

	UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO (formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>Fundamentos de Programación 2</i>				
TÍTULO DE LA PRÁCTICA:	<i>HashMap</i>				
NÚMERO DE PRÁCTICA:	08	AÑO LECTIVO:	2024-B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	29/11/2024	HORA DE PRESENTACIÓN	18/20/00		
INTEGRANTE (s)	<i>Riveros Vilca Alberth Edwar</i>			NOTA (0-20)	<i>Nota colocada por el docente</i>
DOCENTE(s):	<i>Ing. Lino Jose Pinto Oppe</i>				

RESULTADOS Y PRUEBAS	
I. EJERCICIOS RESUELTOS:	<ol style="list-style-type: none"> 1. Cree un Proyecto llamado <i>Laboratorio8</i> 2. Usted deberá crear las dos clases <i>Soldado.java</i> y <i>VideoJuego5.java</i>. Puede reutilizar lo desarrollado en Laboratorios anteriores. 3. Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero). 4. El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para crear el tablero utilice la estructura de datos más adecuada. 5. Tendrá 2 Ejércitos (usar HashMaps). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: <i>Soldado0X1</i>, <i>Soldado1X1</i>, 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 (distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 2

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 diferentes algoritmos de ordenamiento (indicar conclusiones respecto a este

Marco Aedo López 2

ordenamiento de HashMaps). Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacerlo como programa iterativo.

CLASE SOLDADO:

```
 1 public class Soldado {  
 2     private String nombre;  
 3     private int fila;  
 4     private int columna;  
 5     private int nivelVida;  
 6  
 7     public Soldado(String nombre, int fila, int columna, int nivelVida) {  
 8         this.nombre = nombre;  
 9         this.fila = fila;  
10         this.columna = columna;  
11         this.nivelVida = nivelVida;  
12     }  
13  
14     public void setNombre(String n) {  
15         nombre = n;  
16     }  
17  
18     public void setFila(int f) {  
19         fila = f;  
20     }  
21  
22     public void setColumna(int c) {  
23         columna = c;  
24     }  
25  
26     public void setNivelVida(int p) {  
27         nivelVida = p;  
28     }  
29  
30     public String getNombre() {  
31         return nombre;  
32     }  
33  
34     public int getFila() {  
35         return fila;  
36     }  
37  
38     public int getColumna() {  
39         return columna;  
40     }  
41  
42     public int getNivelVida() {  
43         return nivelVida;  
44     }  
45  
46     @Override  
47     public String toString() {  
48         return "[Nombre: " + this.getNombre() + "\tFila: " + (this.getFila() + 1) + "\tColumna: " + (this.getColumna() + 1)  
49                     + "\tNivel de Vida: " + this.getNivelVida() + "]\\n";  
50     }  
51 }
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 3

CLASE VIDEOJUEGO5:

