Estructuras de datos y Algoritmos
Lab4: Árboles binarios

Objetivo:
- Usar y entender la utilización de la estructura de datos Árbol binario (BinTree),
- Practicar el diseño de algoritmos recursivos para la solución de operaciones sobre árboles binarios

Ejercicio I

Escribe en Java un método que crea un nuevo árbol binario (p.ej. de números enteros del ejercicio IV) empezando desde cero.

Ejercicio II

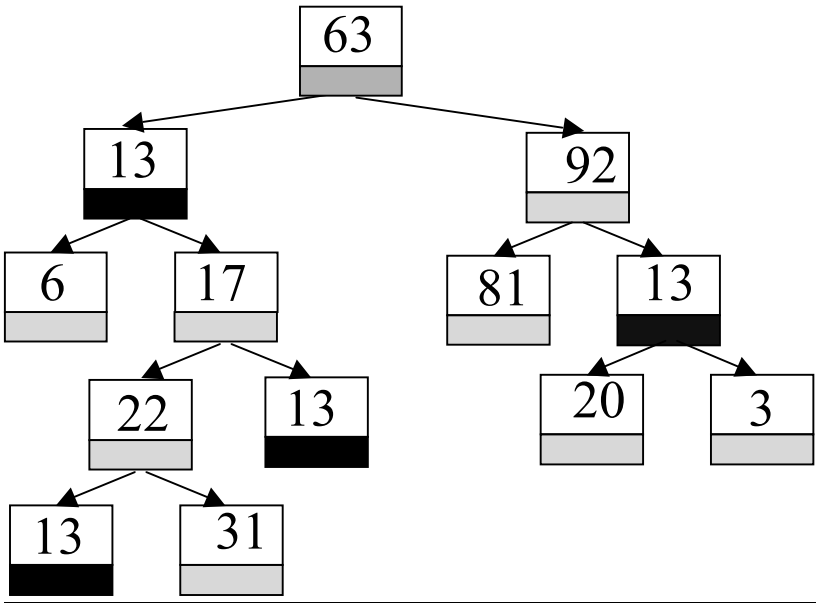
Diseñar y escribir en Java la función *altura* tal que dado un árbol binario calcule y devuelva la altura del árbol. (para el ej. del ejercicio IV devuelve 4)

Ejercicio III

Diseñar y escribir en Java la función tamaño tal que dado un árbol binario calcule y devuelva el tamaño del árbol. El tamaño del árbol es el número de descendientes del nodo raíz. (para el ej. del ejercicio IV devuelve 13)

Ejercicio IV

Diseñar y escribir en Java la función *count* tal que dado un árbol binario y un número, devuelva como resultado las apariciones del número x en el árbol.



P.ej: binTreeItr.count(new Integer(13)) devuelve 4

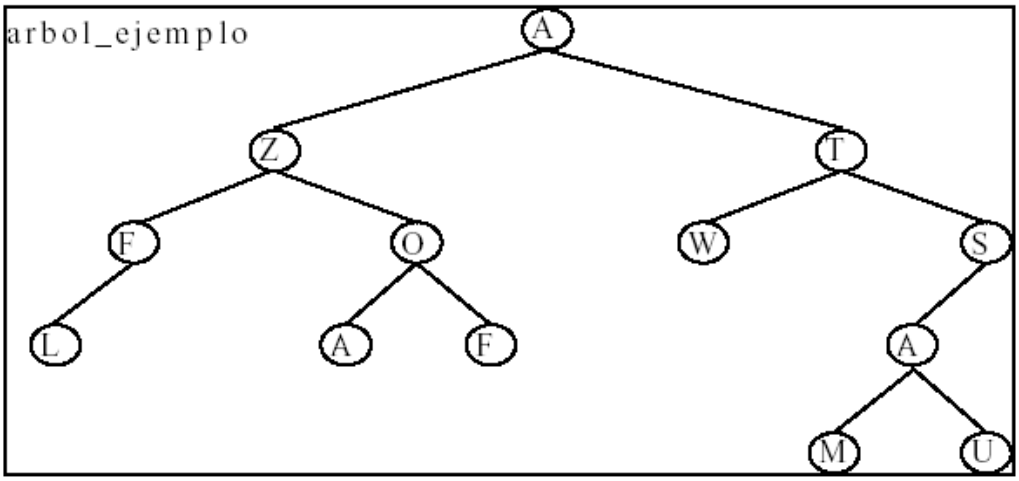
Ejercicio V

Diseñar y escribir en Java la función *profundidad* tal que dado un árbol binario calcule y devuelva la profundidad del árbol. La profundidad del árbol es la **profundidad máxima** alcanzada en uno de sus nodos. (para el ej. del IV devuelve 4)

