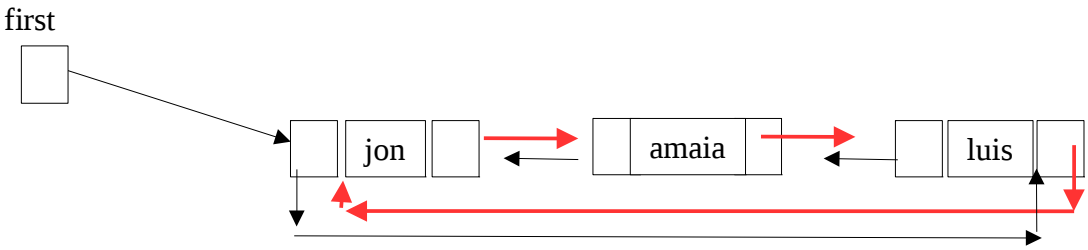
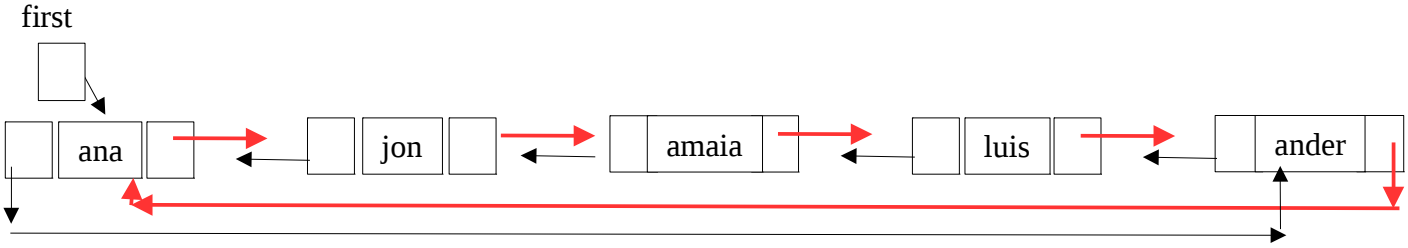
1. Borrar sublista.(1,5 puntos)
Tenemos 2 listas doblemente ligadas:

l1



l2

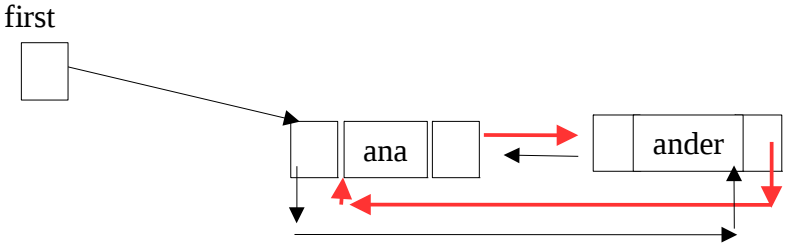


Se quiere implementar el siguiente método:

```
public class DoubleNode<T> {
   T data;
   DoubleNode<T> next;
   DoubleNode<T> prev;
}

public class DoubleLinkedList<T> {
   DoubleNode<T> first;

   public void borrarLista(DoubleLinkedList<T> subLista)
   // Precondición: "subLista" es una parte de la lista
   //     los elementos de "subLista" están en el mismo orden que en la lista principal
   // Postcondición: se han quitado los elementos de "subLista"


}
```

Por ejemplo, l2.borrarLista(l1) obtendría el siguiente resultado:

l2



Se pide:

- Implementar el método
- Calcular razonadamente el coste del algoritmo

2. Evaluación de expresión aritmética (1,5 puntos)

Djikstra presentó un algoritmo para la evaluación de una expresión aritmética. Por ejemplo: `(1+((2+3)*(4*5)))`

Se pide implementar el siguiente método:

```
public double evaluar(String exp)
 // Precondición: La expresión aritmética es correcta.
 //      La expresión está TOTALMENTE parentizada, es decir,
 //                por cada pareja de operandos se tienen paréntesis
 //      Todos los operadores son binarios (de la forma "X OPERADOR Y")
 // Postcondición: el resultado es el valor de la expresión
```

Dada la expresión ejemplo anterior, el resultado será 101.

