Ejercicio 1.

Diseña un método que devuelva una *ListaConPI* con las entradas de un *MapOrdenado* ordenadas ascendentemente.

Ejercicio 2.

Diseña un método estático, genérico e iterativo *mapSort* que, con la ayuda de un *MapOrdenado*, ordene los elementos (*Comparable*) de un array *v*.

Ejercicio 3.

a) ¿Cuántas aristas tiene un árbol con *N* nodos?

b) ¿Longitud de A a D?

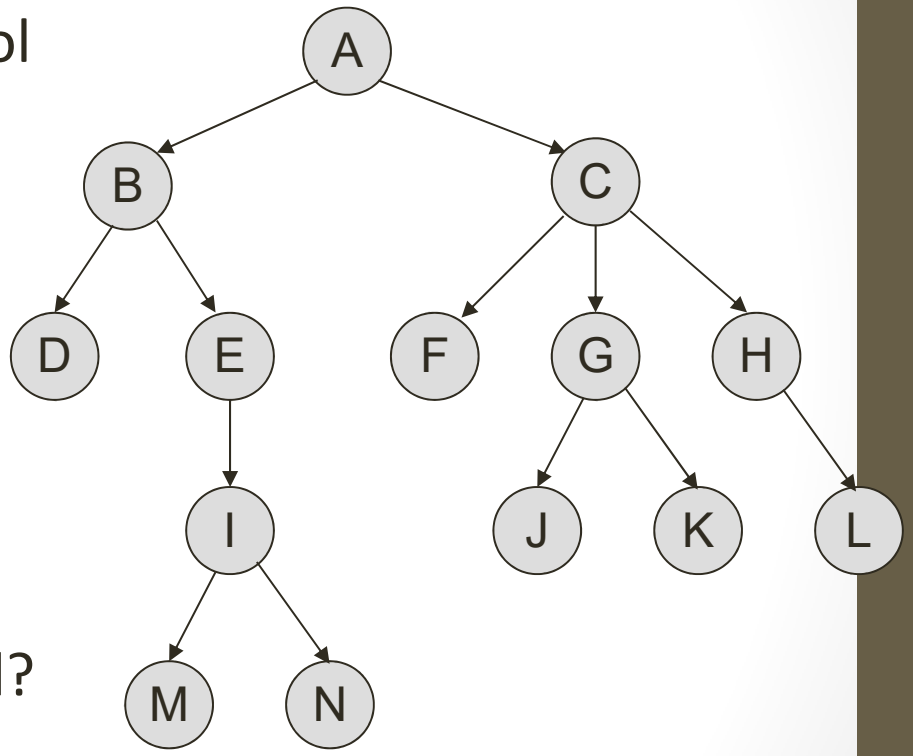
c) ¿Longitud de C a K?

d) ¿Longitud de B a N?

e) ¿Longitud de B a B?

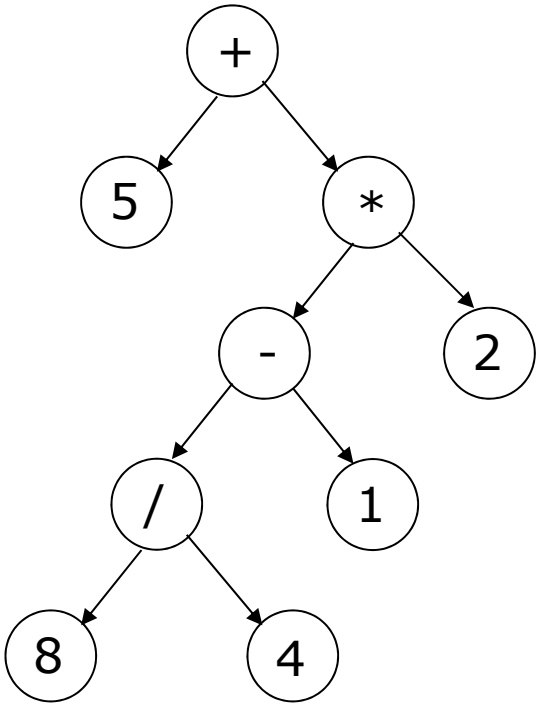
f) ¿Profundidad de A, B, C y F?

g) ¿Altura de B, C, I, F y del árbol?



Ejercicio 4.

Muestra el resultado de recorrer en pre-orden, in-orden, post-orden y por niveles el siguiente árbol:



Ejercicio 5.

Si se busca el número 363 en un *ABB* que contiene números del 1 al 1000 ¿Cuál de las siguientes secuencias de nodos no puede ser la secuencia de nodos examinada?

a) 2, 252, 401, 398, 330, 344, 397, 363
b) 924, 220, 911, 244, 898, 258, 362, 363
c) 925, 202, 911, 240, 912, 245, 363
d) 2, 399, 387, 219, 266, 382, 381, 278, 363
e) 935, 278, 347, 621, 299, 392, 358, 363

Ejercicio 6.

Diseñar un método que devuelva el dato que está en el nodo padre de un elemento dado. Indica el coste temporal del método.

Ejercicio 7.

Diseña un método que devuelva el nivel del nodo que contiene el dato *x* (se supone que no hay datos duplicados).

Ejercicio 8.

Diseña un nuevo constructor para la clase *ABB* que, partiendo de un *ABB* vacío, inserte los datos de un vector de forma que el *ABB* resultante quede equilibrado.

Ejercicio 9.

Diseñar los siguientes métodos en la clase *ABB:*

- Obtener el número total de hojas del árbol
- Visualizar los datos de los nodos del nivel *k* del árbol
- Calcular la altura del árbol

Ejercicio 10.

Diseña la clase *ABBInteger* como un *ABB* que trabaja con datos de tipo *Integer*, y añade los siguientes métodos:

- Un método que obtenga la suma de todos los elementos que sean mayores o iguales a un valor entero dado
- Un método que cambie el signo de todos los datos del árbol. El *ABB* debe seguir manteniendo la propiedad de orden

Ejercicio 11.

Diseñar un método en un ABB para eliminar todos los elementos menores que uno dado.

Ejercicio 12.

Diseña en la clase *ABB* un método para obtener el predecesor de un dato *x* dado

- El predecesor de un nodo es el máximo de su subárbol izquierdo (si tiene) o, en caso contrario, el ascendiente por la izquierda más cercano.

Ejercicio 13.

Diseña en la clase *ABB* un método que devuelva el número de elementos del árbol que están dentro de un intervalo dado [*x*, *y*]