ENTREGA E4 (0,5 puntos)

Antes de empezar...

- Descargar el archivo **E4.zip** e importarlo a Eclipse.
- Renombrar el proyecto (tecla F2). Por ejemplo, para el grupo formado por Ana Pérez, Jon Azkue y Miren Landa el nuevo nombre sería: **E4_APerezJAzkueMLanda**.
- En el fichero **componentesGrupo**, escribir los nombres de los componentes del grupo.

Además, tened en cuenta que:

- Se valorará la eficiencia de las soluciones, es decir, además de que sean correctas, deben ser eficientes.
- Aparte de los casos de prueba que se os entregan, se espera que incorporéis casos de prueba adicionales y significativos.
- Para entregarlo, debéis exportar el proyecto y subirlo a eGela.

Fecha límite de entrega: 05/11/2023 a las 23:59

—

Ayuda para importar:

File -> Import... -> General -> Existing Projects into Workspace -> Select archive file (el .zip descargado) -> Finish

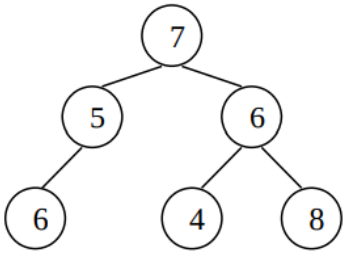
Ayuda para exportar:

Pinchar en el proyecto. File -> Export... -> General -> Archive File -> (seleccionar las carpetas/archivos a exportar) -> To archive file (escribir una ruta y nombre para el nuevo archivo .zip) -> Finish

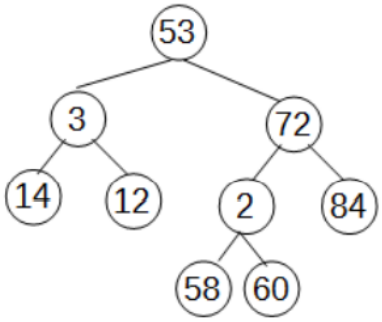
NOTA: A CONTINUACIÓN DEL ENUNCIADO PODÉIS ENCONTRAR LOS DIBUJOS DE LOS ÁRBOLES UTILIZADOS EN LOS CASOS DE PRUEBA QUE SE OS ENTREGAN, ASÍ COMO LOS RESULTADOS DE EJECUCIÓN ESPERADOS PARA DICHOS CASOS DE PRUEBA

Ejercicio 1

Implementar el método **esLLeno()**. Este método devuelve un booleano que indica si el árbol es lleno. Un árbol es lleno si todos los nodos no-hoja del árbol tienen exactamente dos hijos. Supondremos que el árbol vacío es lleno. Ejemplos:



esLleno():false



esLLeno(): true

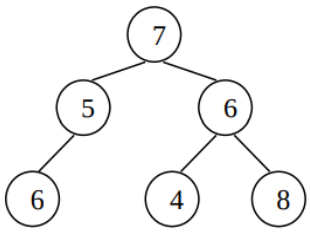
Ejercicio 2

Implementar el método **listaDeMenores(T elem)**. Este método recibe un elemento *elem* y debe devolver un LinkedList con aquellos elementos que sean menores que *elem*.

Notas:

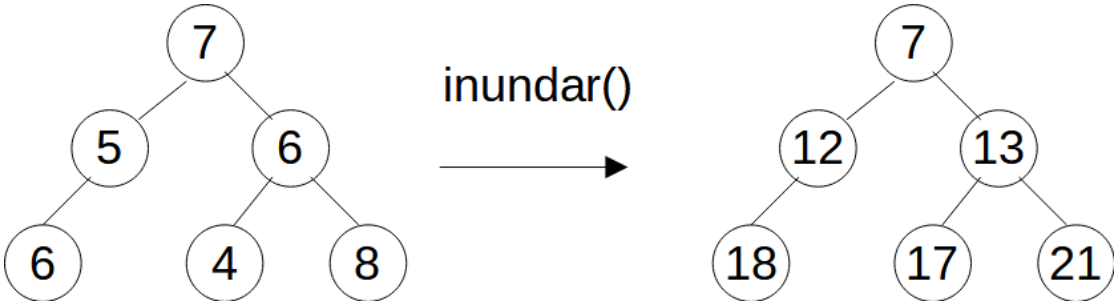
- Los elementos de la lista resultante pueden estar en cualquier orden
- No se puntuarán soluciones en las que primero se obtenga la lista de todos los elementos y luego se filtren los menores que *elem*.