La expresión aritmética parentizada se evaluará de la siguiente manera:

- Se usan 2 pilas, una para operadores y otra para operandos
- Los caracteres de la secuencia de entrada se tomarán de uno en uno, haciendo lo siguiente a cada paso:
  - Si es un paréntesis izquierdo, no se hace nada
  - Meter operandos en la pila de operandos
  - Meter operadores en la pila de operadores
  - Si es un paréntesis derecho, entonces
    - coger 2 operandos de la pila de operandos
    - coger un operador de la pila de operadores
    - calcular el resultado y meter en la pila de operandos

Ejemplo

Dada la siguiente cadena: `(1+((2+3)*(4*5)))`

Pila de operadores	
Pila de operandos	

- Paréntesis izquierdo "(" → no hacer nada

`Cadena: 1+((2+3)*(4*5)))`

- "1" → meter en pila de operandos

Pila de operadores	
Pila de operandos	1

`Cadena: +((2+3)*(4*5)))`

- "+" → meter en pila de operadores

Pila de operadores	+
Pila de operandos	1

Cadena: ((2+3)*(4*5)))

- Paréntesis izquierdo "(" → no hacer nada
- Paréntesis izquierdo "(" → no hacer nada

Cadena: 2+3)*(4*5)))

- "2" → meter en pila de operandos

Pila de operadores	+
Pila de operandos	1 2

Cadena: +3)*(4*5)))

- "+" → meter en pila de operadores

Pila de operadores	+ +
Pila de operandos	1 2

Cadena: 3)*(4*5)))

- "3" → meter en pila de operandos

Pila de operadores	+ +
Pila de operandos	1 2 3

Cadena:)*(4*5)))

- ")" → realizar operación

Pila de operadores	+
Pila de operandos	1 5

Cadena: *(4*5)))

- "*" → meter en pila de operadores

Pila de operadores	+ *
Pila de operandos	1 5

Cadena: (4*5)))

- Paréntesis izquierdo "(" → no hacer nada

Cadena: 4*5)))

- "4" → meter en pila de operandos

Pila de operadores	+ *
Pila de operandos	1 5 4

Cadena: *5)))

- "*" → meter en pila de operadores

Pila de operadores	+ * *
Pila de operandos	1 5 4

Cadena: 5)))

- "5" → meter en pila de operandos

Pila de operadores	+ * *
Pila de operandos	1 5 4 5

Cadena:)))

- ")" → realizar operación

Pila de operadores	+ *
Pila de operandos	1 5 20

Cadena:))

- ")" → realizar operación

Pila de operadores	+
Pila de operandos	1 100

Cadena:)

- ")" → realizar operación

Pila de operadores	
Pila de operandos	101

¡El resultado está en la pila de operandos!

Implementar el algoritmo y calcular su costo de manera razonada.

