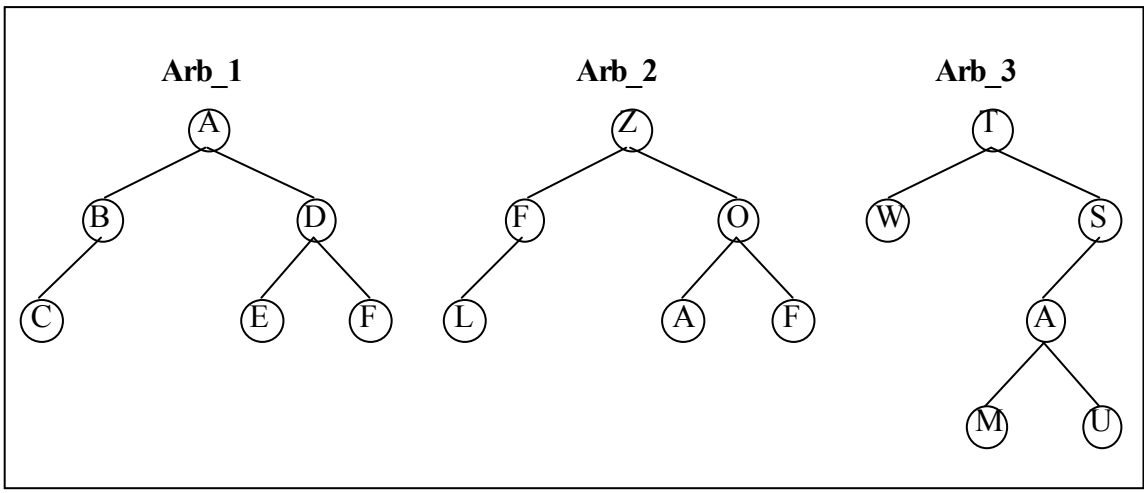
1. Egitura bereko zuhaitzak

Bi zuhaitz bitar emenda, egituraBera izeneko funtzioa diseinatu, zeinek true bueltatuko duen zuhaitzek egitura bera baldin badute (zuhaitzek egitura bera dute, adabegietako balioak ezik).

Irudiko adibidean, egituraBera(Arb_1, Arb_2) deiak true bueltatuko du, eta egituraBera(Arb_1, Arb_3) deiak, aldiz, false.



2.- Maila bateko balio maximoa

Zuhaitz bitar bat eta zuhaitzeko maila bat adierazten duen balio bate manda, apiprograma bat egin nahi dugu, maila horretako balio maximoa bueltatuko duena.

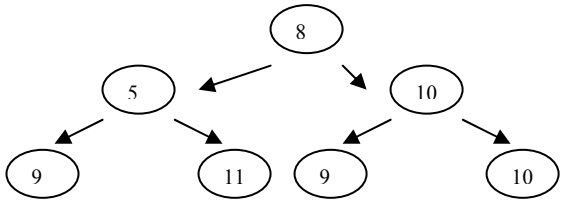
Azpiprograma egiteko, hau eskatzen da:

a) Diseinu errekurtsiboko pausoak

b) Inplementazioa Javaz

public int maxValor (BinTree<T> a, int n)

Adibidez, zuhaitz hau emanda:

