + (suma/cont) );

     return suma;
}
```

- *Este método pregunta para la iteración.*

```
public static int iteracion ()
{
    Scanner sc = new Scanner (System.in);
    System.out.print ("\n¿Desea otra batalla? | Si: 1 | No: 0 |: ");
    return sc.nextInt();
}
```

- *Después, este método se encarga de mostrar el ranking del mayor al menor por inserción.*

```
// Desde ahora, metodos con ArrayList estandar para trabajar el ordenamiento o metodos que lo requieran.
public static void mostrarRankingMayor (ArrayList<Soldado> ejercito)
{
    Soldado intercambio;

    // Algoritmo de ordenamiento Insercion
    for (int i = 1 ; i < ejercito.size() ; i++)
    {
        for (int r = i ; r > 0 && ejercito.get(r-1).getVida() < ejercito.get(r).getVida() ; r--)
        {
                  intercambio = ejercito.remove(r-1);
                  ejercito.add(r, intercambio);
        }
    }

    mostrar (ejercito);
}
```

- *En este método hace un ranking de menor a mayor mediante ordenamiento burbuja.*

```
public static void mostrarRankingMenor (ArrayList<Soldado> ejercito)
{
    Soldado intercambio;

    // Algoritmo de ordenamiento Burbuja
    for (int i = ejercito.size() - 1 ; i > 0 ; i--)
        for (int n = 0 ; n < i ; n++)
        {
            if ( ejercito.get(n).getVida() > ejercito.get(n+1).getVida() )
            {
                intercambio = ejercito.remove(n);
                ejercito.add(n+1, intercambio);
            }
        }

    mostrar(ejercito);
}
```

- *Por último, este método muestra los soldados del tablero en lista.*

```
public static void mostrar (ArrayList<Soldado> ejercito)
{
    for (int i = 0 ; i < ejercito.size() ; i++)
        System.out.println("\t| SOLDADO N-"+(i+1)+" |\n"+ejercito.get(i)+"\n");
}
```

II. PRUEBAS

EJECUCIÓN:

GIT:

- *Aquí hice mi primer commit que tiene "Soldado.java" y "VideoJuego4.java", en donde se combina arreglos bidimensionales y ArrayList hasta mostrar tabla.*

```
Equipo@DESKTOP-VTEJTT1 MINGW32 ~/Desktop/repositorio/L06/FP2-Laboratorios (main)
$ git commit -m "Lab - 07 | Version hasta mostrar tabla"
[main f7d58b6] Lab - 07 | Version hasta mostrar tabla
 2 files changed, 361 insertions(+)
 create mode 100644 Laboratorio07/Soldado.java
 create mode 100644 Laboratorio07/VideoJuego4.java
```

- *Este es mi 2do commit, en el cual tiene métodos con arreglo bidimensional trabajados.*

```
Equipo@DESKTOP-VTEJTT1 MINGW32 ~/Desktop/repositorio/L06/FP2-Laboratorios (main)
$ git add .

Equipo@DESKTOP-VTEJTT1 MINGW32 ~/Desktop/repositorio/L06/FP2-Laboratorios (main)
$ git commit -m "Lab - 07 | Version hasta el manejo del arreglo bidimensional"
[main 890d91d] Lab - 07 | Version hasta el manejo del arreglo bidimensional
 2 files changed, 57 insertions(+), 69 deletions(-)
```

- *Este es mi 3er commit con arreglos bidimensionales y ArrayList combinados.*

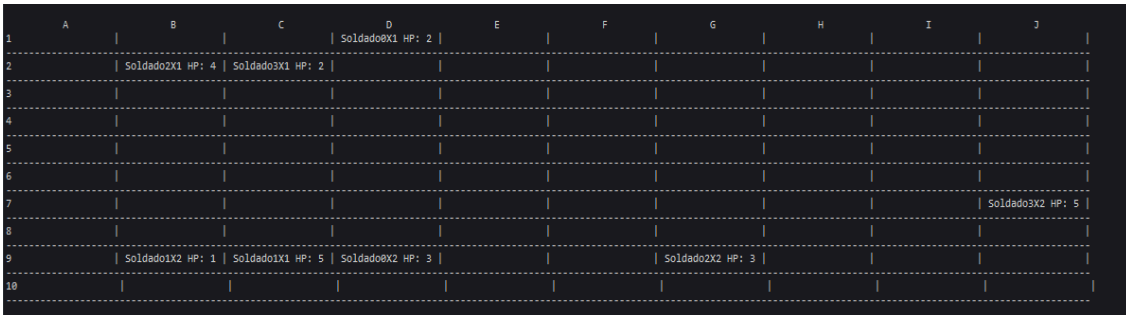
```
Equipo@DESKTOP-VTEJTT1 MINGW32 ~/Desktop/repositorio/L06/FP2-Laboratorios (main)
$ git commit -m "Lab - 07 | Version completa sin iteracion"
[main 1bf03d9] Lab - 07 | Version completa sin iteracion
 1 file changed, 4 insertions(+), 59 deletions(-)
```

- *Por último, este es mi 4to commit con la resolución del laboratorio (versión final iterativa).*

