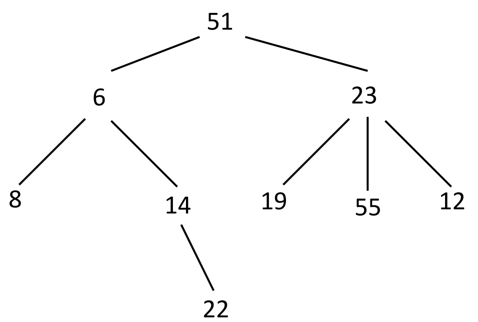
PRACTICAL TUTORIAL
WORKSHEET N°4: TREES

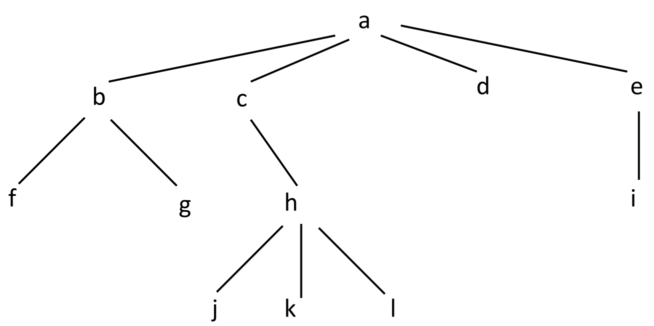
Exercise 1: Consider the sequence (7,9,5,15,13,24,11) representing the pre-order traversal of a binary tree, and the sequence (9,5,7,13,24,15,11) representing the in-order traversal of the same tree. Determine the corresponding tree and its post-order traversal.

Exercise 2: Write a recursive function **treeEquals(Node *t1, Node *t2)** that compare 2 binary trees of respective roots t1 and t2. The function return true if the 2 trees are equal and false otherwise.

Exercise 3: Knowing that the breadth-first traversal (by level) is not recursive on a binary tree, what data structure would be more suitable for the implementation of this traversal? Give the (iterative) algorithm that performs this traversal.

Exercise 4: Give the corresponding binary trees of the following trees:





Exercise 5: Write a routine that calculates the height of a (general) tree (consider dynamic representation of the tree).

Exercise 6: Suppose we have to search for element 363 in a binary search tree containing values between 1 and 1000. Which of these sequences cannot match the keys examined?

a) 2 252 401 398 330 363
b) 399 387 219 266 382 381 278 363
c) 4 924 278 347 621 299 392 358 363
d) 5 925 202 910 245 363

Exercise 7: Give the data structure resulting from the insertion of the following keys in order (6 11 26 28 2 3) in the case of:

a. BST
b. AVL
c. min-Heap
d. max-Heap

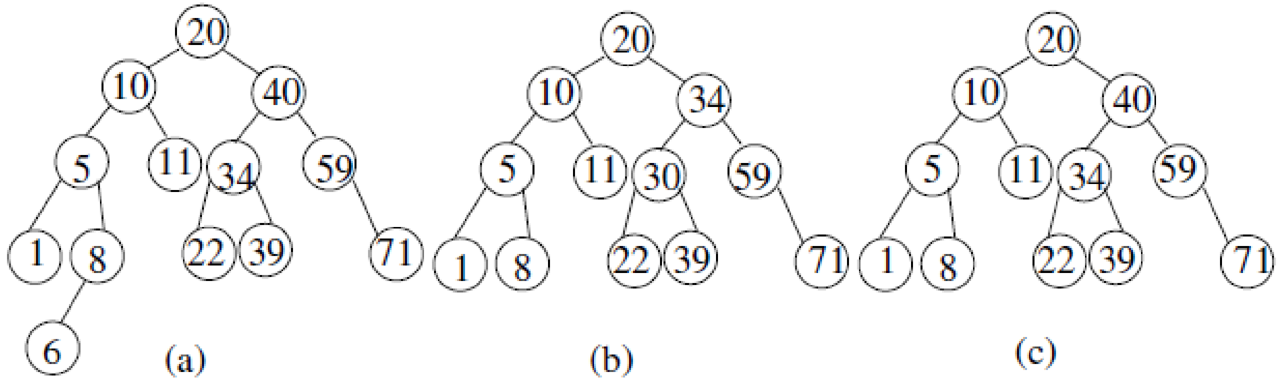
Exercise 8: We want to test whether a binary tree is a binary search tree (BST) or not. Write a function that return true if it is a BST and false otherwise. The function must be recursive and follow these conditions:

- If the height of a node is < 1 then this node is a BST.
- Else if a node has no left child then its value must be less than the minimum element of the right sub-tree and this right sub-tree is a BST.
- Else if a node has no right child then its value must be greater than the maximum element of the left sub-tree and this left sub-tree is a BST.
- Otherwise the considered node has two children; determine in this case the instructions to be executed and include them in your function.

We consider that we already have the three functions:

`searchMin(Node *)` *: which returns the minimum value of a binary tree.*
`searchMax(Node *)` *: which returns the maximum value of a binary tree.*
`height(Node *)` *: which returns the height of a binary tree.*

Exercise 9: Calculate the balance factor f for each node of the following binary search trees. Deduce if they are AVL. Otherwise, apply the necessary rotations.



Exercise 10: Propose an algorithm to test if a BST is an AVL (assume that the balancing factor is not stored in the nodes)

Exercise 11: Insert the integers (5 3 17 10 85 2 19 6 22 4) into a min-heap. From the resulting heap, show the result of removing the root.

Exercise 12: Given a heap, write an iterative procedure to display the ancestors of an element with index i.

Exercise 13: A d-heap is like a regular heap where each of its node has d children. Consider a d-heap implanted in an array A where the root is stored in A[1], its d children are stored in A[2],…, A[d+1], and so on. Say how to find the parent and the kth child of A[i].

Exercise 14: Apply heap sort method to the following list (explain with diagrams the different steps)

40	16	20	23	28