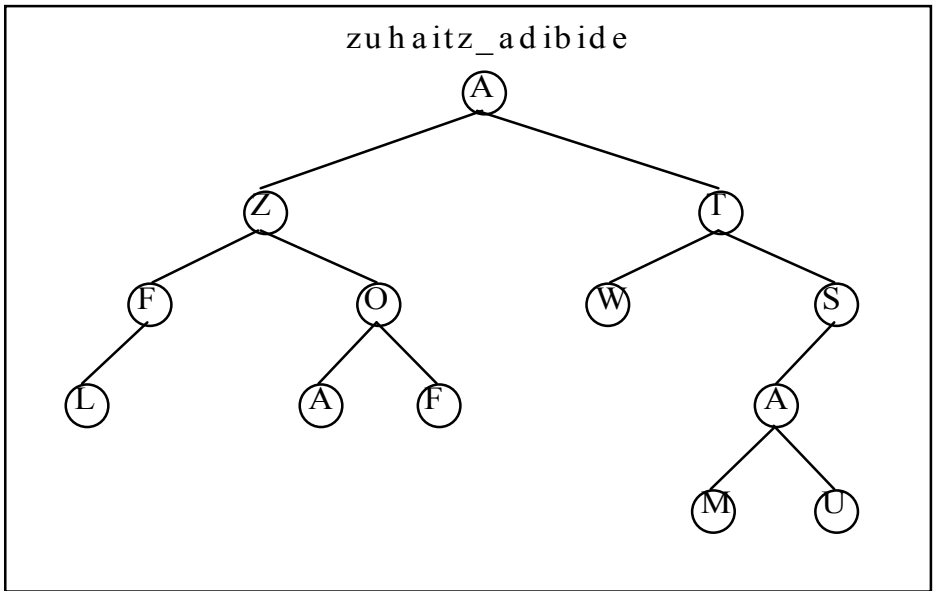


maxValor (a, 1) = 8
maxValor (a, 2) = 10
maxValor (a, 3) = 11

3. Adabegi-kopurua maila batean

Diseinatu eta inplementatu Javaz adabegiKop funtzioa, zeinek zuhaitz bitarra eta maila bat emanda, zuhaitz horretan maila horretan dagoen adabegi-kopurua bueltatuko duen.
Irudian, adibidez:

- adabegiKop (zuhaitz_adibide, 1) 1 bueltatuko luke
- adabegiKop (zuhaitz_adibide, 3) 4 bueltatuko luke
- adabegiKop (zuhaitz_adibide, 5) 2 bueltatuko luke



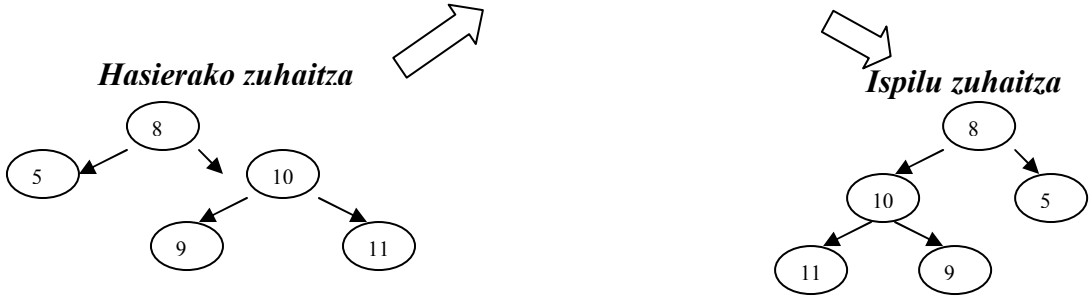
4.- ***Zuhaitz ispilua***

Zuhaitz bitarra emanda, bere ispilua izango den beste zuhaitz bat lortu nahi da. Emaitzak adabegi berdinak izango ditu, ispiluan ikusten diren bezala.

Ispilu funtzioa hasierako zuhaitza aldatu gabe egin nahi da. Soluzioan hau aurkeztuko da:

c) Diseinu errekurtsiboko pausuak
d) Inplementazioa Javaz

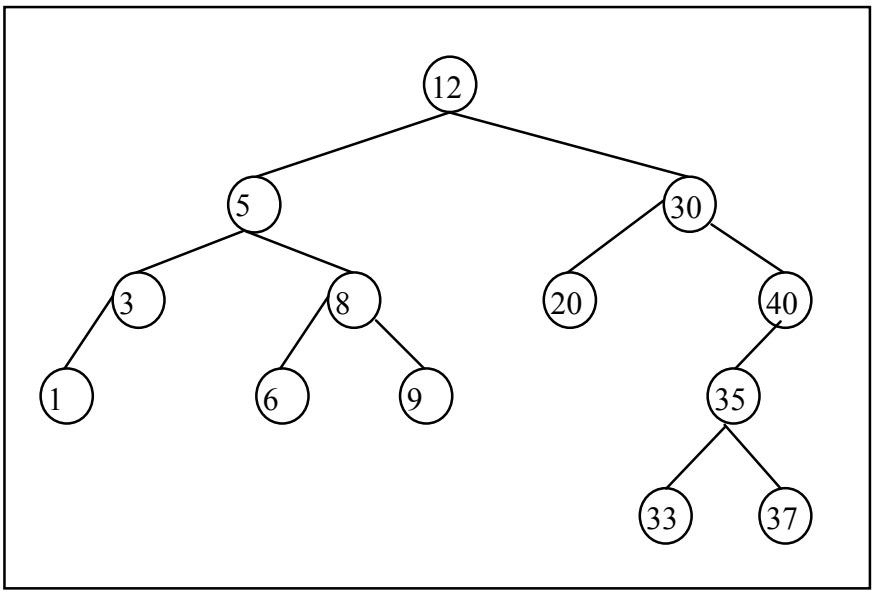
public BinTree<T> ispilu (BinTree<T> a)