Ejercicio VI

Especificar, diseñar y escribir en Java la función *numNodosIzquierdos* tal que dado un árbol binario, devuelva el número de nodos izquierdos que se encuentran en el árbol. En la figura, por ejemplo:

• *numNodosIzquierdos* devolvería 7



Ejercicio VII

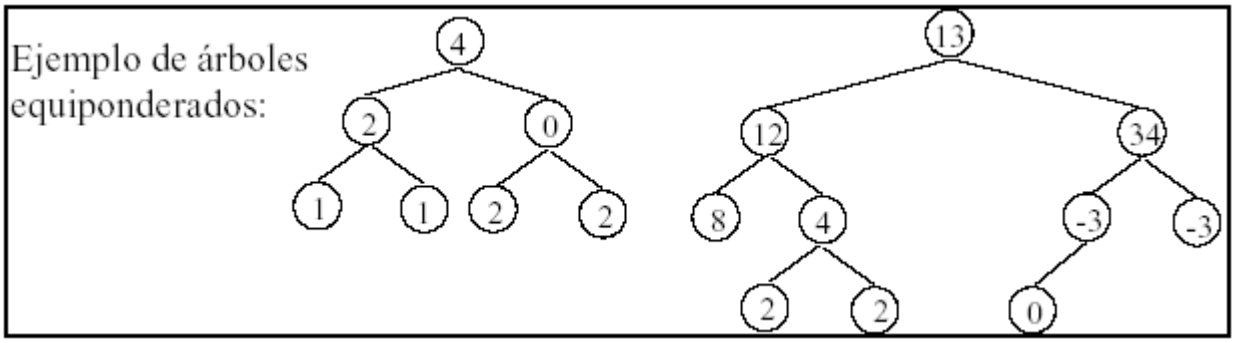
Especificar, diseñar y escribir en Java la función *longitudCaminoConMásNodosIzquierdos* tal que dado un árbol binario, devuelva la longitud del camino más largo que más número de nodos izquierdos se encuentran en él.

Ejercicio VIII

(J95) Especificar, diseñar y escribir en Java un subprograma tal que dado un árbol binario de enteros (Integer), determine si dicho árbol está *equiponderado*.

- Un árbol binario está *equiponderado* si se cumple que para todo nodo el *peso* del subárbol izquierdo es igual al *peso* del subárbol derecho.
- El árbol binario vacío está equiponderado por definición.

Llamamos *peso* de un árbol a la suma de todos sus elementos.

