Relación de ejercicios sobre árboles y tablas hash

1. Escribe una función que determine cual es la hoja más profunda de un árbol binario.
2. Escribe una función que acepte un valor de tipo *node* y un árbol general *T* y devuelva el nivel del nodo en el árbol.
3. Construye una función que determine si un árbol binario es completo.
4. El recorrido en preorden de un determinado árbol binario es:

   GEAIBMCLDFKJH

   y en inorden

   IABEGLDCFMKHJ

   Resolver:

   a) Dibuja el árbol binario.

   b) Da el recorrido en postorden.

   c) Diseña una función para dar el recorrido en postorden dados los recorridos en preorden e inorden (sin construir el árbol) y escribe un programa para comprobar el resultado del apartado anterior.
5. Escribe una función que realice la reflexión de un árbol binario. Es decir, una función a la que se le pase un árbol y se modifique de forma que para cada nodo su hijo a la izquierda pase a ser el derecho y viceversa.
6. Definimos densidad de un árbol binario A como la suma de las profundidades de las hojas de un árbol. Construir un algoritmo para calcular la densidad de un árbol binario.
7. Supongamos que tenemos una función *valor* tal que dado un valor de tipo *char* (una letra del alfabeto) devuelve un valor entero asociado a dicho identificador. Supongamos también la existencia de un árbol de expresión *T* cuyos nodos hoja son letras del alfabeto y cuyos nodos interiores son los caracteres *, +, -, /. Diseñar una función que tome como parámetros un nodo y un árbol binario y devuelva el resultado de la evaluación de la expresión representada.
8. Implementa una función que dada una expresión en postfijo devuelva el árbol binario asociado. Los operandos serán letras y los operadores +,-,*,/
9. Escribe una función no recursiva (usando una pila) para calcular la altura de un árbol binario.
10. Dos árboles binarios son similares si son iguales en cuanto a su estructura, es decir, cada nodo en un árbol tiene los mismos hijos y en el mismo lugar que el correspondiente en el otro árbol (sin importar las etiquetas). Escribe una función que dados dos árboles binarios devuelva si son o no similares.
11. Dado un bintree T, organizado como un BST, implementa una función a la que se le pasen dos valores a y b y que determine de forma eficiente los valores presentes en el árbol y que estén comprendidos entre ambos. Tanto a como b no tienen porqué aparecer en el árbol.
12. Construye un BST y un POT con las claves 50,25,75,10,40,60,90,35,45,70,42.

13. Indica la secuencia de rotaciones resultante de la inserción del conjunto de elementos {1, 2, 3, 4, 5, 6, 7, 15, 14, 13, 12, 11, 10, 9, 8} en un árbol AVL.

14. ¿Bajo qué condiciones puede un árbol ser parcialmente ordenado y binario de búsqueda simultáneamente? Razona la respuesta.

15. Implementa una función `bintree<int>::node ancestro_comun(bintree<int>::node n1,` que devuelva el nodo que sea el primer ancestro común a los nodos $n1$ y $n2$. La eficiencia debe ser proporcional a la altura del árbol.

16. Dada una secuencia de claves $S = \{5, 8, 4, 13, 66, 2, 9, 12, 11, 17\}$

    - Inserta la secuencia anterior, en el orden indicado, en una tabla hash cerrada con resolución de colisiones lineal y que tiene tamaño 11. A continuación, borrar los elementos 2 y 17.
    - Construye el árbol AVL que se obtiene al insertar los elementos de la secuencia S (en el orden en que aparecen).

17. Dada la representación `vector<list<T> > TH` de un tipo de dato `hash_table<T>`. Diseña una representación del iterador e implementa las operaciones:

    - `hash_table<T>::iterator::operator--(),tabla_hash<T>::begin()` y
    - `hash_table<T>::end()`.