Adibidean ikusten da nola ezkerrean zeuden adabegiak eskuinera pasatu diren eta alderantziz.

5. Lista ordenatua

Azpiprograma bat diseinatu, bilaketa-zuhaitz bitar bat emanda, bere balioen zerrenda ordenatua bueltatuko duena.




```
public LinkedList<T> listaOrdenatua()
// pre:
// post: emaitza zuhaitzeko elementuen lista ordenatua da
```

Zerrendaren mota honela definitua da:

```
public class LinkedList<T> {
   LinkedNode<T> first;
   LinkedNode<T> last;
}
public class LinkedNode<T> {
   T data;
   LinkedNode<T> next;
}
```

Adibidez, irudiko zuhaitza emanda, lista hau lortuko da:

(1, 3, 5, 6, 8, 9, 12, 20, 30, 33, 35, 37, 40).

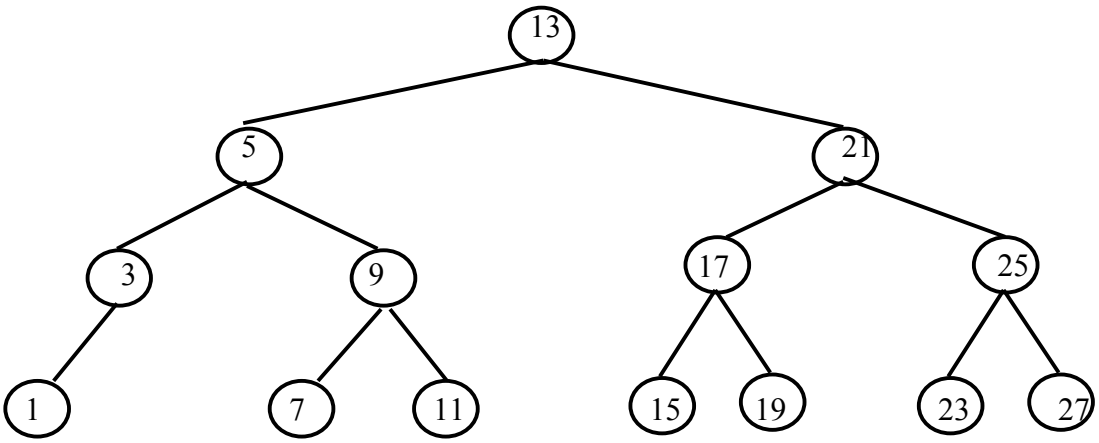
Soluzioa denbora linealekoa izango da O(n).

6.- Sortu BZB bat

Osokoen array bat emanda, bere balioak goranzko ordenan daudelarik, idatzi azpiprograma bat bilaketa-zuhaitz bitar orekatua lortzeko (hau da, ezkerreko eta eskuineko azpizuhaitzen adabegi-kopuruen diferentzia gehienez bat izango duena).

```
public BinTree<T> surtuBZBOrekatua(T[] taula)
   // aurre: taula gorantz ordenatuta dago
   // post: taula-ko "a" zuhaitza sortu da, BZBa da
   //       eta orekatuta dago azpizuhaitzetako adabegi-kopuruen arabera
```

Adibidez: taula = (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27) emanda, emaitza ondokoa izango da:



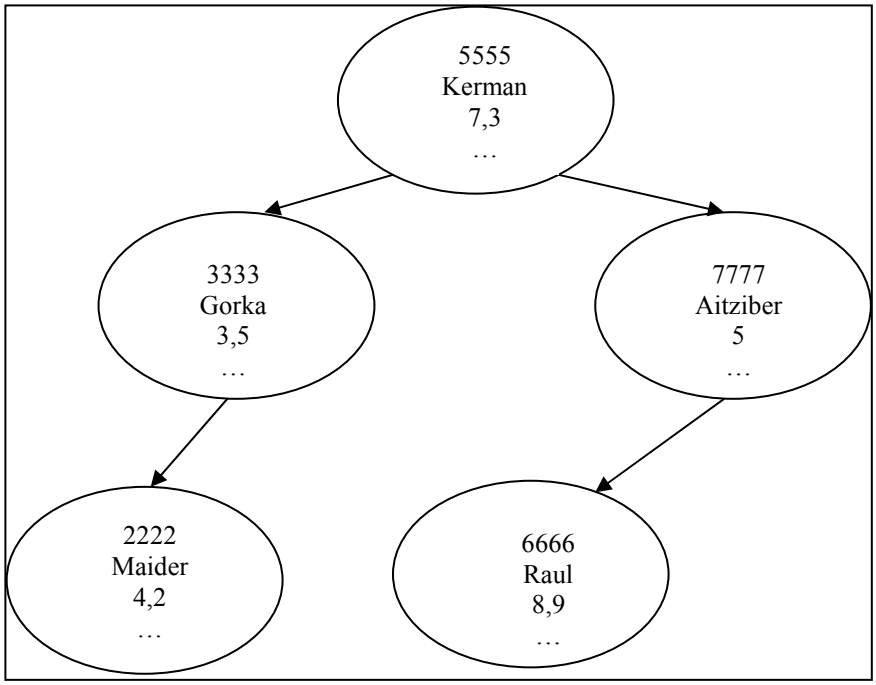
7. Listatu ordenatua bi Bilaketa Zuhaitz Bitarrekin

DEA irakasgaiko ikasleei buruzko informazioa Bilaketa Zuhaitz Bitar bitan dago gordeta:

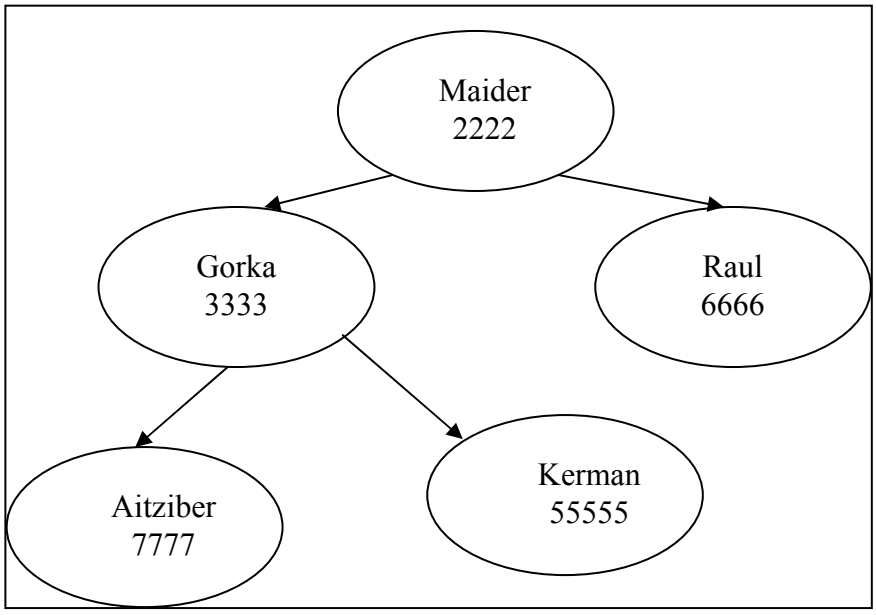
- BZB_NA. Zuhaitz hau NA zenbakiaren arabera ordenatuta dago. Adabegi bakoitzean ikasle baten datuak (NA Zenbakia, Izena, Batezbesteko Nota, eta beste datu batzuk) dauzkagu.
- BZB_IZ. Zuhaitz hau izenaren arabera ordenatuta dago. Zuhaitz honetako adabegi bakoitzean ikasle baten 2 datu (Izena eta NA Zenbakia) daukagu.