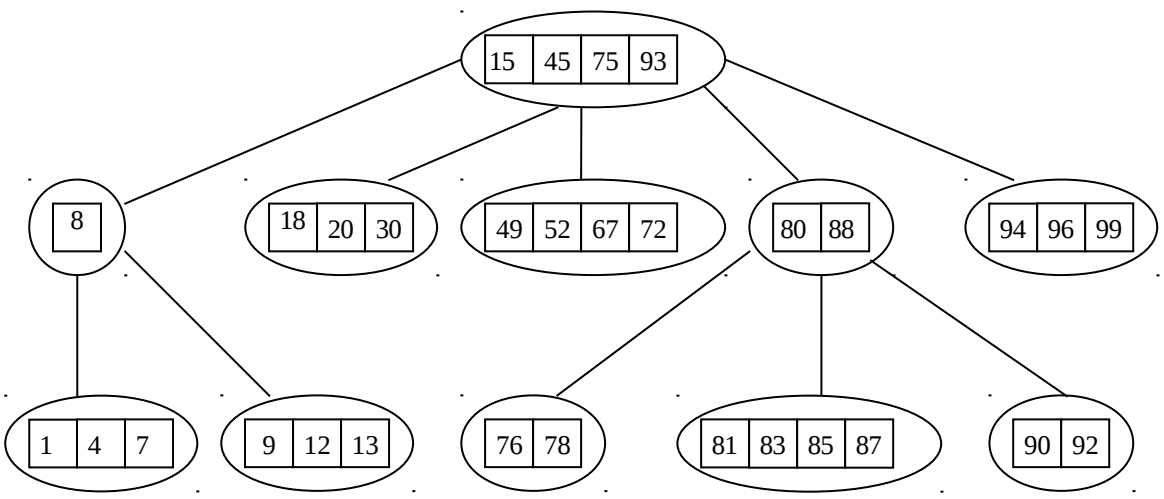
Nota: para calcular el valor de un número, dado, se puede usar el siguiente método (no hay que implementarlo):

```
public Double parse(String s)
// Post: dado un string que contiene un número entero, devuelve ese entero.
```

3. Búsqueda en árbol (1,5 puntos)

Tenemos un árbol N-ario de búsqueda, tal y como se muestra en la figura. Las restricciones son:

- Todos los valores del árbol son diferentes
- Cada nodo tiene N valores como máximo, ordenados ascendentemente
- Si un nodo tiene N valores a_1, a_2, … a_n, entonces ese nodo tiene N+1 hijos, de tal manera que el hijo más a la izquierda contiene los valores menores que a_1, el segundo subárbol contiene los valores entre a_1 y a_2, … y el último subárbol contiene los valores mayores que a_n.



Las declaraciones de datos son:

```
public class BinaryTreeNode<T> {
   T[] valores;
   BinaryTreeNode<T>[] hijos;
         // el tamaño es de la tabla de valores + 1
}


public class Arbol {

    BinaryTreeNode<Integer> root;

    public boolean esta(Integer elem)
      // post: el resultado es true si "elem" está en el árbol
      //       y false si no

}
```

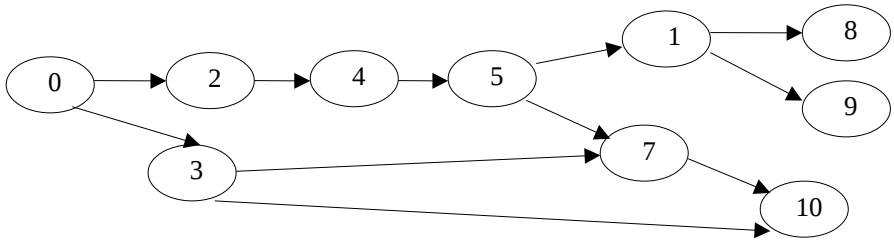
Se pide:

a) Implementar la función "esta"

b) Calcular, de manera razonada, el coste del algoritmo implementado.

4. Red de ordenadores (1,5 puntos)

Se tiene un grafo dirigido que representa una red de ordenadores, que presentamos parcialmente en la siguiente figura:

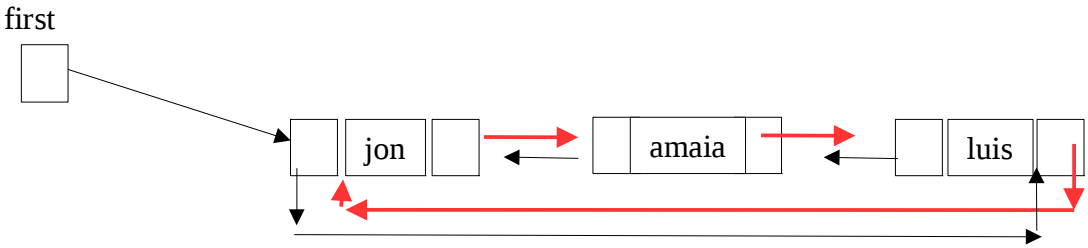


Cada ordenador X puede comunicarse con cada uno de sus vecinos Y al precio de 1 doblón por comunicación. A través de las conexiones de Y y subsiguientes, X puede comunicarse con el resto de ordenadores de la red pagando a cada ordenador que utilice para la conexión el doblón correspondiente.
Escriba un algoritmo que calcule la tabla precio(1..n) tal que precio(i) sea el número mínimo de doblones que le cuesta al ordenador numerado con el 0 establecer conexión con el ordenador numerado con el i.