Ejemplo: Para el siguiente árbol, la llamada **listaDeMenores(6)** devolvería **[5, 4]** (en cualquier orden)



Ejercicio 3

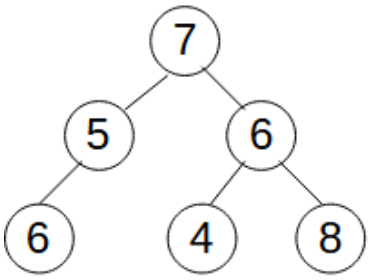
Implementar el método **inundar().** Este método actualiza el valor de cada nodo sumándole el valor original de todos sus ascendientes. Ejemplo:



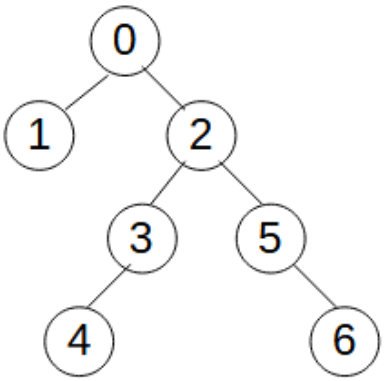
En el ejemplo, el nodo que originalmente tenía el valor 5 pasa a tener el valor 12 porque se le ha sumado el valor de su único ascendiente (la raíz). El nodo que originalmente tenía el valor 8 pasa a tener el valor 21, porque se le han sumado los valores originales de sus dos ascendientes (7 y 6). El valor final del resto de nodos se ha calculado de la misma manera.

ÁRBOLES DE LOS CASOS DE PRUEBA

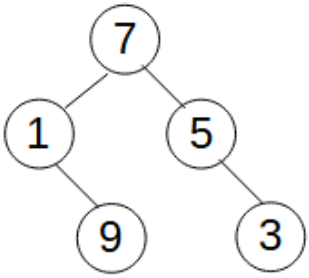
arbol0.txt



arbol1.txt



arbol2.txt

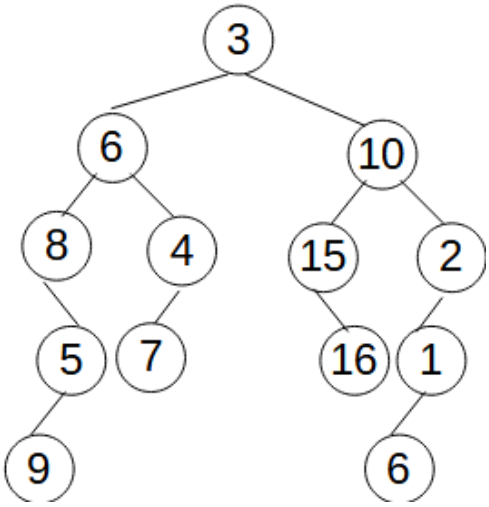


arbol3.txt

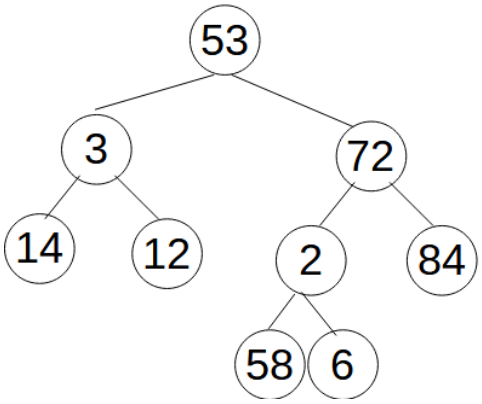
(vacío)

arbol4.txt

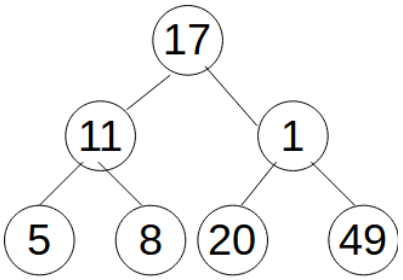
arbol5.txt



arbol6.txt



arbol7.txt



RESULTADOS DE LA EJECUCIÓN DE LOS CASOS DE PRUEBA PROPORCIONADOS

EJERCICIO 1

```
-------------
 ÁRBOL 0
-------------
esLLeno? false

-------------
 ÁRBOL 1
-------------
esLLeno? false

-------------
 ÁRBOL 2
-------------
esLLeno? false

-------------
 ÁRBOL 3
-------------
esLLeno? true

-------------
 ÁRBOL 4
-------------
esLLeno? true

-------------
 ÁRBOL 5
-------------
esLLeno? false

-------------
 ÁRBOL 6
-------------
esLLeno? true

-------------
 ÁRBOL 7
-------------
esLLeno? true
```