Adibidez:

BZB_NA



BZB_IZ



Hauek dira datu mota erazagupenak:

```
public class IkasleID {
   String izena;
   String na;
}

public class IkasleDatuak {
   IkasleID id;
   double nota;
}
```

Honako hau eskatzen da:

a) Ondoko prozedura inplementatu:

```
public void lortuListatuOrdenatua(BinSearchTree<T> IZ,
                                  BinSearchTree<T> NA)
  // aurre: IZ bilaketa zuhaitz bitarra izenaren arabera ordenatuta dago.
  //      NA bilaketa zuhaitz bitarra NA zenbakiaren arabera ordenatuta dago.
  //      Zuhaitz bietan ikasle berberak daude
  //      Ez dago ikasle-izen errepikaturik.
  // post:  Listatu ordenatua, izenaren arabera, pantailan idatzi da.
```

Aurreko adibidea kontuan hartuta, hau izango litzateke lortu behar den listatua:

```
Izena       NA Zenbakia             Batezbesteko Nota
-----------------------------------------------------
Aitziber          7777                      5
Gorka             3333                      3,5
Kerman            5555                      7,3
Maider            2222                      4,2
Raul              6666                      8,9
```

b) Kalkulatu, modu arrazoituan, algoritmoaren kostua. Aldagaiak erabiliz gero, aldagai bakoitza zer den adierazi behar da.

8. Deskargen bilaketa zuhaitz bitarra (2,5 puntu)

Erabiltzaileek deskargatutako fitxategiak kudeatzeko *Bilaketa-Zuhaitz Bitar* bat eman zaigu. Adabegi bakoitzean honako informazio hau dago:

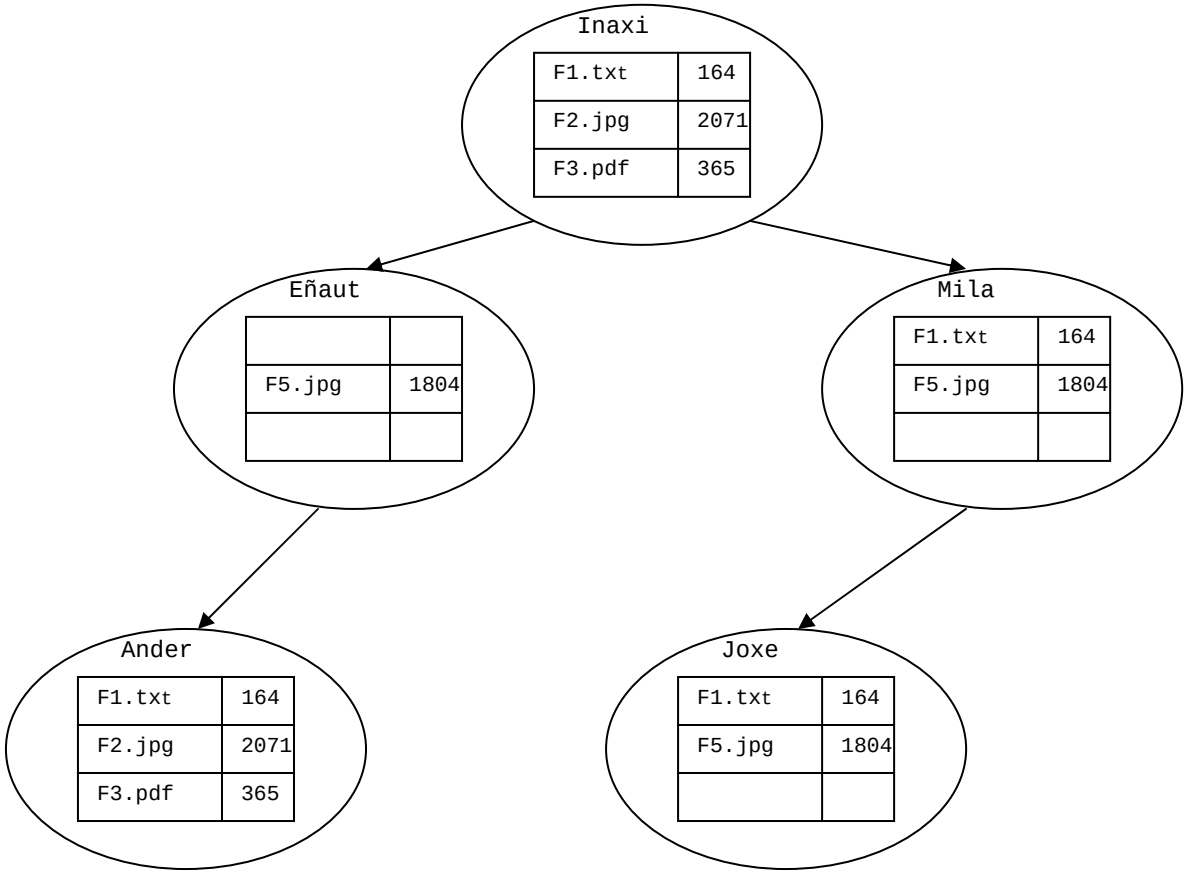
- Erabiltzailearen izena.
- Hash Taula bat, non deskargatutako fitxategien informazioa dagoen: fitxategiaren gakoa (gako gisa erabiltzen dena), fitxategiaren izena eta tamaina (zenbat Kb-takoa den).