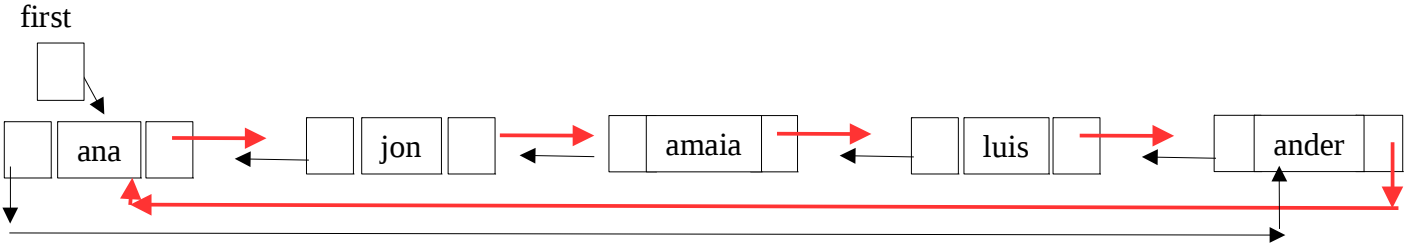
```
public class RedDeOrdenadores {

   protected Boolean[][] adjMatrix;   // adjacency matrix

   public int[] obtenerCostes() // Método a desarrollar

}
```

1. Ezabatu azpilista.(1,5 puntu)
Estekadura bikoitzeko bi zerrenda zirkular ditugu:

l1



l2

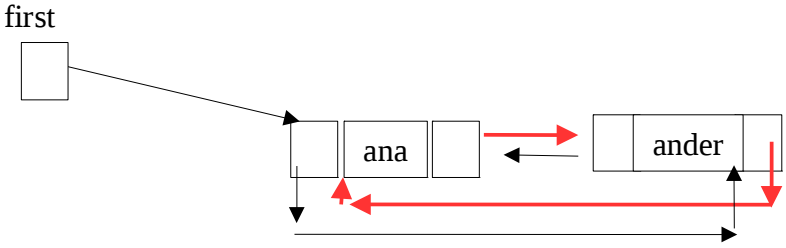


Metodo hau inplementatu nahi dugu:

```
public class DoubleNode<T> {
   T data;
   DoubleNode<T> next;
   DoubleNode<T> prev;
}

public class DoubleLinkedList<T> {
   DoubleNode<T> first;

   public void ezabatuLista(DoubleLinkedList<T> azpilista)
   // Aurrebaldintza: “azpilista” uneko listaren zati bat da
   //              “azpilista” zerrendako elementuak ordena berean daude lista nagusian
   // Postbaldintza: “azpilista” osoa kendu egin da zerrendatik


}
```

Adibidez, l2.ezabatuLista(l1) deiak zerrenda hau utziko luke:

l2



Hau eskatzen da:

- Algoritmoa inplementatu
- Kostua kalkulatu modu arrazoituan

2. Adierazpen aritmetikoaren ebaluazioa (1,5 puntu)

Djikstrak adierazpen aritmetiko bat ebaluatzeko algoritmo bat eman zuen. Adibidez: `(1+((2+3)*(4*5)))`

Metodo hau inplementatu behar da:

```
public double ebaluatu(String esp)
// Aurrebaldintza: Espresio aritmetikoa zuzena da.
//                 Adierazpena GUZTIZ parentizatuta dago, hau da,
//                 Eragile bakoitzeko, parentesiak daude
//                 Eragile guztiak bitarrak dira ("X ERAGILE Y" motakoak)
// Postbaldintza: emaitza adierazpenaren balioa da
```

Aurreko adierazpen hori emanda, emaitza 101 izango da.

