Pregunta 1 — 0 pts

Yo, en mi calidad de estudiante de la ESPOL, declaro que he sido informado y conozco las normas que rigen a la ESPOL, en particular el Código de Ética y el Reglamento de Disciplina.

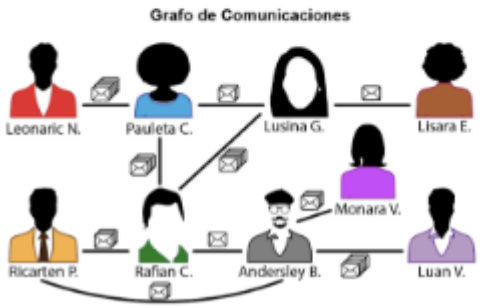
Acepto con pleno conocimiento este compromiso de honor por el cual reconozco y estoy consciente de que la presente evaluación está diseñada para ser resuelta de forma individual; que sólo puedo comunicarme con la persona responsable de la recepción de la evaluación; que no haré consultas indebidas o no autorizadas por el evaluador; ni usaré dispositivos electrónicos.

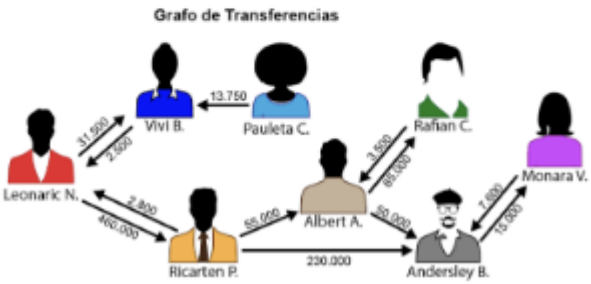
Acepto el presente compromiso, como constancia de haber leído y aceptar la declaración anterior y me comprometo a seguir fielmente las instrucciones que se indican para la realización de la presente evaluación.

- ○ Verdadero
- ○ Falso

Pregunta 2 — 40 pts

Considere los siguientes grafos, que representan una red de corrupción investigada por la Fiscalía:





En ambos grafos, los nodos representan personas. Una misma persona puede aparecer en ambos grafos, pero hay personas que solo aparecen en uno.

El primer grafo representa las **comunicaciones** entre los miembros de la red. Allí, cada arco almacena una lista de mensajes como metadata. Un mensaje tiene un remitente, un destinatario, una fecha, un asunto, y un contenido.

El segundo grafo representa **transferencias** que se han realizado entre los individuos de la red. El monto total transferido se almacena en el peso del arco correspondiente, cuya dirección indica, además, el rol de los involucrados. Es decir, el vértice de origen envía el dinero y el vértice de destino lo recibe. La metadata almacenada en los arcos de este grafo es siempre igual al peso del arco.

Implemente el método **calcularActividad**, que cuantifica la actividad de cada individuo considerando tanto sus comunicaciones como sus movimientos financieros. Para cada individuo que aparezca en ambas redes, el método debe crear un objeto de tipo **Perfil**, de acuerdo con la siguiente definición:

```
public class Perfil {

    private Persona sospechoso;
    private List<Mensaje> mensajesEnviados;
    private List<Mensaje> mensajesRecibidos;
    private int dineroEnviado;
    private int dineroRecibido;

    // constructores, getters, y setters

}
```

El método debe retornar un mapa cuyas claves son los nombres de los sospechosos, cada uno de los cuales estará asociado al perfil calculado.

Adicionalmente, antes de retornar el resultado correspondiente, el método **calcularActividad** debe imprimir un reporte de acuerdo al siguiente formato:

Nombre de la persona, saldo neto de transferencias, conteo de mensajes

El saldo neto de transferencias es la suma de todas las transferencias salientes menos la suma de las entrantes.

El conteo de mensajes, es la suma de los mensajes enviados y recibidos por cada individuo.

Al implementar sus métodos, asuma que usted cuenta con una implementación funcional de la clase Grafo, implementada mediante listas de adyacencia. Esta clase, incluye métodos para:

Agregar, eliminar, y buscar nodos y arcos

Obtener vecinos y grado de un nodo

Obtener lista de todos los nodos y todos los arcos

Recorrer el grafo por anchura y profundidad

Algoritmos de camino más corto: Dijkstra, Floyd, y Floyd-Warshall

Algoritmos de árbol de costo mínimo: Warhsall, y Kruskal.

Usted es responsable de definir las entradas y salidas de cada uno de los métodos listados arriba. Utilice comentarios en su código para especificar claramente las entradas y salidas de los métodos de la clase grafo; esto será evaluado como parte de su calificación.

Pregunta 3 — 30 pts

IMPORTANTE (Condiciones obligatorias):

- Resolver **exclusivamente con pilas** (`java.util.Stack`).
- **No** se permite usar ninguna otra colección (p. ej., `ArrayList`, `LinkedList`, `Queue`, `Deque`, etc.).
- Si el método se invoca sin especificar `k`, se debe asumir por defecto "`k" = 1`.
- El valor de "**k**" debe ser mayor que cero, si se ingresa un valor negativo o igual a cero, deberá lanzar una excepción mostrando el mensaje: "El valor de k debe ser positivo y mayor que cero."