Bilaketa-Zuhaitz Bitarra erabiltzaile-izenaren arabera dago ordenatua. Aurre-baldintza da ez dagoela erabiltzaile-izen errepikaturik.

Honakoa eskatzen da:

a. Definitu Adan datu-egitura hori.
b. Espezifikatu, diseinatu eta inplementatu Adan azpiprograma bat, fitxategi-izen bat emanda, bi emaitza hauek itzultzen dituena:
   - Zenbat aldiz deskargatu den fitxategi hori, eta
   - Zerrenda estekatu bat fitxategi hori deskargatu duten erabiltzaileen izenekin.

Adibidez:



Adibide honetan, “F5.jpg” fitxategi-izena emanda, honako emaitzak itzuli beharko lituzke azpiprogramak:

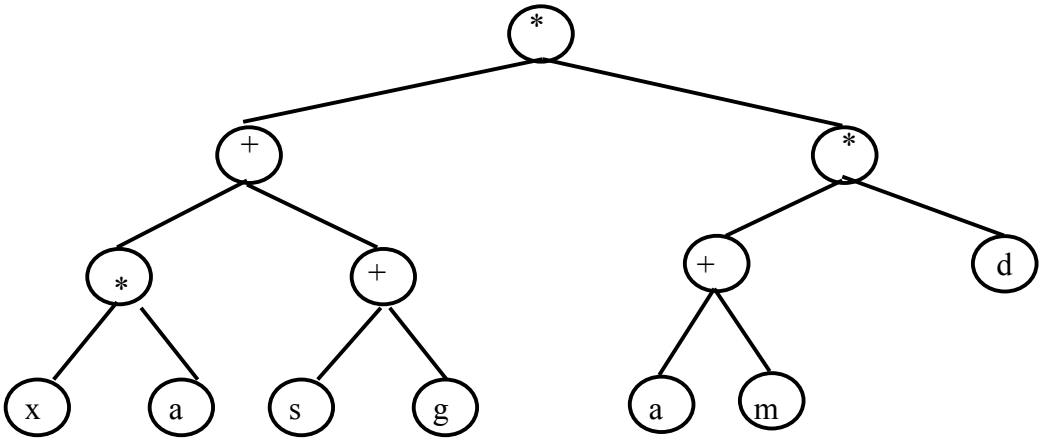
- Deskarga-kopurua: 3
- Erabiltzaileen zerrenda: <”Eñaut”, “Joxe”, “Mila”>

9.- Espresio aritmetikoa ebaluatu

Azpiprograma bat egin nahi dugu, sarreratzat 2 zuhaitz hartuko dituena. Bata espresio aritmetikoa adierazten duen zuhaitza da, eta bestea programa baten exekuzioaren une batean aldagaien balioak gordetzen dituena.