```
Equipo@DESKTOP-VTEJTT1 MINGW32 ~/Desktop/repositorio/L06/FP2-Laboratorios (main)
$ git commit -m "Lab -  07 | Version final iterativa"
[main 9676066] Lab -  07 | Version final iterativa
 1 file changed, 69 insertions(+), 55 deletions(-)
```

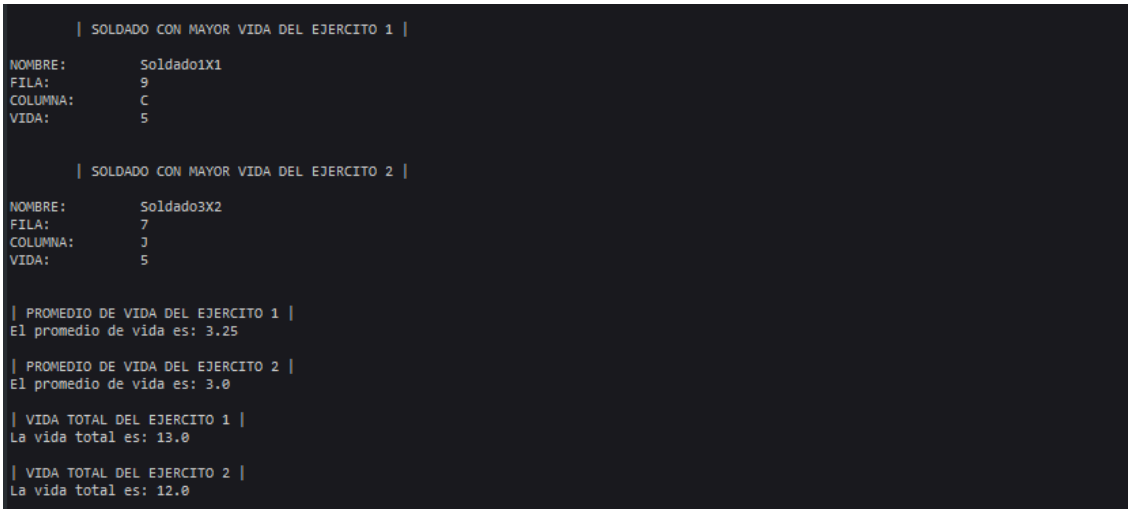
EJECUCIÓN:

- *Tablero de objetos "Soldado".*



	A	B	C	D	E	F	G	H	I	J
1				Soldado0X1 HP: 2						
2		Soldado2X1 HP: 4	Soldado3X1 HP: 2							
3										
4										
5										
6										
7										Soldado3X2 HP: 5
8										
9		Soldado1X2 HP: 1	Soldado1X1 HP: 5	Soldado0X2 HP: 3			Soldado2X2 HP: 3			
10										

- *Se muestra soldado con mayor vida, con promedio de vida y la vida total por ejército.*



```
        | SOLDADO CON MAYOR VIDA DEL EJERCITO 1 |

NOMBRE:         Soldado1X1
FILA:           9
COLUMNA:        C
VIDA:           5


        | SOLDADO CON MAYOR VIDA DEL EJERCITO 2 |

NOMBRE:         Soldado3X2
FILA:           7
COLUMNA:        J
VIDA:           5


| PROMEDIO DE VIDA DEL EJERCITO 1 |
El promedio de vida es: 3.25

| PROMEDIO DE VIDA DEL EJERCITO 2 |
El promedio de vida es: 3.0

| VIDA TOTAL DEL EJERCITO 1 |
La vida total es: 13.0

| VIDA TOTAL DEL EJERCITO 2 |
La vida total es: 12.0
```

- *Se muestra el arreglo en orden de creación.*

```
| SOLDADOS - EJERCITO 1 - ORDEN DE CREACION |
        | SOLDADO N-1 |
NOMBRE:         Soldado0X1
FILA:           1
COLUMNA:        D
VIDA:           2

        | SOLDADO N-2 |
NOMBRE:         Soldado1X1
FILA:           9
COLUMNA:        C
VIDA:           5

        | SOLDADO N-3 |
NOMBRE:         Soldado2X1
FILA:           2
COLUMNA:        B
VIDA:           4

        | SOLDADO N-4 |
NOMBRE:         Soldado3X1
FILA:           2
COLUMNA:        C
VIDA:           2


| SOLDADOS - EJERCITO 2 - ORDEN DE CREACION |
        | SOLDADO N-1 |
NOMBRE:         Soldado0X2
FILA:           9
COLUMNA:        D
VIDA:           3

        | SOLDADO N-2 |
NOMBRE:         Soldado1X2
FILA:           9
COLUMNA:        B
VIDA:           1

        | SOLDADO N-3 |
NOMBRE:         Soldado2X2
FILA:           9
COLUMNA:        G
VIDA:           3

        | SOLDADO N-4 |
NOMBRE:         Soldado3X2
FILA:           7
COLUMNA:        J
VIDA:           5
```

- *Se muestra el ranking de mayor a menor según su vida.*