```
1 import java.util.*;
2
3 public class VideoJuego5 {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6
7         do {
8             Random rand = new Random();
9             HashMap<String, Soldado> ejercito1 = new HashMap<>();
10            HashMap<String, Soldado> ejercito2 = new HashMap<>();
11            String[] nombresEjercito1 = new String[10];
12            String[] nombresEjercito2 = new String[10];
13            HashMap<String, Boolean> casillasOcupadas = new HashMap<>();
14
15            int numSoldados1 = rand.nextInt(10) + 1;
16            int numSoldados2 = rand.nextInt(10) + 1;
17
18            // Crear los ejércitos
19            crearEjercito(rand, numSoldados1, ejercito1, casillasOcupadas, nombresEjercito1, 1);
20            crearEjercito(rand, numSoldados2, ejercito2, casillasOcupadas, nombresEjercito2, 2);
21
22            // Mostrar el estado del tablero
23            showBoard(casillasOcupadas, ejercito1, ejercito2);
24
25            // Mostrar los detalles de los ejércitos
26            System.out.println("Ejército 1:");
27            mostrarDatosEjercito(ejercito1, nombresEjercito1, numSoldados1);
28            System.out.println("Ejército 2:");
29            mostrarDatosEjercito(ejercito2, nombresEjercito2, numSoldados2);
30
31            // Determinar el ganador
32            determinarGanador(ejercito1, ejercito2);
33
34            // Preguntar si se quiere jugar otra partida
35            System.out.print("¿Quieres jugar otra partida? (si/no): ");
36            } while (scanner.nextLine().trim().equalsIgnoreCase("si"));
37
38            System.out.println("Gracias por jugar. ¡Hasta luego!");
39            scanner.close();
40        }
41
42        // Compara la vida total de ambos ejércitos para determinar al ganador
43        // o si el enfrentamiento termina en empate.
44        public static void determinarGanador(HashMap<String, Soldado> ejercito1,
45                                              HashMap<String, Soldado> ejercito2) {
46            int totalVidaEjercito1 = calcularTotalVida(ejercito1);
47            int totalVidaEjercito2 = calcularTotalVida(ejercito2);
48
49            System.out.printf("Total Vida Ejército 1: %d\n", totalVidaEjercito1);
50            System.out.printf("Total Vida Ejército 2: %d\n", totalVidaEjercito2);
51
52            if (totalVidaEjercito1 > totalVidaEjercito2) {
53                System.out.println("El Ejército 1 gana la batalla.");
54            } else if (totalVidaEjercito2 > totalVidaEjercito1) {
55                System.out.println("El Ejército 2 gana la batalla.");
56            } else {
57                System.out.println("La batalla termina en empate.");
58            }
59        }
60    }
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 4

```
61      // Muestra información del ejército, como el soldado con mayor vida,
62      // el promedio de vida, el orden de creación y rankings basados en la vida.
63      public static void mostrarDatosEjercito(HashMap<String, Soldado> ejercito,
64                                              String[] nombresEjercito, int numSoldados) {
65          String mayorVida = findMaxLifeSoldier(ejercito);
66          System.out.println("Soldado de Mayor Vida: " + mayorVida);
67
68          double promedioVida = calcularPromedioVida(ejercito, numSoldados);
69          System.out.println("Promedio de nivel de vida: " + promedioVida);
70
71          System.out.println("Soldados en el orden de creación:");
72          armyCreationOrder(nombresEjercito, numSoldados);
73
74          Soldado[] soldadosLista = toArray(ejercito);
75          bubbleSortLife(soldadosLista);
76          System.out.println("Ranking de poder (Bubble Sort):");
77          showArmyInfo(soldadosLista);
78
79          soldadosLista = toArray(ejercito);
80          insertionSortLife(soldadosLista);
81          System.out.println("Ranking de poder (Insertion Sort):");
82          showArmyInfo(soldadosLista);
83      }
84
85      // Genera un ejército de soldados en posiciones aleatorias del tablero,
86      // asegurándose de que las casillas no se repitan.
87      public static void crearEjercito(Random rand, int numSoldados, HashMap<String, Soldado> ejercito,
88                                       HashMap<String, Boolean> casillasOcupadas, String[] nombresEjercito,
89                                       int ejercitoNum) {
90          int count = 0;
91
92          while (count < numSoldados) {
93              int randColumn = rand.nextInt(10);
94              int randRow = rand.nextInt(10);
95
96              String key = randRow + "," + randColumn;
97
98              if (!casillasOcupadas.containsKey(key) || !casillasOcupadas.get(key)) {
99                  casillasOcupadas.put(key, true);
100
101                  String nombreSoldado = "Soldado" + count + "X" + ejercitoNum;
102                  nombresEjercito[count] = nombreSoldado;
103
104                  ejercito.put(key, new Soldado(nombreSoldado, randRow, randColumn,
105                                              rand.nextInt(5) + 1));
106                  count++;
107              }
108          }
109      }
110
111 }
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 5

