# *DATU-EGITURAK ETA ALGORITMOAK I 2. MAILA*
## 2006ko EKAINAREN 26a

***1 ( 2 puntu)***

Node eta DoubleLinkedListklaseak emanda (eranskina I), eta zerrendan gordeta dauden elementuak **Integer** datu motakoak izanda suposatuz, diseina eta inplementatu hurrengo metodoa DoubleLinkedListItr klasean:

*public class **DoubleinkedList<T>** {*

*DoubleLinkedList<T> birkokatu(int zenb)*

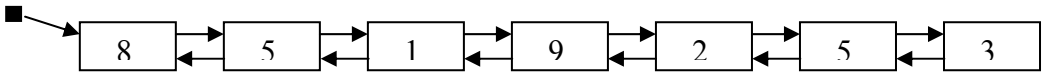
*}*

Metodo honek zerrendako zenbaki guztiak birkokatzen ditu zerrenda berri batean. Zerrenda berri honetan, *zenb* baino txikiagoak edo berdinak diren elementuak ezkerrean kokatuko dira eta *zenb* baino handiagoak direnak eskuinean. Prozesuaren bukaeran elementuen ordenak ez du axola, betiere aipatu den baldintza betetzen bada.

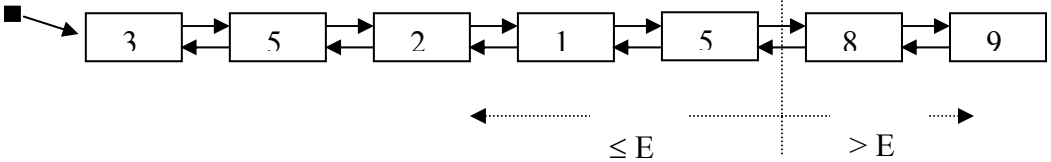
**O(N) ordenako (non N hasierako zerrendako elementuen kopurua den) ebazpenak hobeto baloratuko dira puntuazioan.**

Adibidea:
Izan bitez

zenb = 7




Emaitza zerrenda honelakoa izan daiteke:



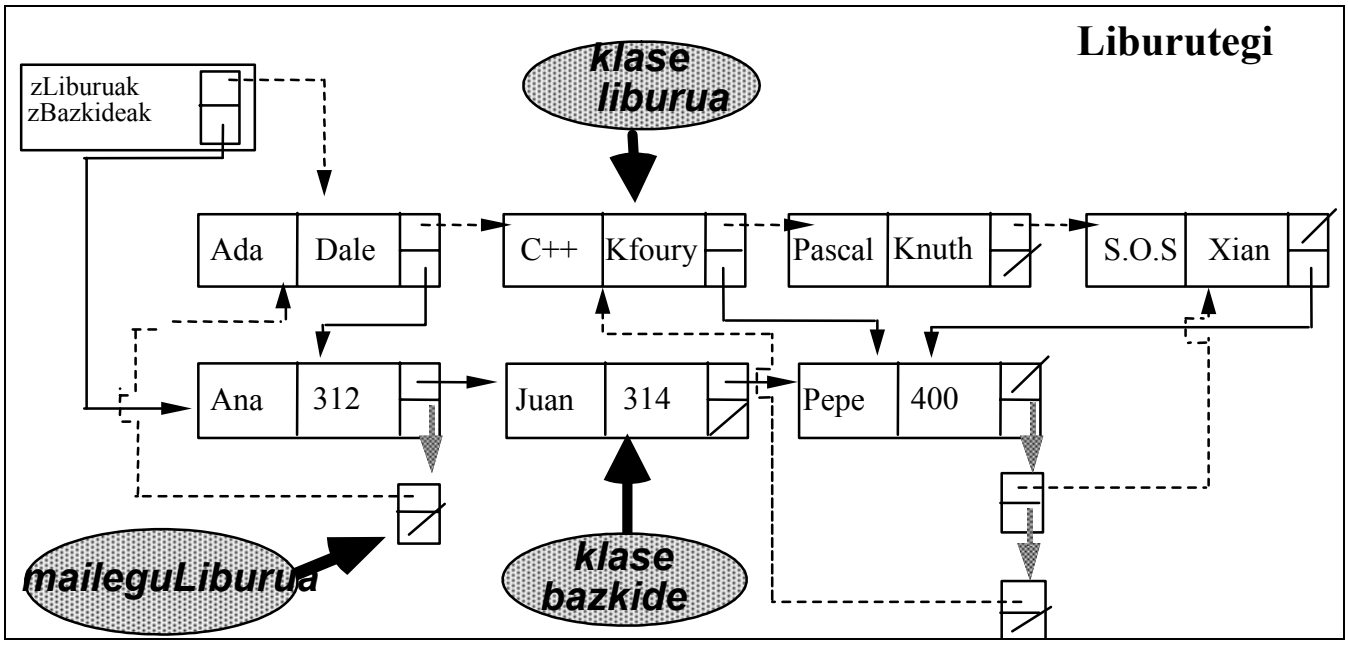

```
class Node<T> {
      T element;
      Node<T> next;
      Node<T> previous;
   }
public class DoubleLinkedList<T> {
      Node<actual> top;
      Node<T> actual;
}
```

