Apellidos: ***Nombre:***
Grupo: ***DNI:***
Duración: 3:00 h. (incluyendo tests)

Ejercicio 1	Ejercicio 2	Ejercicio 3	TOTAL

1.1: **[1.5 puntos]** Dada la función Hash H(*k*) = (DNI mod 100) mod (TAM_TABLA), y la siguiente secuencia de inserciones

Seq.	Nombre	Apellido	DNI
1	Mariano	Rajoy	41.002.103
2	Alfredo	P. Rubalcaba	8.424.220
3	Cayo	Lara	85.669.039
4	Josep A.	Durán	35.875.692
5	Rosa	Díez	41.002.119

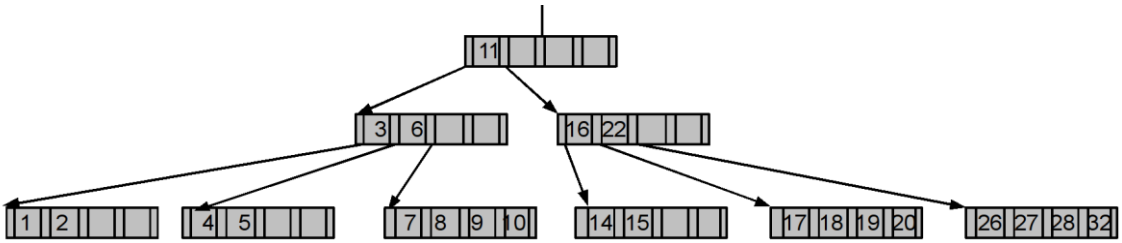
Represente gráficamente cómo sería la tabla hash asociada si las colisiones se resuelven por encadenamiento separado o direccionamiento cerrado (prueba lineal).

1.2: **[1.5 puntos]** La base de datos IMDB_URJC dispone de un fichero de datos con nombres de actores y sueldo que cobran (en promedio) por cada película en la que participa. El fichero tiene un aspecto como el siguiente:

	Nombre	Apellido	Sueldo (M $)
1	Mickey	Rourke	2
2	Jason	Statham	5
3	Jet	Li	10
4	Dolph	Lundgren	13
5	Chuck	Norris	1
6	Jean-Claude	Van Damme	12
7	Bruce	Willis	16
8	Sylvester	Stallone	14
9	Arnold	Schwarzenegger	15

Para poder hacer consultas eficientes, esta empresa quiere almacenar esta información en diccionarios ordenados. Como no sabe cuál va a ser más eficiente, quiere que los alumnos de EDAs implementen un diccionario ordenado basado en un árbol binario de búsqueda, otro basado en un AVL y otro basado en un Rojo-Negro. ¿Cuál de ellos considera que es el más adecuado? Justifique su respuesta.

1.3: **[1.0 punto]** Dado el siguiente árbol B de orden 5 que aparece en la figura, representar cómo sería el fichero que lo almacena. Se pide además representar el árbol después de la **inserción** de la clave 33 (árbol resultante con todos los pasos intermedios) y la **eliminación** de la clave 2 (árbol resultante con todos los pasos intermedios).



Ejercicio 2 [3 puntos].
Se desea crear una estructura de datos, llamada `ConjuntoDifuso`, para almacenar números de tipo `double`. La estructura de datos recibirá en el constructor un *array* con los números a insertar. Obsérvese que la estructura de datos será inmutable, ya que no se le pueden añadir más datos una vez construida. Además del constructor, la estructura de datos solo dispondrá del siguiente método:

```
public double busquedaAproximada(double valorABuscar)
```

El método `busquedaAproximada` lanzará una excepción si se invoca sobre un `ConjuntoDifuso` vacío. En caso de que no esté vacío devolverá el valor más "parecido" al que se le pide buscar de entre los que hayan sido previamente insertados con el constructor. El criterio de parecido se basará en la diferencia en valor absoluto de los números. Por ejemplo, 6 es más parecido a 5.7 que a 7 porque la diferencia en valor absoluto de 6 y 5.7 es 0.3, mientras que la de 6 y 7 es 1.

a) Implementar en Java la estructura de datos `ConjuntoDifuso` de manera que las búsquedas sean eficientes. Para hacer más simple el ejercicio, puede suponerse que el usuario de `ConjuntoDifuso` ordenará los datos a insertar en la secuencia que más le convenga al método de inserción. Obsérvese que si los datos se insertan en algún orden específico el algoritmo de inserción puede simplificarse mientras que las búsquedas seguirían siendo óptimas. (**2 puntos**)
b) Explicar con palabras la mejora que se debería implementar para que no fuese necesario disponer los valores a insertar en un orden determinado (**0.5 puntos**)
c) Explicar con palabras qué mejora se podría realizar para que el criterio de comparación utilizado al buscar pudiese cambiarse a voluntad del usuario de `ConjuntoDifuso` (**0.5 puntos**)

Nota.- El ejercicio puede realizarse sin utilizar ninguna otra clase de Java, pero si se desea puede utilizarse cualquiera de las clases vistas durante el curso.

Ejercicio 3 [3 puntos].
Se pide implementar el método `parejas` en Java:

```
public static void parejas(ArrayList<Integer> array, Integer m){


}
```

Dicho método recibe un array de *N* enteros (`array`) y muestra por pantalla todos los pares cuya suma sea igual a un valor `m`. La complejidad del método desarrollado deberá ser O(n).

Como ejemplo, dado un vector de entrada `[1 3 12 23 1 2 2]`, y tomando `m` valor 4, la salida obtenida debería ser: `{1, 3} {3,1}, {2,2}`