```
111 // Imprime el estado actual del tablero, mostrando las posiciones ocupadas
112 // por los soldados de ambos ejércitos y sus niveles de vida.
113 public static void showBoard(HashMap<String, Boolean> casillasOcupadas, HashMap<String,
114     Soldado> ejercito1, HashMap<String, Soldado> ejercito2) {
115     System.out.print("\t");
116     for (char i = 'A'; i < 'K'; i++) {
117         System.out.print("    " + i + "    ");
118     }
119     System.out.println();
120     System.out.print("    ");
121     for (int l = 0; l < 10; l++) {
122         System.out.print("-----");
123     }
124     System.out.println();
125     for (int j = 0; j < 10; j++) {
126         System.out.print((j + 1) + (j < 9 ? "   :   "));
127         for (int k = 0; k < 10; k++) {
128             System.out.print("|");
129             String key = j + "," + k;
130             if (casillasOcupadas.getOrDefault(key, false)) {
131                 if (ejercito1.containsKey(key)) {
132                     System.out.printf(" +%-2d ", ejercito1.get(key).getNivelVida());
133                 } else if (ejercito2.containsKey(key)) {
134                     System.out.printf(" *%-2d ", ejercito2.get(key).getNivelVida());
135                 }
136             } else {
137                 System.out.print("        ");
138             }
139         }
140         System.out.println("|");
141         System.out.print("    ");
142         for (int l = 0; l < 10; l++) {
143             System.out.print("-----");
144         }
145         System.out.println();
146     }
147 }
148
149 // Imprime los nombres de los soldados en el orden en que fueron creados.
150 public static void armyCreationOrder(String[] nombresEjercito, int numSoldados) {
151     for (int i = 0; i < numSoldados; i++) {
152         System.out.println(nombresEjercito[i]);
153     }
154     System.out.println();
155 }
156
157 // Convierte un HashMap de soldados a un arreglo para facilitar el procesamiento.
158 public static Soldado[] toArray(HashMap<String, Soldado> ejercito) {
159     return ejercito.values().toArray(new Soldado[0]);
160 }
161
```



```
162     // Imprime información de cada soldado en el ejército, como su posición y nivel de vida.
163     public static void showArmyInfo(Soldado[] ejercito) {
164         for (Soldado soldado : ejercito) {
165             System.out.print(soldado);
166         }
167     }
168
169     // Calcula la suma total de los niveles de vida de todos los soldados del ejército.
170     public static int calcularTotalVida(HashMap<String, Soldado> ejercito) {
171         int totalVida = 0;
172         for (Soldado soldado : ejercito.values()) {
173             totalVida += soldado.getNivelVida();
174         }
175         return totalVida;
176     }
177
178     // Calcula el promedio del nivel de vida de los soldados en el ejército.
179     public static double calcularPromedioVida(HashMap<String, Soldado> ejercito,
180                                                 int numSoldados) {
181         int totalVida = calcularTotalVida(ejercito);
182         return (double) totalVida / numSoldados;
183     }
184
185     // Encuentra al soldado con el nivel de vida más alto en el ejército.
186     public static String findMaxLifeSoldier(HashMap<String, Soldado> ejercito) {
187         Soldado max = null;
188         for (Soldado soldado : ejercito.values()) {
189             if (max == null || soldado.getNivelVida() > max.getNivelVida()) {
190                 max = soldado;
191             }
192         }
193         return max != null ? max.toString() : "No hay soldados.";
194     }
195
196     // Ordena el arreglo de soldados por nivel de vida en orden ascendente
197     // utilizando el algoritmo de inserción (Insertion Sort).
198     public static void insertionSortLife(Soldado[] ejercito) {
199         for (int i = 1; i < ejercito.length; i++) {
200             Soldado key = ejercito[i];
201             int j = i - 1;
202             while (j ≥ 0 && ejercito[j].getNivelVida() > key.getNivelVida()) {
203                 ejercito[j + 1] = ejercito[j];
204                 j--;
205             }
206             ejercito[j + 1] = key;
207         }
208     }
209 }
```