*2 ( 4 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.
Hurrengo irudian, liburutegi bat aurkezten da 4 liburuekin eta 3 bazkideekin.

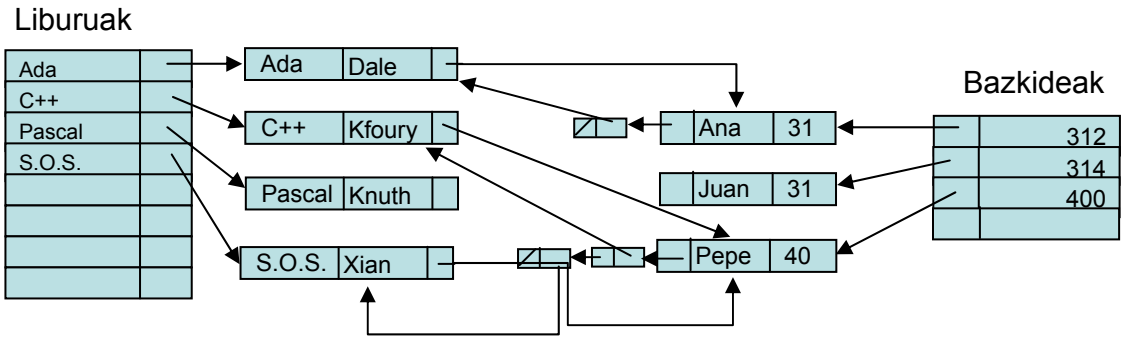



Liburu zerrendako elementu guztiak, izenburua, idazlea eta erakusle bat maileguan duen bazkideari dauka. (liburua, liburutegian badago erakusle hori null balioa edukiko du). Bazkide zerrrendako elementu guztiak, izena, karneta zenbakia eta bazkideen mailegu zerrenda (liburutegitik atera dituen liburuak) 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<Integer> zBakideLiburuak;
}
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 liburuekin eta 3 bazkideekin.




*Liburuak* taulako elementu guztiak, izenburua, idazlea eta erakusle bat maileguan duen bazkideari dauka. (liburua, liburutegian ez badago erakusle hori null balioa edukiko du). *Bazkideak* taulako elementu guztiak, izena, karneta 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> zBakideLiburuak;
}
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.

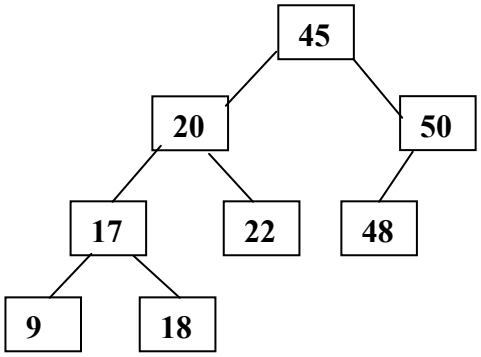
-- Post: Liburutegian *zahark* karnetadun bazkidea ezabatu da, eta *berk* karnetadun bazkidea sartu da. Ezabatutako *zahark* bazkideak zeuzkan mailegu guztiak *berk* bazkide berriari pasa zaizkio.