EJERCICIO 2 (LOS RESULTADOS SON VÁLIDOS EN CUALQUIER ORDEN)

```
-------------
 ÁRBOL 0
-------------
Árbol: [ 7 [ 5 [ 6 ] * ] [ 6 [ 4 ] [ 8 ] ] ]
Menores que 6: [5, 4]
Menores que 0: []
Menores que 40: [6, 5, 7, 4, 6, 8]

-------------
 ÁRBOL 1
-------------
Árbol: [ 0 [ 1 ] [ 2 [ 3 [ 4 ] * ] [ 5 * [ 6 ] ] ] ]
Menores que 6: [1, 0, 4, 3, 2, 5]
Menores que 0: []
Menores que 40: [1, 0, 4, 3, 2, 5, 6]

-------------
 ÁRBOL 2
-------------
Árbol: [ 7 [ 1 * [ 9 ] ] [ 5 * [ 3 ] ] ]
Menores que 6: [1, 5, 3]
Menores que 0: []
Menores que 40: [1, 9, 7, 5, 3]

-------------
 ÁRBOL 3
-------------
Árbol: *
Menores que 6: []
Menores que 0: []
Menores que 40: []

-------------
 ÁRBOL 4
-------------
Árbol: [ 7 ]
Menores que 6: []
Menores que 0: []
Menores que 40: [7]

-------------
 ÁRBOL 5
-------------
Árbol: [ 3 [ 6 [ 8 * [ 5 [ 9 ] * ] ] [ 4 [ 7 ] * ] ] [ 10 [ 15 * [ 16 ] ] [ 2
[ 1 [ 6 ] * ] * ] ] ]
```

```
Menores que 6: [5, 4, 3, 1, 2]
Menores que 0: []
Menores que 40: [8, 9, 5, 6, 7, 4, 3, 15, 16, 10, 6, 1, 2]

-------------
 ÁRBOL 6
-------------
Árbol: [ 53 [ 3 [ 14 ] [ 12 ] ] [ 72 [ 2 [ 58 ] [ 6 ] ] [ 84 ] ] ]
Menores que 6: [3, 2]
Menores que 0: []
Menores que 40: [14, 3, 12, 2, 6]

-------------
 ÁRBOL 7
-------------
Árbol: [ 17 [ 11 [ 5 ] [ 8 ] ] [ 1 [ 20 ] [ 49 ] ] ]
Menores que 6: [5, 1]
Menores que 0: []
Menores que 40: [5, 11, 8, 17, 20, 1]
```

EJERCICIO 3

```
-------------
 ÁRBOL 0
-------------
Árbol antes de inundar:
[ 7 [ 5 [ 6 ] * ] [ 6 [ 4 ] [ 8 ] ] ]
Árbol después de inundar:
[ 7 [ 12 [ 18 ] * ] [ 13 [ 17 ] [ 21 ] ] ]

-------------
 ÁRBOL 1
-------------
Árbol antes de inundar:
[ 0 [ 1 ] [ 2 [ 3 [ 4 ] * ] [ 5 * [ 6 ] ] ] ]
Árbol después de inundar:
[ 0 [ 1 ] [ 2 [ 5 [ 9 ] * ] [ 7 * [ 13 ] ] ] ]

-------------
 ÁRBOL 2
-------------
Árbol antes de inundar:
[ 7 [ 1 * [ 9 ] ] [ 5 * [ 3 ] ] ]
Árbol después de inundar:
[ 7 [ 8 * [ 17 ] ] [ 12 * [ 15 ] ] ]

-------------
 ÁRBOL 3
-------------
Árbol antes de inundar:
*
Árbol después de inundar:
*

-------------
 ÁRBOL 4
-------------
Árbol antes de inundar:
[ 7 ]
Árbol después de inundar:
[ 7 ]

-------------
 ÁRBOL 5
-------------
Árbol antes de inundar:
```

[3 [6 [8 * [5 [9] *]] [4 [7] *]] [10 [15 * [16]] [2 [1 [6] *] *]]]
Árbol después de inundar:
[3 [9 [17 * [22 [31] *]] [13 [20] *]] [13 [28 * [44]] [15 [16 [22] *] *]]]

 ÁRBOL 6

Árbol antes de inundar:
[53 [3 [14] [12]] [72 [2 [58] [6]] [84]]]
Árbol después de inundar:
[53 [56 [70] [68]] [125 [127 [185] [133]] [209]]]

 ÁRBOL 7

Árbol antes de inundar:
[17 [11 [5] [8]] [1 [20] [49]]]
Árbol después de inundar:
[17 [28 [33] [36]] [18 [38] [67]]]