```
209
210     // Ordena el arreglo de soldados por nivel de vida en orden descendente
211     // utilizando el algoritmo de burbuja (Bubble Sort).
212     public static void bubbleSortLife(Soldado[] ejercito) {
213         for (int i = 0; i < ejercito.length - 1; i++) {
214             for (int j = 0; j < ejercito.length - i - 1; j++) {
215                 if (ejercito[j].getNivelVida() < ejercito[j + 1].getNivelVida()) {
216                     Soldado temp = ejercito[j];
217                     ejercito[j] = ejercito[j + 1];
218                     ejercito[j + 1] = temp;
219                 }
220             }
221         }
222     }
223 }
```

codesnap.dev

II. PRUEBAS

¿Con qué valores comprobaste que tu práctica estuviera correcta?

Utilicé valores generados aleatoriamente por el programa para el número de soldados (entre 1 y 10) y sus posiciones en el tablero. Además, verifiqué que los niveles de vida de los soldados estuvieran dentro del rango esperado (1 a 5).

¿Qué resultado esperabas obtener para cada valor de entrada?

Los soldados se colocaron en casillas únicas del tablero sin superposición, el tablero mostrara correctamente las posiciones de los soldados con sus niveles de vida, los datos de los ejércitos se calcularan correctamente, el programa determinara al ejército ganador o un empate según las vidas totales de los ejércitos.

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

Los soldados se colocaron en casillas únicas, el tablero mostró correctamente los símbolos de cada ejército (+ para el ejército 1 y * para el ejército 2) con los niveles de vida correspondientes y el ganador se determinó correctamente según las vidas totales.



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 8

EJECUCIÓN:

	A	B	C	D	E	F	G	H	I	J
1										*:3
2										
3										
4										
5						*:5				
6						+:2				
7										
8										
9										
10						*:1				

codesnap.dev



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 9



Ejército 1:

Soldado de Mayor Vida: [Nombre: Soldado0X1 Fila: 6 Columna: 5 Nivel de Vida: 2]

Promedio de nivel de vida: 2.0

Soldados en el orden de creación:

Soldado0X1

Ranking de poder (Bubble Sort):

[Nombre: Soldado0X1 Fila: 6 Columna: 5 Nivel de Vida: 2]

Ranking de poder (Insertion Sort):

[Nombre: Soldado0X1 Fila: 6 Columna: 5 Nivel de Vida: 2]

Ejército 2:

Soldado de Mayor Vida: [Nombre: Soldado2X2 Fila: 5 Columna: 5 Nivel de Vida: 5]

Promedio de nivel de vida: 3.0

Soldados en el orden de creación:

Soldado0X2

Soldado1X2

Soldado2X2

Ranking de poder (Bubble Sort):

[Nombre: Soldado2X2 Fila: 5 Columna: 5 Nivel de Vida: 5]

[Nombre: Soldado1X2 Fila: 1 Columna: 8 Nivel de Vida: 3]

[Nombre: Soldado0X2 Fila: 10 Columna: 4 Nivel de Vida: 1]

Ranking de poder (Insertion Sort):

[Nombre: Soldado0X2 Fila: 10 Columna: 4 Nivel de Vida: 1]

[Nombre: Soldado1X2 Fila: 1 Columna: 8 Nivel de Vida: 3]

[Nombre: Soldado2X2 Fila: 5 Columna: 5 Nivel de Vida: 5]

Total Vida Ejército 1: 2

Total Vida Ejército 2: 9

El Ejército 2 gana la batalla.

¿Quieres jugar otra partida? (si/no): no

Gracias por jugar. ¡Hasta luego!

Process finished with exit code 0

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 10

III. CUESTIONARIO:

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

CAPTURAS DE LOS COMMIT:

```
rivxd@pop-os:~/Imágenes/U/Laboratorios FP2-Lino
~/U/Laboratorios FP2-Lino ) on 🐒 P main +1 git add .
~/U/Laboratorios FP2-Lino ) on 🐒 P main +1 git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:   RIVEROS_VILCA_LABORATORIO_08/VideoJuego5.java

~/U/Laboratorios FP2-Lino ) on 🐒 P main +1 git commit -m "lab08-finalizado"
[main 81d9efa] lab08-finalizado
 1 file changed, 224 insertions(+), 209 deletions(-)
 rewrite RIVEROS_VILCA_LABORATORIO_08/VideoJuego5.java (60%)

~/U/Laboratorios FP2-Lino ) on 🐒 P main +1 git push -u origin main
Username for 'https://github.com': rivX241
Password for 'https://rivX241@github.com':
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Comprimiendo objetos: 100% (4/4), listo.
Escribiendo objetos: 100% (4/4), 1.93 KiB | 1.93 MiB/s, listo.
Total 4 (delta 2), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/rivX241/RIVEROS_VILCA_LABORATORIOS.git
  707fe1e..81d9efa  main -> main
Rama 'main' configurada para hacer seguimiento a la rama remota 'main' de 'origin'.
```

The screenshot shows the GitHub interface displaying a list of commits. The commits are organized by date:

- Commits on Nov 29, 2024:
 - lab08-finalizado (rivX241 committed 5 minutes ago)
- Commits on Nov 26, 2024:
 - lab08-avance (rivX241 committed 3 days ago)
- Commits on Nov 15, 2024:
 - Add files via upload (rivX241 authored 2 weeks ago)
 - lab07-finalizado (rivX241 committed 2 weeks ago)
- Commits on Oct 25, 2024:
 - lab06-finalizado (rivX241 committed on Oct 25)
 - lab06-finalizado (rivX241 committed on Oct 25)
- Commits on Oct 22, 2024:
 - avance_lab06_gradle (rivX241 committed on Oct 22)
- Commits on Oct 18, 2024:

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 11

CAMBIOS REALIZADOS:

