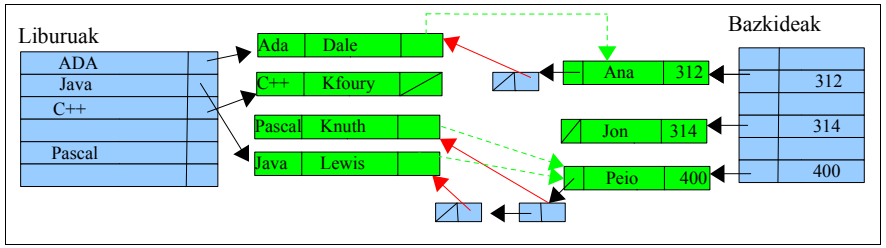
2 (X puntu)
Liburutegi bateko liburuak, bazkideak eta maileguak errepresentatzeko bi inplementazio ditugu:

Inplementazio 1: Zerrenda Estekatuak
Liburutegiak bi zerrenda dauzka: liburuen zerrenda eta bazkideen zerrenda. Bi zerrendak ordenatuta daude, lehenengoa alfabetikoki liburuaren izenburuagatik, eta bigarrena bazkideen karneta zenbakiagatik.

Liburu zerrendako elementu bakoitzak izenburua, idazlea eta erakusle bat maileguan duen bazkideari dauka. (liburua, liburutegian badago erakusle hori null balioa edukiko du). Bazkide zerrrendako elementu bakoitzak izena, karneta zenbakia eta bazkideen mailegu zerrenda (liburutegitik atera dituen liburuak) dauka. Mailegu zerrendako elementu bakoitzak, erakusle bat dauka maileguan daukan liburuari.

```
class Liburutegia {
   LinkedList<Bazkide> zBazkideak;
   LinkedList<Liburu> zLiburuak;
}
class Bazkide {
   String izena;
   int zenb;
   LinkedList<Liburu> zBazkideLiburuak;
}
class Liburu {
   String izenburua;
   String egilea;
   Bazkide maileguBazkide;
}
```

Inplementazio 2: Hash taulak
Liburutegi klaseak, bi hash taula dauzka: liburuen eta bazkideen taula. Hurrengo irudian, liburutegi bat aurkezten da 4 libururekin eta 3 bazkiderekin.



Liburuak taulako elementu bakoitzak izenburua, idazlea eta erakusle bat maileguan duen bazkideari dauka. (liburua, liburutegian ez badago erakusle hori null balioa edukiko du).
Bazkideak taulako elementu bakoitzak izena, karnet zenbakia eta bazkideen mailegu zerrenda (liburutegitik atera dituen liburuak) dauka. Mailegu zerrendako elementu bakoitzak, erakusle bat dauka maileguan daukan liburuari.

```
class Liburutegia {
   Hashtable<Integer, Bazkide> bazkideak;
   Hashtable<String, Liburu> liburuak;
}
class Bazkide {
   String izena;
   int zenb;
   LinkedList<Liburu> zBazkideLiburuak;
}
class Liburu {
   String izenburua;
   String egilea;
   Bazkide maileguBazkide;
}
```

Honakoa eskatzen da:
A) **Diseinatu eta inplementatu** `bazkideaTransferitu` metodoa liburutegiaren bi inplementazioetarako.

public void bazkideaTransferitu **(int zahark, int berk, String beriz);**
-- Aurre*: zahark* eta *berk* bazkide zaharraren eta berriaren karnet zenbakiak dira, hurrenez hurren. beriz bazkide berriaren izena da. *zahark* liburutegian dago, eta *berk* ez dago liburutegian.
-- Post: Liburutegian *zahark* karnetadun bazkidea ezabatu da, eta *berk* karnetadun bazkidea sartu da. Ezabatutako *zahark* bazkideak zeuzkan mailegu guztiak *berk* bazkide berriari pasa zaizkio.
B) Esan, **modu arrazoituan**, zein den A ataleko inplementazio bakoitzaren **konplexutasun-ordena**. Kontuan hartu lista sekuentzialeko metodoak ordena konstantekoak direla, eta bazkide batek eduki ditzakeen maileguen kopuru maximoa balore konstante batek mugatzen duela.