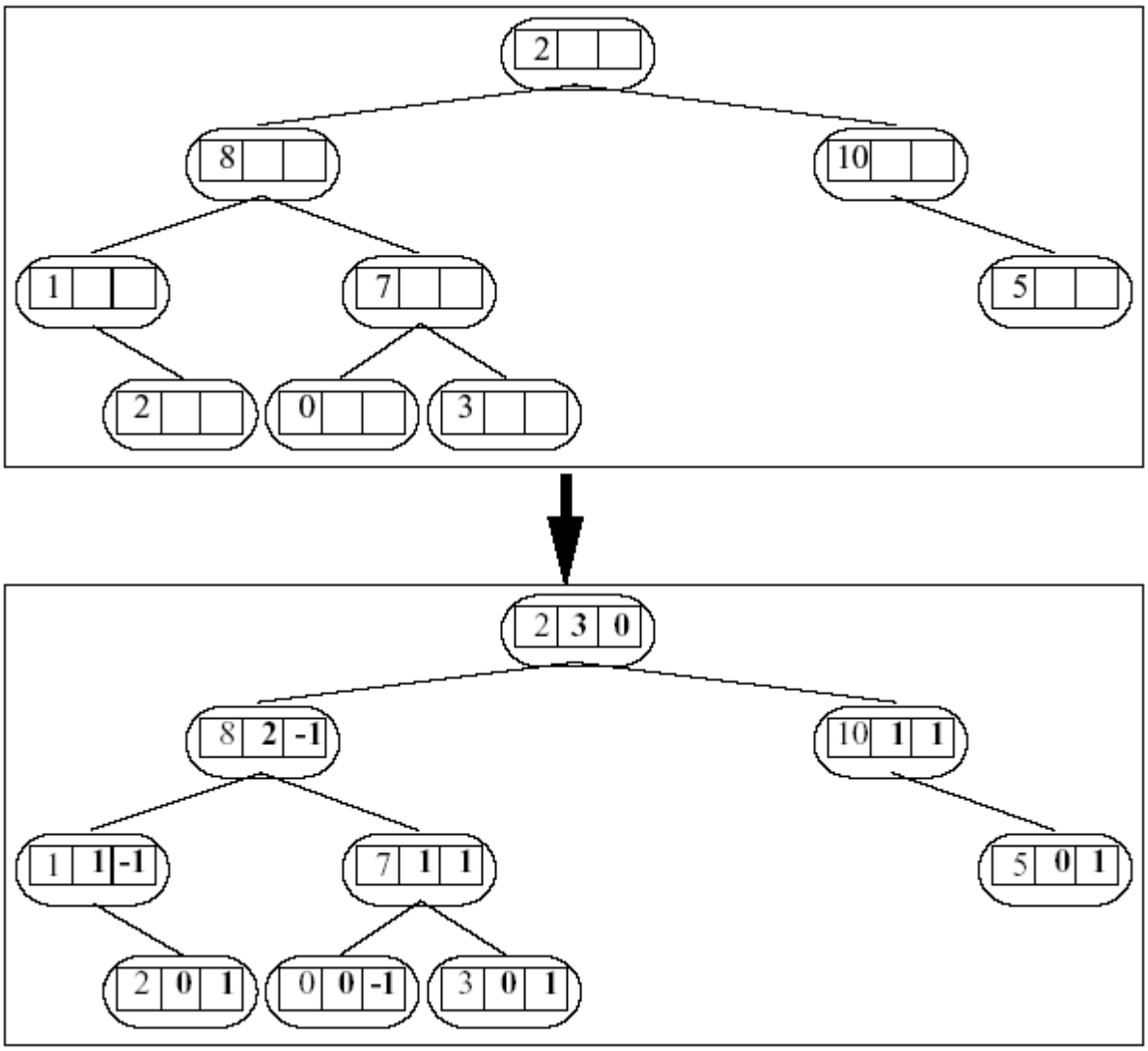


Ejercicio IX

(J96)

```
public class Nodo extends Object {
int dato;
int profundidad;
int claseDeHijo;
}
```

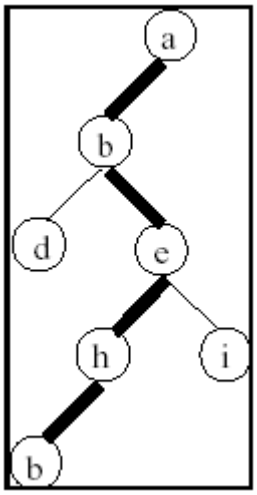
Diseñar y escribir en Java un subprograma tal que dado un árbol binario de elementos de tipo Nodo, devuelva el mismo árbol pero etiquetado. Esto es, que cada nodo del árbol contenga en su subcampo profundidad la información correspondiente a la profundidad del subárbol que comienza en ese nodo, y en el subcampo claseDeHijo el valor 0 si se trata del nodo raíz del árbol, -1 si es raíz de un subárbol izquierdo y 1 si es raíz de un subárbol derecho.



Ejercicio X

(S95) Diseñar y escribir en Java un subprograma tal que dado un árbol binario y una lista doblemente ligada, determine si la lista **coincide exactamente** con alguna ***rama completa*** del árbol.

Una *rama del árbol* es *completa* si partiendo de la raíz acaba en alguna de las hojas.



Ejemplo de lista ligada que SI coincide exactamente con alguna de las ramas completas del árbol de la figura:

Ejemplos de listas que NO coinciden exactamente con ninguna de las ramas completas del árbol anterior: