Práctica Individual 2

1. Dado un array bidimensional de n x n números enteros (siendo n una potencia de 2), decidir si cumple que el valor en la casilla superior izquierda es menor al de la casilla inferior derecha y cada uno de los cuatro subarrays cumplen la propiedad.

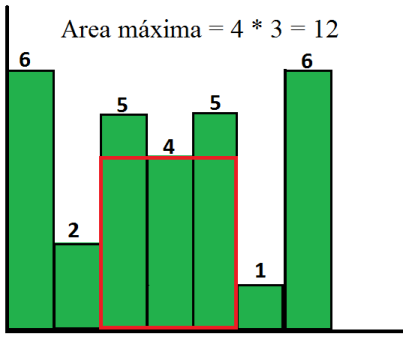
2. Dado un array ordenado de números enteros y rotado k veces, diseñar un algoritmo que encuentre el elemento mayor del array, suponiendo que no conocemos el valor de k.

3. Dada una lista ordenada de tipo genérico (List<E> lista), diseñar un algoritmo que devuelva un conjunto que incluya los elementos de dicha lista que se encuentren en un rango [a, b) dado (siendo a y b del mismo tipo que los elementos de la lista). Tenga en cuenta que el método que desarrolle debe incluir un comparador sobre E como parámetro de entrada.

4. Diseñar un algoritmo para el cálculo de un número combinatorio a partir de la siguiente definición:

$$\binom{n}{k} = \begin{cases} 1, & si\ k = 0 \vee k = n \\ n, & si\ k = 1 \vee k = n-1 \\ \binom{n-1}{k-1} + \binom{n-1}{k} & en\ otro\ caso \end{cases}$$

5. Dado un array de números enteros que representa un histograma, diseñe un algoritmo que determine el mayor área rectangular que se puede encontrar. Por simplicidad considere que todas las barras tienen anchura de 1 unidad.



6. Dado un árbol BinaryTree<E> y una función que convierte E en R, diseñe un algoritmo que lo transforme en otro BinaryTree<R>. Realice lo mismo para un árbol n-ario. **(*)**

7. Diseñe un algoritmo que dado un árbol n-ario de tipo genérico devuelva una lista que contenga las etiquetas de las hojas de dicho árbol.

8. Dado un árbol binario ordenado de números enteros, diseñe un algoritmo que devuelva el nivel en el que se encuentra un elemento dado, o -1 en caso de que el elemento no se encuentre en el árbol.

9. Dado un árbol n-ario de números enteros, diseñe un algoritmo que devuelva una tupla de 2 elementos que se correspondan con el elemento menor y mayor del árbol.

10. Dado un árbol binario ordenado de cadenas de caracteres (BinaryTree<string>), diseñe un algoritmo que devuelva un conjunto con todos los elementos mayores o iguales que uno dado.

(*) Se recomienda hacer uso de la solución dada al ejercicio 6 para facilitar la lectura desde archivo en algunos de los ejercicios posteriores.

Tenga en cuenta que:

- **Para cada ejercicio debe leer los datos de entrada de un fichero, y mostrar la salida por pantalla. Dicha lectura debe ser independiente del algoritmo concreto que resuelva el ejercicio.**
- **La solución que se le debe dar a cada ejercicio tiene que ser acorde al material de la asignatura proporcionado.**

Entrega PI2A – Se pide resolver de forma <u>eficiente</u>:

- **Ejercicio 1: proporcione una solución recursiva tanto en C como en Java.**
- **Ejercicio 2: proporcione una solución recursiva en C.**
- **Ejercicio 3: proporcione una solución recursiva en Java.**
- **Ejercicio 4: proporcione una solución recursiva sin memoria, otra recursiva con memoria, y otra iterativa (en C).**
- **Ejercicio 5: proporcione una solución recursiva en Java.**

Entrega PI2B – Se pide resolver de forma <u>eficiente</u>:

- **Ejercicio 6: proporcione una solución recursiva tanto en C como en Java para árboles binarios, y una solución en Java para árboles n-arios.**
- **Ejercicios 7 y 9: proporcione una solución recursiva en Java.**
- **Ejercicios 8 y 10: proporcione una solución recursiva en C.**

Cada una de las entregas debe incluir:

- **Proyecto en eclipse con las soluciones en C.**
- **Proyecto en eclipse con las soluciones en Java.**
- **Memoria de la práctica, que debe contener:**
    - **Código realizado**
    - **Volcado de pantalla con los resultados obtenidos para las pruebas realizadas, incluyendo al menos los resultados obtenidos para los tests proporcionados.**