Implemente un método que elimine los primeros `"` `k"` elementos de un arreglo de enteros que:

- Sean menores que el siguiente elemento
- Se conviertan en menores que el siguiente como resultado de eliminaciones previas.

El método debe retornar una **pila** con los elementos restantes del arreglo original.

Ejemplo:

Entrada: `arr = [20, 10, 25, 30, 40]`, `"k = 2"`

Salida: `[25, 30, 40]`

Explicación: primero se elimina `10` porque `10 < 25`. Tras esa eliminación, `20` queda antes de `25` y también se elimina porque `20 < 25`. No se realizan más eliminaciones.

Prototipos requeridos:

```
// Versión con k explícito
public static Stack<Integer> eliminarPrimerosK(int[] array, int k);

// Versión con k por defecto = 1
public static Stack<Integer> eliminarPrimerosK(int[] array) {
    return eliminarPrimerosK(array, 1);
}
```

Pregunta 4 — 30 pts

La biblioteca de la universidad tiene un sistema básico de gestión de libros, los libros están almacenados en una estructura tipo Mapa personalizado (**MyMap<K, V>**), donde la **Clave (K) (String)** almacena el Código único del libro y el **Valor (V) es un Objeto** de tipo Libro, que contiene, titulo (String), Autor (String) y cantidad (int). El sistema cuenta con la clase Libro, la clase MyMap<K, V>, y con los métodos básicos como put(), get(), remove().

A) Conocer el libro que tiene más copias (10 Puntos)

Debido a que está cerca una feria de libros y hay que preparar los libros para la exhibición, se necesita conocer cuál es el libro con la mayor cantidad disponible en stock, dado que no se cuenta con un método que realice esta búsqueda de forma automática, se requiere que implemente en la clase MyMap<K, V> el método:

```
public V obtenerLibroMayorStock(),
```

el cual debe recorrer todo el mapa y retornar el **objeto Libro** que tenga la cantidad (cantidad (int)) más alta y en el caso de que el mapa esté vacío, el método debe retornar **null**.

Nota: En caso de que exista un empate entre dos o más libros con la misma cantidad máxima, el método debe retornar **el primer libro encontrado** durante el recorrido

Ejemplo de la ejecución del método:

```
MyMap<String, Libro> miMapa = new MyMap<>();
// Agregar libros
miMapa.put("L001", new Libro("Estructuras de Datos en Java", "Ing. Freire", 4));
miMapa.put("L002", new Libro("Algoritmos y Complejidad", "Ing. Cevallos", 2));
miMapa.put("L003", new Libro("Programación Orientada a Objetos", "Ing. Martínez", 6));
miMapa.put("L004", new Libro("Bases de Datos Avanzadas", "Ing. Ramírez", 5));
miMapa.put("L005", new Libro("Fundamentos de Programación", "Ing. Perez", 6));

// -------- Literal A --------
Libro libroMayor = miMapa.obtenerLibroMayorStock();
if (libroMayor != null) {
        System.out.println("Libro con mayor stock: " + libroMayor);
} else {
        System.out.println("Mapa vacío");
}
```

Salida esperada:

Libro con mayor stock es: Programación Orientada a Objetos - Ing. Martínez - Cantidad: 6

Restricciones:

No se permite usar HashMap, TreeMap, LinkedHashMap ni otras colecciones predefinidas de Java.

B) Conocer los libros que tiene más copias – cuando hay un Empate (10 Puntos)

Dado que en algún momento puede coincidir que, dos libros tengan la misma cantidad de libros y esta cantidad sea la más alta registrada en el sistema, cree el método:

```
public MyList<Libro> obtenerLibrosMayorStock(),
```

para que devuelva una lista personalizada (MyList<Libro>) con todos los libros que tengan la cantidad máxima y en caso de que el mapa esté vacío, el método devolverá una lista vacía.

\\-------- Literal B --------

```
MyList<Libro> listaEmpates = miMapa.obtenerLibrosMayorStock();
System.out.println("Libros empatados en mayor stock:");
for (int i = 0; i < listaEmpates.size(); i++) {
        System.out.println(listaEmpates.get(i));
}
```

Salida esperada:

Programación Orientada a Objetos - Ing. Martínez - Cantidad: 6
Fundamentos de Programación - Ing. Perez - Cantidad: 6

C) Cantidad total de libros que tiene la biblioteca (10 Puntos)

Dentro del desarrollo, ahora se tiene la necesidad de conocer la cantidad total de libros que se tiene en la biblioteca, implemente dentro de la clase MyMap<K, V> el método:

```
public int obtenerTotalLibros();
```

el cual recorre todo el mapa y cuenta la cantidad de cada libro y permite así conocer la cantidad total de libros que se tiene en biblioteca.

```
// -------- Literal C --------
int totalLibros = miMapa.obtenerTotalLibros();
System.out.println("Cantidad total de libros en biblioteca: " + totalLibros);
```

Salida esperada:

Cantidad total de libros en biblioteca: 23