DATU-EGITURAK ETA ALGORITMOAK I **2. maila**
2006ko otsailaren 6a

1. ariketa: (3 puntu)

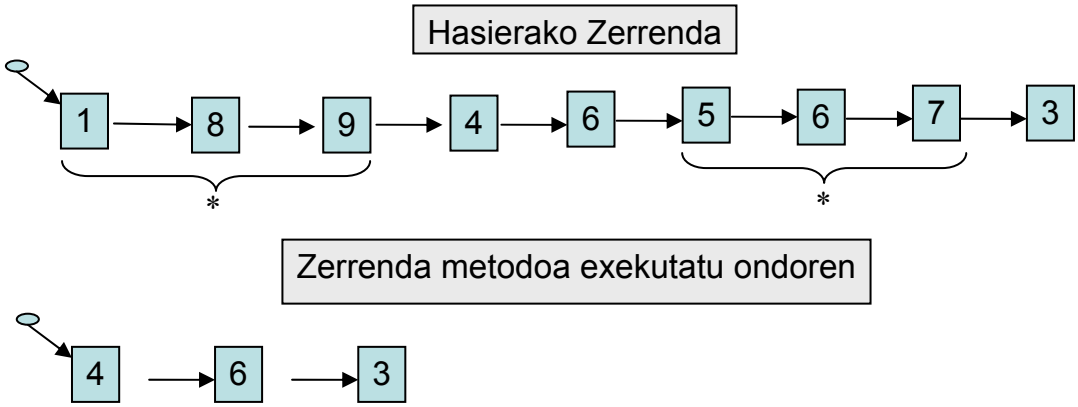
Elementu osoko esteka sinpleko zerrenda batean, diseina eta inplementatu hurrengo metodoa LinkedListItr klasean:

void **goranzkoKateakEzabatu**(int luz)

Metodoak *luz* luzeerako goranzko elementu sekuentziak ezabatzen ditu.

Adibidea:

z. goranzkoKateakEzabatu(3)

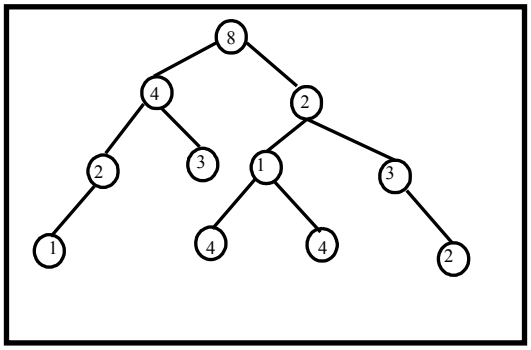


2. ariketa: (3 puntu)

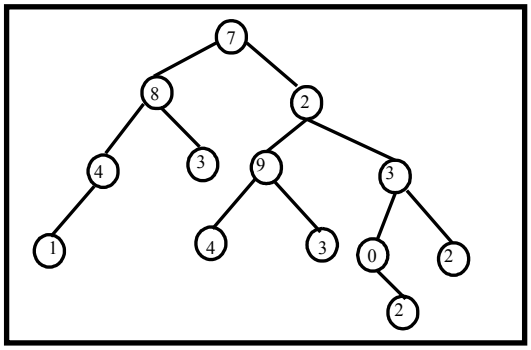
Osoko zenbakien zuhaitz bitar ez-huts bat emanda, zuhaitz hori **bideratuta** dagoen ala ez erabakitzen duen metodo bat espezifikatu, diseinatu eta inplementatu.

Zuhaitz bat **bideratuta** dago baldin eta soilik baldin errotik hostoetara doazen bere bide guztiek pisu berdina badute. Bide baten pisua bere adabegi guztien pisuen batura da. Zuhaitz hutsa bideratuta dago.

