Normas del examen

- En todos los ejercicios se valorará la eficiencia de las estructuras de datos utilizadas y del código implementado, así como la concisión y claridad del código.
- Si un código resuelve el problema de forma no eficiente se podrá considerar como no válido, aunque la salida sea correcta.
- Para comenzar a resolver el examen, descarga el fichero con el código de apoyo del Aula Virtual, descomprime ese fichero y copia el contenido de la carpeta *src* dentro de la carpeta *src* de tu proyecto.
- La entrega se basará en un único fichero ZIP que contendrá la carpeta *src* con el código de todo tu proyecto.
- Puedes implementar tantas clases y métodos adicionales sean necesarios, pero sin modificar las cabeceras de los métodos propuestos.
- No se permite la utilización de ningún tipo de apoyo, ni prácticas realizadas previamente. Cualquier intento de utilización de este tipo de herramientas implicará el suspenso automático de la convocatoria y las medidas disciplinarias oportunas.

Parte teórica

Ejercicio 1 [1 punto]

En el sorteo de navidad se quiere disponer de una aplicación que permita que cada persona pueda comprobar desde el móvil si su número ha sido premiado o no, y se quiere hacer mediante una tabla hash. Se pide simular como quedaría una tabla hash con 10 posiciones que usa direccionamiento abierto con la siguiente función hash:

$$H(k) = k \bmod 17$$
$$D(k) = 7 - (k \bmod 7)$$

Si realizan las siguientes operaciones en dicho en un mapa Map<String, String>:

```
mapa.put("75206","6k");
mapa.put("66212","6k");
mapa.put("41710","20k");
mapa.put("10989","125k");
mapa.put("14823","50k");
```

mapa.put("26590","400k");
mapa.put("49797", "20K");
mapa.remove("14823");
mapa.put("00750", "50K");

Para simplificar los cálculos solo se útilizarán las dos ultimas cifras de la clave. Por ejemplo, para la clave "00750" realizarán los cálculos con k=50. Las posiciones sin usar se indicarán mediante un guión "-" y no será necesario hacer rehash en ningún momento.

0	1	3	4	5
(41710, 20k)]	AVAILABLE	(66212, 6k)	49797,20K)	(10989, 125k)

5	6	7	8	9
(26590, 400k)	(75206, 6k)	-	(00750, 50k)	-

Ejercicio 2 [1 punto]

Se tiene un árbol AVL en el que se realizan las operaciones:
Inserciones: 10, 14, 20, 3, 1, 12, 4, 11, 25, 22, 28
Eliminaciones: 11
Se debe escribir el recorrido preorden del árbol resultante.

1	2	3	4	5	6	7	8	9	10
14	10	3	1	4	12	22	20	25	28

Parte práctica

Ejercicio 1 [2 puntos]

Implementa el método *removeFrontier* definido en la clase *RemoveFrontier*. Dado un árbol n-ario, este método elimina del mismo todos los nodos que sean hojas del árbol.

Ejercicio 2 [2.5 puntos]

Implementa el método *findHalf* definido en la clase *Half*. Dado un árbol binario, este método devuelve una colección iterable con todos los nodos que están a una altura $h/2$ del árbol, siendo h la altura máxima del árbol.

Ejercicio 3 [3.5 puntos]

En el programa de La Resistencia, David Broncano siempre hace las mismas preguntas a los invitados: cuánto dinero cobra y … otra pregunta sobre actividades físicas. En este examen nos interesa la primera de ellas, por lo que ignoraremos la segunda. Tras tres temporadas, la realización del programa está empezando a tener problemas de eficiencia para saber cuánto cobraba cada uno de los famosos y en qué mes visitó el programa, para poder sacar estadísticas.

Por ello, nos han encargado que implementemos un programa que almacene, de forma eficiente, los famosos que visitan el programa y el dinero que tienen en el momento de la visita. Hay que tener en cuenta que algunos famosos visitan varias veces el programa, y debemos guardar todas las visitas.

Se pide:

a) **[0.5 puntos]** Definir la estructura(s) de datos necesaria(s) para almacenar las visitas que hacen los famosos al programa, así como cuánto cobraban y en qué mes hicieron la visita.
b) **[0.5 puntos]** Implementar el método addVisit, cuya cabecera está definida en la clase LaResistencia, que simula la visita de un famoso y la almacena en la(s) estructura(s) de datos definida(s).
c) **[1 punto]** Implementar el método overMedian, cuya cabecera está definida en la clase LaResistencia, que devuelve una colección iterable con el nombre de todos los famosos que ganan un valor **estrictamente** superior a la mediana de todos los invitados. Por ejemplo, dados los datos 1, 3, 3, 6, 7, 8, 9, la mediana sería 6, ya que ocupa la posición central. Si un famoso ha visitado varias veces el programa solo se tendrá en cuenta lo que respondió la última vez.
d) **[1 punto]** Implementa los métodos moneyInMonth y visitsInMonth, cuyas cabeceras estarán definidas en la clase LaResistencia. El primero recibe el nombre del mes y devuelve la suma de dinero que tienen los invitados recibidos en dicho mes. El segundo recibe el nombre del mes y devuelve

todas las visitas que se produjeron en dicho mes. Se asume que ningún famoso visita el programa dos veces en el mismo mes.

e) **[0.5 puntos]** Implementa un programa principal que añada los siguientes invitados y ejecute los métodos implementados:

Nombre	Dinero	Mes
M. Rajoy	12000	Marzo
Antonia Larroca	500000	Junio
Jorge Niño-Grande	20000	Agosto
Gerardo Picado	1000000	Septiembre
Santi Caballo	75000	Diciembre
Horse Luis	1000000	Septiembre