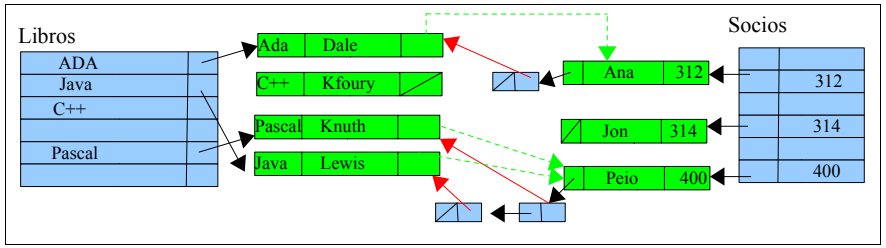
2 (4 puntos)
Tenemos 2 implementaciones para representar los libros, socios y préstamos de una biblioteca:

Implementación 1: listas enlazadas
La biblioteca tiene 2 listas: una para libros y otra para socios. Las 2 listas están ordenadas, la primera alfabéticamente por título del libro, y la segunda por el número de carné del socio.

Cada libro tiene su título, autor y una referencia al socio que ha tomado prestado el libro (con valor null si el libro no ha sido prestado). Cada socio tendrá su nombre, número de carné y una lista de los préstamos de ese socio. Cada elemento de la lista de préstamos es una referencia al libro prestado.

```
class Biblioteca {
   LinkedList<Socio> listaSocios;
   LinkedList<Liburu> listaLibros;
}
class Socio
   String nombre;
   int numero;
   LinkedList<Libro> listaLibrosPrestados;
}
class Libro {
   String titulo;
   String autor;
   Socio quienLoTiene;
}
```

Implementación 2: tablas Hash
La clase Biblioteca tiene dos tablas hash: una de libros y otra de socios. La siguiente figura muestra una biblioteca con 4 libros y 3 socios:




```
class Biblioteca {
   Hashtable<Integer, Socio> socios;
   Hashtable<String, Libro> libros;
}
class Socio {
   String nombre;
   int numero;
   LinkedList<Libro> listaLibrosPrestados;
}
class Libro {
   String titulo;
   String autor;
   Socio quienLoTiene;
}
```

Se pide lo siguiente:
A) Diseñar e implementar el método transferirSocio en las 2 implementaciones.

public void transferirSocio (int carneViejo, int carneNuevo, String nombreNuevo);
// Pre: carneViejo y carneNuevo son los números de carné del socio viejo y nuevo
// nombreNuevo es el nombre del nuevo socio. carneViejo se encuentra en la
// biblioteca, y carneNuevo no.
// Post: se ha borrado el socio de número carneViejo de la biblioteca, y se ha añadido el socio de
// nombre carneNuevo. Además, se han pasado todos los préstamos de carneViejo al
// nuevo socio.

B) Decir, de manera razonada, cuál es el coste de cada una de las implementaciones del apartado A. Tener en cuenta que las operaciones de la clase LinkedList son de tiempo constante, y que el número máximo de préstamos que puede tener un socio viene dado por una constante.

4. Red social (1,5 puntos)

Queremos diseñar un TAD que representa una red social en la que unas personas están relacionadas con otras. Cada persona está relacionada con otras por medio de la relación de amistad. Una persona registrada en la red puede tener cero, uno o más amigos.

En concreto, hemos decidido que, entre otras, el TAD Red_Social contendrá las siguientes operaciones:

```
public interface iRedSocial {

    public SimpleLinkedList<Integer> amigos(Integer dni)
   // pre:
   // post: el resultado es la lista de amigos de "dni" (sus DNIs)
   //     Si no pertenece a la red o no tiene amigos, el resultado será la lista vacía

   public void añadir(Integer dniX, Integer dniY, Integer afinidad)
   // pre:
   // post: Si X ó Y no pertenecían a la red, han sido añadidos
   //       X e Y se han añadido como amigos a la red, con la afinidad indicada

   public boolean estanRelacionados(Integer dniX, Integer dniY)
   // pre:
   // post: el resultado es true si X e Y pertenecen a la red y además son amigos o
   //       existe una persona Z que es a la vez amiga de X e Y
   //       el resultado es false en caso contrario
}
```

Se ha propuesto la siguiente implementación para el TAD Red_Social:

```
public class RedSocial implements iRedSocial {
   private HashMap<String, Socio> miembros;              // clave: DNI
}

public class Socio {
   HashMap<String, Integer>  tablaAmigos; // clave: DNI, datos: afinidad
   SimpleLinkedList<String> listaAmigos; // contiene los DNIs de tablaAmigos
}
```

Se pide:

a. **Justifica razonadamente el coste** de **todas** las operaciones (en caso de usar nombres de variables como N, M, ó P, se debe especificar claramente qué representa cada una).

b. Se pide **diseñar e implementar la operación "estanRelacionados"**