Zuhaitz bideratua



Zuhaitz EZ bideratua



3. ariketa: (4 puntu)

Liburutegi bateko funtzionalitateari eusteko egitura bat diseinatu da. Diseinurako honako murriztapenak hartu dira:
1. Liburu bat bazkide bakar bati dago mailegatzea,
2. Bazkide batek gehienez 4 liburu eduki ditzake maileguan hartuta,
3. Liburu bat autore batek baino gehiagok idatz dezakete,
4. Ez da mugatzen autore batek idatz ditzakeen liburuen kopurua.
5. Ez dira existitzen bi liburu izen berdinarekin

Liburutegiaren errepresentazioa partzialki ematen zaizue. Liburutegiak honako bi egitura hauek, behintzat, izango ditu:

- Liburutegiko **bazkideen informazioa** errepresentatzeko **hash taula** bat. Bazkide bakoitzari buruz honako informazioa edukiko da: karneta (desberdina bazkide bakoitzarentzat), izena, eta maileguan hartuta dauzkan liburuak.
- Liburuen **autoreen informazioa** errepresentatzeko **bilaketa-zuhaitz bitar** bat erabiliko da. Zuhaitza autoreen izenaren arabera dago ordenatuta, eta autore bakoitzeko, izenaz gain, berak idatziko liburutegiko liburuen informazioa edo erreferentziak gordeko dira.
- Liburuak gordetzeko ez da ezer esaten, eta egitura bat definitu beharko da.

Liburutegi klasearen metodoak honako hauek dira:

Liburutegi();	Eraikitzailea, egitura hasieratzen duena
boolean **liburuaDago**(String titulo)	post: True itzultzen du bsb. *titulo* horretako Liburua liburutegian badago
boolean **egileaDago**(String izena)	post: True itzultzen du bsb. *izena* horretako Egilea liburutegian badago
Liburu **getLiburu**(String titulo)	post: Titulo baten liburuari buruzko informazioa itzultzen du
void **insertLiburu**(String titulo, LinkedListItr autore)	aurre: titulo horretako liburua ez dago liburutegian post: liburutegian sartu ditu liburua eta autore-listako autoreak. Liburua ez dago mailegatuta.
void **insertMailegu**(String titulo, int bazkideZenb)	aurre: bazkidea liburutegian dago post: titulo izeneko liburua mailegatzen zaio bazkideZenb bazkideari, liburua existitzen bada liburutegian eta libre badago. Hala ez bada, dagokion errore-mezua idatziko da.
LinkedListItr **getLiburuak**(String autore)	aurre: autorea liburutegian dago post: Liburutegian dauden autorea-ren liburu guztiak itzultzen ditu

Honakoa eskatzen da:

1. **Osatu errepresentazioa**, eta idatzi Java-n liburutegiko informazioa egoki kudeatzeko behar diren **klase guztien definizioak** (liburutegi osoko errepresentazioa). Horretarako kontuan hartu algoritmoen kostuek liburu kopurutik, bazkide kopurutik eta autore batek idatzitako liburu kopurutik independente izan behar dutela. Hau da, orokorrean, lehen espezifikatutako eragiketa guztietarako, **algoritmoen kostuak lineala baino hobea izan behar du**. **Adierazi garbi zein den erabilitako klase guztien elementuen mota**, bai emandakoetan (autoreen bilaketa-zuhaitz bitarrean eta bazkideen hash taulan) bai zuek asmatutakoan. **Marraztu liburutegi egitura oso-osorik.**

2. **Diseinatu eta inplementatu Javan** `Liburutegi` klaseko honako metodoak: `eraikitzailea`, `liburuaDago`, `insertLiburu` **eta** `insertMailegu`.

3. **Konplexutasunaren azterketa**. Azaldu argi eta garbi, eta egindako diseinuak aipatuz, zein den *liburuaDago*, *insertLiburu* eta *insertMailegu* algoritmoen kostua, BZ bazkide kopurua eta LZ liburu zenbakia izanda.

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> {
void goFirst()
--  actual lehendabiziko elementuan kokatzen da
void goNext()
-- actual hurrengo elementuan kokatzen da
boolean hasMoreElements()
-- true actual null bada, false kontrako kasuan
T getActual();
-- actual posizioaren edukia itzulzen du
void insert(T o);
-- actual elementuaren jarraian o objektua txertatzen du
void insertFirst(T o)
-- o objektua zerrandaren hasieran txertatzen du

}
```

BilaketaZuhaitzBitarra<T> klasea

```
class BZB<T> {
//actual erroan kokatzen da
void goRoot()
// actual mugitzen da actual adabegiaren ezkerreko adabegira
void goLeft()
// actual mugitzen da actual adabegiaren eskubiko adabegira
void goRight()
// actual posizioaren edukia itzulzen du
T getActual();
// o objektua zuhaitzen txertatzen du dagokion posizioan
void insert(T o);
}
```

Eranskin I: Zerrendaren egitura

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

Eranskin II: BTNode eta BinTreeegiturak

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