Primer Parcial de Estructuras de Datos y Algoritmos (EDA) – 28 de Marzo de 2022 - Duración: 1h. 30m.

APELLIDOS, NOMBRE	GRUPO

1.- ***(2,25 puntos)*** Implementa un método estático tal que:

- Reciba una **`ListaConPI`** genérica, cuyos elementos están ordenados ascendentemente, y que puede contener valores repetidos.
- Borre de la lista recibida todos los valores repetidos.
- Devuelva el número de elementos borrados.

La solución debe ser eficiente (coste lineal con la talla de la lista), y no se debe usar ninguna EDA auxiliar.

Ejemplos.

- Si se recibe la lista [1,1,2,2,2,3,3,4,4,5,7,7,7,8], el método devolverá 7 y el contenido de la lista cambiará a [1,2,3,4,5,7,8].
- Si se recibe la lista [“aa”,”aa”,”ab”,”ba”,”bb“,“bb”,”bc”,”cc”,”cc”,”cc”], el método devolverá 4 y el contenido de la lista cambiará a [“aa”,”ab”,”ba”,”bb“,”bc”,”cc”].

2.- ***(2,50 puntos)*****.** Aplicación de la estrategia **Divide y Vencerás (DyV)**.

Sea **v**, un array genérico de elementos **`Comparable`**, ordenado ascendentemente y que puede contener valores repetidos, y sea **e**, un elemento de la misma clase que los elementos de **v**. Se quiere obtener el número de apariciones de **e** en **v**.

Aplicando DyV, se pide implementar dos métodos (uno lanzadera y otro recursivo) que devuelvan el número de apariciones de **e** en **v**.

El problema tiene instancias significativas. Con una correcta aplicación de DyV, el caso mejor, y el caso promedio, tendrán coste logarítmico. Aunque existe un caso peor de coste lineal (cuando el elemento buscado **e** está repetido en todas las posiciones del array **v**).

Algunos ejemplos en la siguiente tabla:

v	**e**	*resultado*
[1,1,2,2,2,3,3,4,4,5,7,7,7,8,9,9,9]	3	2
[1,1,2,2,2,3,3,4,4,5,7,7,7,8,9,9,9]	6	0
[1,1,2,2,2,3,3,4,4,5,7,7,7,8,9,9,9]	7	3
[1,1,2,2,2,3,3,4,4,5,7,7,7,8,9,9,9]	8	1
[“aa”,”aa”,”ab”,”ba”,”bb“,“bb”,”bc”,”cc”,”cc”,”cc”,”da”,”da”]	“aa”	2
[“aa”,”aa”,”ab”,”ba”,”bb“,“bb”,”bc”,”cc”,”cc”,”cc”,”da”,”da”]	“cc”	3

3.- ***(2,75 puntos)*** Durante unos disturbios, la policía ha monitorizado todas las comunicaciones en la zona y dispone de una **`ListaConPI`** de **`String`** (conexiones, identificadas mediante números de móvil), con todas las conexiones realizadas. La policía estima que aquellos números de móvil que hayan hecho más de **`umbral`** conexiones podrían corresponder a los cabecillas de los disturbios. La policía también estima que el número de cabecillas es, como máximo, un 10% del número total de conexiones.

Para identificar a los cabecillas, se quiere implementar un método estático que reciba la **`ListaConPI`** de **`String`**, con todas las conexiones, y el **`umbral`** (un entero), y que, eficientemente (en el menor tiempo posible), devuelva otra **`ListaConPI`** de **`String`**, con todos los números de móvil que hayan realizado al menos el número de conexiones establecidas por dicho **`umbral`**. Los titulares de esos móviles serán, probablemente, los cabecillas de los disturbios.

4.- ***(2,50 puntos)*** Implementa en la clase **`MonticuloBinario`**, un método que elimine del montículo aquellos elementos para los que un método auxiliar **`valido`** devuelve **`false`**. Dicho método, que se supone implementado, tiene la siguiente cabecera:

```
private boolean valido(E e);
```

La clase **`MonticuloBinario`** representa un montículo minimal. Por tanto, el objeto modificado al ejecutar el método pedido deberá seguir siendo un montículo minimal (es decir, se deben preservar sus propiedades estructural y de orden).

Se debe implementar con el menor coste y sin usar ninguna EDA auxiliar.

ANEXO.

Interfaces **ListaConPI, ColaPrioridad** y **Map** del paquete **modelos**. Clase **MonticuloBinario** del paquete **jerarquicos**.

```
public interface ListaConPI<E> {
  void insertar(E e);
  void eliminar();
  void inicio();
  void siguiente();
  void fin();
  E recuperar();
  boolean esFin();
  boolean esVacia();
  int talla();
}


public interface ColaPrioridad<E extends
Comparable<E>> {
  void insertar(E e);
  E eliminarMin();
  E recuperarMin();
  boolean esVacia();
}
```

```
public interface Map<C, V> {
  V insertar(C c, V v);
  V eliminar(C c);
  V recuperar(C c);
  boolean esVacio();
  int talla();
  ListaConPI<C> claves();
}


public class MonticuloBinario<E extends
Comparable<E>> implements ColaPrioridad<E>
{
  protected E[] elArray;
  protected int talla;
  ...
  protected void hundir(int pos) { ... }
  ...
}
```