Adierazpen aritmetiko parentizatua era honetan ebaluatuko da:

- Bi pila behar dira, bata eragigaien pila, bestea eragileena
- Sarrerako katearen karaktereak banan-banan aztertuko dira, eta hau egin behar da pauso bakoitzean:
  - Ezkerreko parentesia bada, ezer ez
  - Eragigaia bada, orduan eragigaien pilan sartu
  - Eragilea baldin bada, orduan eragileen pilan sartu
  - Eskuineko parentesia baldin bada, orduan:
    - hartu bi eragigai eragigaien pilatik
    - hartu eragilea eragileen pilatik
    - kalkulatu emaitza eta eragileen pilan sartu

Adibidea

Kate hau emanda: `(1+((2+3)*(4*5)))`

Eragileen pila	
Eragigaien pila	

- Lehen parentesia "(" → ezer ez

`Katea: 1+((2+3)*(4*5)))`

- "1" → sartu eragigaien pilan

Eragileen pila	
Eragigaien pila	1

`Katea: +((2+3)*(4*5)))`

- "+" → sartu eragileen pilan

Eragileen pila	+
Eragigaien pila	1

`Katea: ((2+3)*(4*5)))`

- parentesia “(“ → ezer ez
- parentesia “(“ → ezer ez

`Katea: 2+3)*(4*5)))`

- “2” → sartu eragigaien pilan

Eragileen pila	+
Eragigaien pila	1 2

`Katea: +3)*(4*5)))`

- “+” → sartu eragileen pilan

Eragileen pila	+ +
Eragigaien pila	1 2

`Katea: 3)*(4*5)))`

- “3” → sartu eragigaien pilan

Eragileen pila	+ +
Eragigaien pila	1 2 3

`Katea: )*(4*5)))`

- “)” → eragiketa egin

Eragileen pila	+
Eragigaien pila	1 5

`Katea: *(4*5)))`

- “*” → sartu eragileen pilan

Eragileen pila	+ *
Eragigaien pila	1 5

`Katea: (4*5)))`

- parentesia “(“ → ezer ez

`Katea: 4*5)))`

- “4” → sartu eragigaien pilan

Eragileen pila	+ *
Eragigaien pila	1 5 4

`Katea: *5)))`

- “*” → sartu eragileen pilan

Eragileen pila	+ * *
Eragigaien pila	1 5 4

Katea: 5)))

- “5” → sartu eragigaien pilan

Eragileen pila	+ * *
Eragigaien pila	1 5 4 5

Katea:)))

- “)” → eragiketa egin

Eragileen pila	+ *
Eragigaien pila	1 5 20

Katea:))

- “)” → eragiketa egin

Eragileen pila	+
Eragigaien pila	1 100

Katea:)

- “)” → eragiketa egin

Eragileen pila	
Eragigaien pila	101

Emaitza eragigaien pilan dago!

Algoritmoa inplementatu, eta bere kostua kalkulatu, modu arrazoituan.