```
| SOLDADOS - EJERCITO 1 - RANKING DE VIDA MAYOR A MENOR |
        | SOLDADO N-1 |
NOMBRE:         Soldado1X1
FILA:           9
COLUMNA:        C
VIDA:           5

        | SOLDADO N-2 |
NOMBRE:         Soldado2X1
FILA:           2
COLUMNA:        B
VIDA:           4

        | SOLDADO N-3 |
NOMBRE:         Soldado0X1
FILA:           1
COLUMNA:        D
VIDA:           2

        | SOLDADO N-4 |
NOMBRE:         Soldado3X1
FILA:           2
COLUMNA:        C
VIDA:           2
```

```
| SOLDADOS - EJERCITO 2 - RANKING DE VIDA MAYOR A MENOR |
        | SOLDADO N-1 |
NOMBRE:         Soldado3X2
FILA:           7
COLUMNA:        J
VIDA:           5

        | SOLDADO N-2 |
NOMBRE:         Soldado0X2
FILA:           9
COLUMNA:        D
VIDA:           3

        | SOLDADO N-3 |
NOMBRE:         Soldado2X2
FILA:           9
COLUMNA:        G
VIDA:           3

        | SOLDADO N-4 |
NOMBRE:         Soldado1X2
FILA:           9
COLUMNA:        B
VIDA:           1
```

- *Y por último, se muestra el ranking de menor a mayor según su vida.*

```
| SOLDADOS - EJERCITO 1 - RANKING DE VIDA MENOR A MAYOR |
        | SOLDADO N-1 |
NOMBRE:         Soldado0X1
FILA:           1
COLUMNA:        D
VIDA:           2

        | SOLDADO N-2 |
NOMBRE:         Soldado3X1
FILA:           2
COLUMNA:        C
VIDA:           2

        | SOLDADO N-3 |
NOMBRE:         Soldado2X1
FILA:           2
COLUMNA:        B
VIDA:           4

        | SOLDADO N-4 |
NOMBRE:         Soldado1X1
FILA:           9
COLUMNA:        C
VIDA:           5
```

```
| SOLDADOS - EJERCITO 2 - RANKING DE VIDA MENOR A MAYOR |
        | SOLDADO N-1 |
NOMBRE:         Soldado1X2
FILA:           9
COLUMNA:        B
VIDA:           1

        | SOLDADO N-2 |
NOMBRE:         Soldado0X2
FILA:           9
COLUMNA:        D
VIDA:           3

        | SOLDADO N-3 |
NOMBRE:         Soldado2X2
FILA:           9
COLUMNA:        G
VIDA:           3

        | SOLDADO N-4 |
NOMBRE:         Soldado3X2
FILA:           7
COLUMNA:        J
VIDA:           5
```

- *Por último, se muestra el ejército ganador y la pregunta para la iteración.*

```
| EJERCITO 1 - GANADOR POR MAYOR VIDA DEL EJERCITO |

¿Desea otra batalla? | Si: 1 | No: 0 | :
```

III. CUESTIONARIO:

Colocar la evidencia de las respuestas realizadas al cuestionario enunciado en la guía práctica de laboratorio.

Preguntas:

¿Con qué valores comprobaste que tu práctica estuviera correcta?

- *En este ejercicio, comprobé mediante datos o valores inicializados de tipo String e int, como también un int para la iteración.*

¿Qué resultado esperabas obtener para cada valor de entrada?

- *Esperaba los resultados de mostrar un soldado con mayor vida, promedio de vida y suma total de vida por ejército y mostrar los soldados con su respectiva vida según el algoritmo de ordenamiento.*

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

- *Obtuve los valores que deseaba para satisfacer el enunciado.*

CONCLUSIONES

En conclusión, para que el código del laboratorio pueda tener un buen funcionamiento, el uso de arreglos bidimensionales y ArrayList serán necesarios especialmente en el tablero y el uso de objetos para implementar los datos de los soldados. Además, la importancia del uso de métodos que devuelvan los ArrayList ordenados de diferentes formas, así mostremos gran optimización, accesibilidad y orden al momento de programar. Así también, la importancia de los métodos al instante de trabajar y tener un código que tenga una facilidad de observación al momento de las correcciones del código o en este caso, la visualización de algoritmos de ordenamiento.

METODOLOGÍA DE TRABAJO

Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.

- *La secuencia que utilicé fue, primero, analizar el código base ya proporcionado o que había realizado y tratar de darle parte de la solución en mi mente. Segundo, me dediqué a armar la estructura de los algoritmos faltantes, los distintos métodos. Tercero, analizar errores y formas de optimizar el código para que se visualice de una manera entendible. Cuarto y último, compilar o correr el código para visualizar su funcionamiento o en dado caso no funcione, volver al paso 3 para corregir.*

REFERENCIAS Y BIBLIOGRAFÍA

Ninguna referencia externa.

RÚBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió con el ítem correspondiente.

Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Contenido y demostración		**Puntos**	**Checklist**	**Estudiante**	**Profesor**
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	0	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
TOTAL		20		17	

Tabla 2: Rúbrica para contenido del Informe y demostración