Espresio aritmetikoak bi adabegi-mota ditu: aldagaiak eta eragileak (sinplifikatzearren, batuketa eta biderketarako eragileak bakarrik onartuko dira).

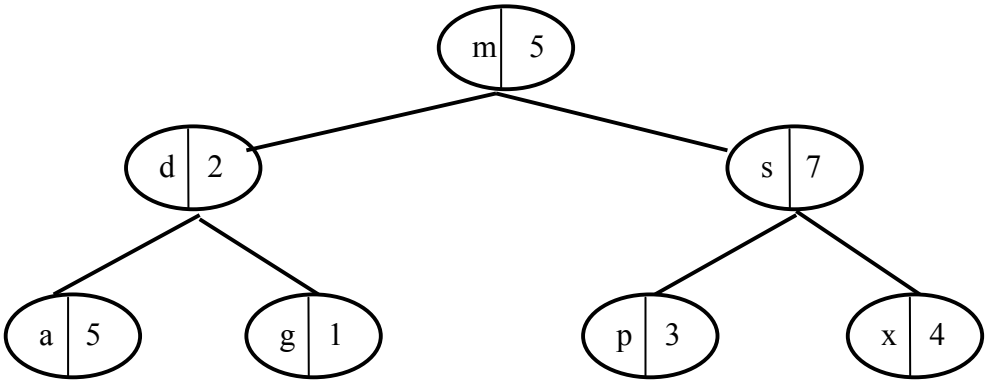
```
public class InfoElemEspresioa {

   String elem;       // *, +, edo aldagai baten izena
   boolean operador; // true -> eragilea, false -> aldagaia
}
```



Zuhaitz horrek espresio hau adierazten du: ((x * a) + (s + g)) * ((a + m) * d)

Aldagaien balioak dauzkan zuhaitza bilaketa zuhaitz bitarra da, aldagaiaren izenaren arabera ordenatuta, eta adabegi bakoitzak ondoko informazioa du:

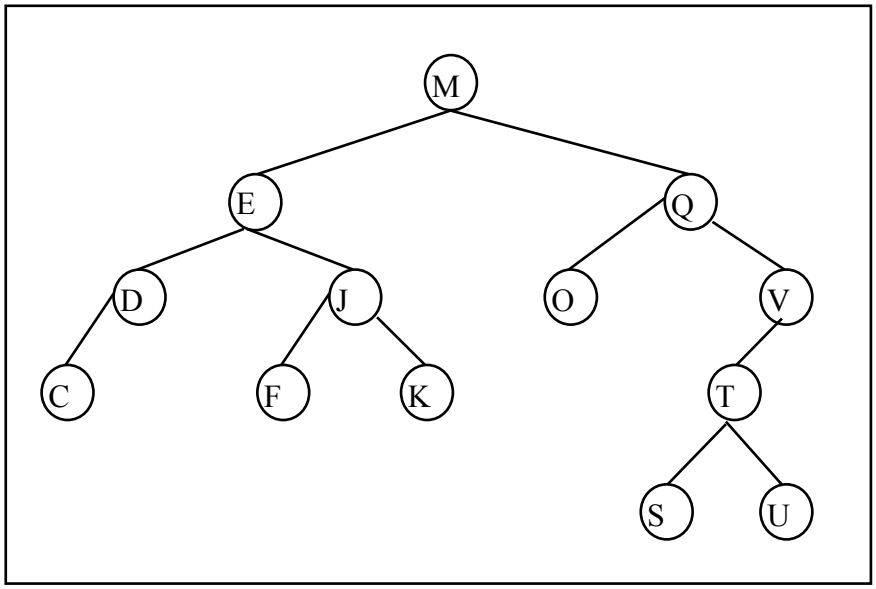
```
public class InfoAldagaia {

   String nom;
   int valor;
}
```



Azpiprogramak espresio horren ebaluazioaren emaitza eman beharko du..
Aurreko adibidean emaitza hau da: 560 = ((4 * 5) + (7 + 1)) * ((5 + 5) * 2)

3. Eraginkortasunaren kalkulua (2,5 puntu)

Bilaketa-zuhaitz bitarra dugu, adabegi bakoitzak String motako elementua duelarik. Bilaketetan aztertuko den batezbesteko elementu-kopurua kalkulatu nahi da, eta horretarako zenbatu egingo dira zerrenda bateko elementuen bilaketan aztertuko diren elementuak.