B) Esan, **modu arrazoituan**, zein den A ataleko inplementazio bakoitzaren **konplexutaxun-ordena**. Kontuan hartu lista sekuentzialeko metodoak ordena konstantekoak direla, eta bazkide batek eduki ditzakeen maileguen kopuru maximoa balore konstante batek mugatzen duela.

*3 ( 4 puntu)*

**Diseinatu eta inplementatu** Javan **Arbaso komun gertukoena** kalkulatzen duten metodoak.

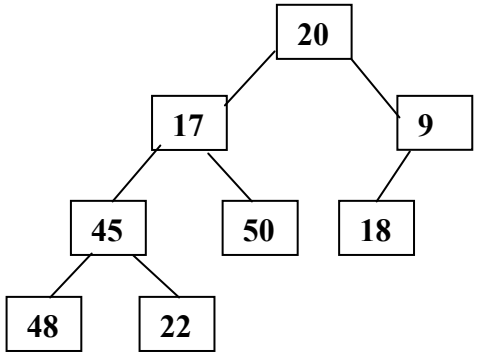
**a) Bilaketa-zuhaitz bitar** bat (elementu errepikaturik gabea), eta zuhaitzean dauden bi elementu emanda, bi elementu horien arbaso komun gertukoena itzultzen du.




Adibidez, honako BZB hau hartuta:

- 17 eta 48 elementuen arbaso komun gertukoena 45 da
- 9 eta 20 elementuen arbaso komun gertukoena 20 da
- 48 eta 50 elementuen arbaso komun gertukoena 50 da

**b) Zuhaitz bitar** bat (elementu errepikaturik gabea), eta zuhaitzean dauden bi elementu emanda, bi elementu horien arbaso komun gertukoena itzultzen du..




Adibidez, honako zuhaitz bitar hau hartuta::

- 17 eta 48 elementuen arbaso komun gertukoena 17 da
- 9 eta 20 elementuen arbaso komun gertukoena 20 da
- 48 eta 50 elementuen arbaso komun gertukoena 17 da

Zuhaitzaren egitura hurrengoa da:

```
class BTNode<T> {
      T content;
      BTNode<T> left;
      BTNode<T> right;
   }
public class ZuhaitzBitarra<T> {
      BTNode<T> root;
      BTNode<T> actual;
}
```

***HashTable<K,V>*** klasea

```
class HashTable<K,V> {
void insert (K key, V value);
-- pos: key gakoa duen elementua value objektuarekin txertatzen du. Gakoa
existitzen bda, ez du ezer egiten.
void modify (K key, V value)
--pos: key gakoak daukan elementua aldatzen da value elementuarengatik. Gakoa
existitzen ez bada, ez du ezer egiten.
V find (K key);
-- pos: key dagokion objektua itzultzen du. Gakoa taulan ez badago, ez du ezer
egiten.
void remove (K key);
-- pos: key gakoa duen elementua ezabatzen du hash taulatik. Gakoa existitzen
ez bada, ez du ezer egiten.
boolean exist (K key);
-- pos: true itzultzen du bsb key gakoa existitzen bada.
int numPos ();
-- pos: hash taularen elementu kopurua itzultzen du.
}
```

***LinkedList<T>*** klasea

```
class LinkedList<T>{

LinkedList();
-- pos: actual elementua zerrendaren lehenengoa da. list zerrenda hutsa bada,
actual elementua hutsa izango da.
void goNext ();
-- pos: actual zerrendaren hurrengo elementura pasa da. Hurrengo elementurik
ez badago, actual null izango da.
void goFirst();
-- pos: actual zerrendaren lehenengo elementuan kokatzen da.
T getActual();
-- aurre: actual existitzen da
-- pos: actual elementuraren edukia itzultzen du
boolean isEmpty ();
-- pos: true itzultzen du bss actual elementua null ez bada.
void insertFirst(T x)
-- pos: x elementua txertatzen du zerredaren hasieran. actual elementu berrian
kokatzen da
void insertNext(T x)
--pos: x elementua txertatzen du actual elementuaren jarraian. actual elementu
berrian kokatzen da
void insertPrevious(T x)
--pos: x elementua txertatzen du actual elementuaren aurrean. actual elementu
berrian kokatzen da
}
```