```
diff --git a/RIVEROS_VILCA_LABORATORIO_08/VideoJuego5.java b/RIVEROS_VILCA_LABORATORIO_08/VideoJuego5.java
--- a/RIVEROS_VILCA_LABORATORIO_08/VideoJuego5.java
+++ b/RIVEROS_VILCA_LABORATORIO_08/VideoJuego5.java
@@ -1,34 +1,48 @@
1  import java.util.*;
2 +
3  public class VideoJuego5 {
4      public static void main(String[] args) {
5          Random rand = new Random();
6          HashMap<Integer, Soldado> ejercito1 = new HashMap<>();
7          HashMap<Integer, Soldado> ejercito2 = new HashMap<>();
8          ArrayList<String> nombresEjercito1 = new ArrayList<>();
9          ArrayList<String> nombresEjercito2 = new ArrayList<>();
10         HashMap<String, Boolean> casillasOcupadas = new HashMap<>();
11
12         int numSoldados1 = rand.nextInt(10) + 1;
13         int numSoldados2 = rand.nextInt(10) + 1;
14
15         // Crear los ejércitos
16         crearEjercito(rand, numSoldados1, ejercito1, casillasOcupadas, nombresEjercito1, 1);
17         crearEjercito(rand, numSoldados2, ejercito2, casillasOcupadas, nombresEjercito2, 2);
18
19         // Mostrar el estado del tablero
20         showBoard(casillasOcupadas, ejercito1, ejercito2);
21
22         // Mostrar los detalles de los ejércitos
23         System.out.println("Ejército 1:");
24         mostrarDatosEjercito(ejercito1, nombresEjercito1, numSoldados1);
25         System.out.println("Ejército 2:");
26         mostrarDatosEjercito(ejercito2, nombresEjercito2, numSoldados2);
```

Se terminó de completar las implementaciones de los métodos al uso de HashMap.

Cambie la rama a main, añadí el informe al área de stage y hice un commit lab05-finalizado y realice el git push -u origin main el origin ya estaba previamente configurado para todos los laboratorios.

LINK:https://github.com/rivX241/RIVEROS_VILCA_LABORATORIOS

CONCLUSIONES

Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.

Trabajar con arreglos bidimensionales y estructuras como HashMap en Java me permitió comprender su utilidad para organizar datos de forma estructurada, similar a un tablero donde cada elemento tiene una posición definida. Aunque enfrenté desafíos al manejar datos no contiguos, esto me ayudó a desarrollar mejores habilidades para recorrer y manipular estructuras más complejas. Este ejercicio me mostró la importancia de planificar cuidadosamente la lógica al implementar algoritmos en contextos con datos



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 12

distribuidos.

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.

Análisis de requisitos: Identifiqué las necesidades del problema, como gestionar soldados en un tablero bidimensional y calcular métricas como vida total y promedio.

Diseño del código: Definí clases y métodos para crear ejércitos, asignar posiciones únicas en el tablero y representar visualmente los soldados (+ :nivelDeVida y * :nivelDeVida).

Implementación: Programé la lógica para crear ejércitos, calcular estadísticas y ordenar soldados por nivel de vida con algoritmos de burbuja e inserción.

Pruebas y ajustes: Verifiqué que no hubiera posiciones repetidas, que los cálculos fueran correctos y que el tablero se visualizara claramente. Optimicé la representación y corregí errores detectados durante las pruebas.

REFERENCIAS Y BIBLIOGRAFÍA

E. G. Castro Gutiérrez y M. W. Aedo López, *Fundamentos de programación 2: tópicos de programación orientada a objetos*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021. ISBN: 978-612-5035-20-2. 170 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 20.5 x 29 cm.



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 13

RUBRICA:

Contenido y demostración		Puntos	Checklis t	Estudiant e	Profeso r
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	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
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	1	
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	3	
TOTAL		20		17	