Funtzio hau inplementatu nahi da:

```
public class BinaryTreeNode<T> {
       protected T content;
       protected BinaryTreeNode<T> left;
       protected BinaryTreeNode<T> right;
}

public class BinarySearchTree<T> {
       protected BinaryTreeNode<T> root;
       protected int count;
}


public class NireZuhaitza extends BinarySearchTree<String> { // Herentzia

   public float batezbestekoElementuak(SimpleLinkedList<String> l)
   // Aurre: l listako elementu guztiak zuhaitzean daude
   // Post: emaitza l zerrendako elementuak bilatzean aztertutako
   //       elementuen batezbestekoa izango da
}
```

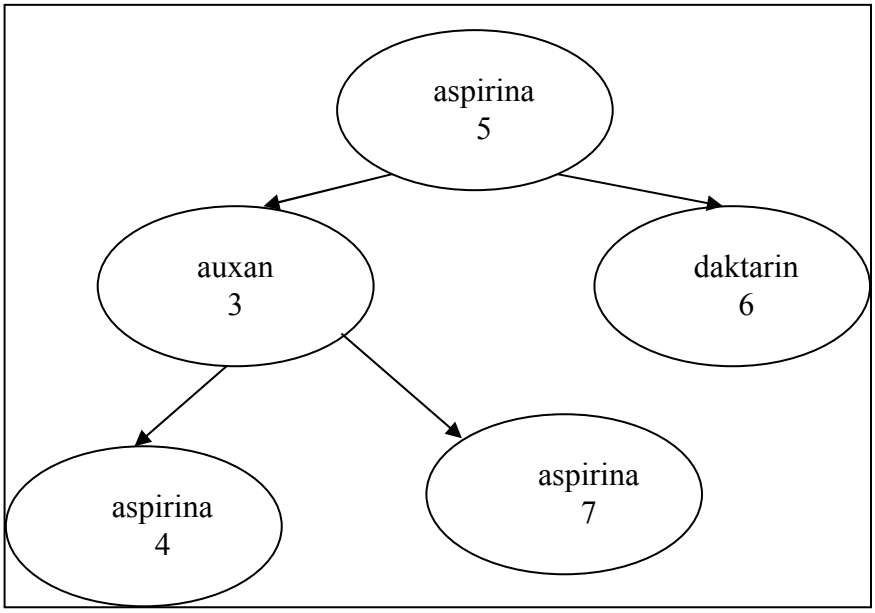
Adibidez, sarrerako zerrenda hau balitz: <T, M, E, S>, emaitza hau izango litzateke:

- 4 elementu aztertuko dira T bilatzeko
- 1 elementu aztertuko da M bilatzeko
- 2 elementu aztertuko dira E bilatzeko
- 5 elementu aztertuko dira S bilatzeko

Eta 3 izango da azken emaitza = (4 + 1 + 2 + 5) / 4

4. Zuhaitzaren bihurketa (1,5 puntu)

Produktuen salmentak dituen zuhaitz bitarra emanda (adabegi bakoitzean (produktua, saldutako unitateak) moduko bikotea), zuhaitz horretako elementuak hash taula batera pasako dituen azpiprograma nahi dugu, produktu bakoitzeko elementu bakarra emanez, produktu horren salmenta guztiekin.



Adibideko zuhaitza emanda, lortuko den hash taulak hau izango luke:

0		
1	daktarin	6
2		
3	aspirina	16
4	auxan	4
5		

```
public class BinaryTreeNode<T> {
       protected T content;
       protected BinaryTreeNode<T> left;
       protected BinaryTreeNode<T> right;
}

public class BinaryTree<T> {
   protected int count;
   protected BinaryTreeNode<T> root;
}

public class Produktu {
   int salmentak;
   String izena;
}

public class ProduktuenZuhaitza extends BinaryTree<Produktu> {

   public HashMap<String, Integer> zuhaitzaHTBihurtu()
}
```