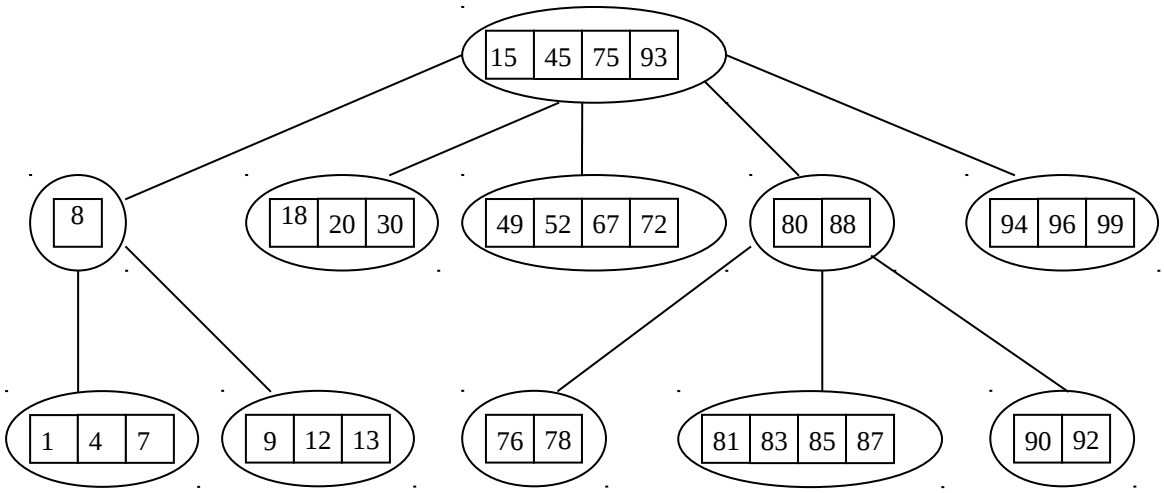
Oharra: zenbaki baten balioa kalkulatzeko, ondoko metodoa erabil daiteke (ez da inplementatu behar):

```
public Double parse(String s)
// Post: zenbaki bat daukan string bat emanda, balioa bueltatzen du.
```

3. Bilaketa zuhaitz n-tarrean (1,5 puntu)

Bilaketa-zuhaitz N-tarra dugu, irudian ikusten den moduan. Hauek dira murrizpenak:

- Balio guztiak desberdinak dira
- Adabegi bakoitzak N balio ditu gehienez, goranzko ordenan
- Adabegi batek N balio baldin baditu, a_1, a_2, … a_n, orduan adabegi horrek N+1 ume izango ditu. Ezkerreko umeak a_1 baino txikiagoak diren balioak izango ditu; bigarren azpizuhaitzak a_1 eta a_2 bitarteko balioak izango ditu, … eta azken azpizuhaitzak a_n baino handiagoak diren balioak izango ditu.



Hauek dira datu-erazagupenak:

```
public class BinaryTreeNode<T> {
   T[] balioak;
   BinaryTreeNode<T>[] umeak; // Balioen taulako tamaina + 1
}


public class Zuhaitza {

    BinaryTreeNode<Integer> root;

    public boolean badago(Integer elem)
      // post: emaitza true da "elem" zuhaitzean baldin badago eta
      //       false bestela
}
```

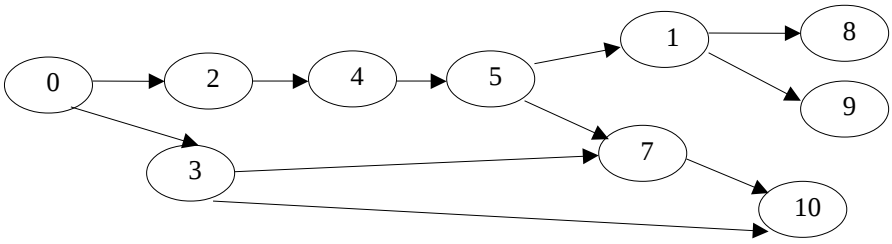
Hau eskatzen da:

a) *badago* funtzioa inplementatu

b) Kalkulatu, modu arrazoituan, algoritmoaren kostua.

4. Ordenagailuen sarea (1,5 puntu)

Eman dezagun ordenagailu-sare bat dugula. X ordenagailu bakoitza bere auzoko Y ordenagailu bakoitzarekin komunika daiteke euro 1eko prezioan.



Idatzi algoritmo bat, zeinak prezioa(1..n) taula kalkulatuko duen:
prezioa(i) = 0 ordenagailuak i ordenagailuarekin konektatzeko ordaindu beharreko prezio minimoa izanik.

```
public class OrdenagailuenSarea {


   protected Boolean[][] adjMatrix;   // adjacency matrix

   public int[] kostuakLortu() // Egin behar den metodoa
}
```