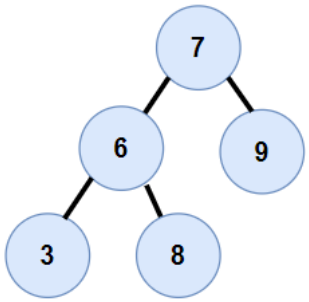
Introduction to Algorithms (CELEN086)

Problem Sheet 8

Topics: Binary search tree (BST); Traversal schemes; Recursive algorithms on BST

1. Write the leaf and node notation (pseudocode) for the binary tree:

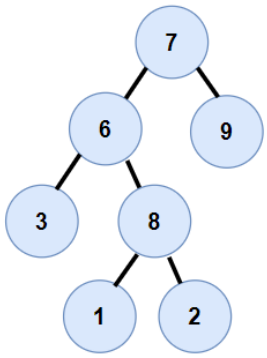


2. Explain why Q1 is not a binary search tree. Redraw the above tree to make it a binary search tree.

3. Write a recursive algorithm called **max(BST)** that finds the maximum value stored in a binary search tree. Considering the time complexity, is it $O(n)$ or $O(h)$?

4. Write a recursive algorithm called **insert(x, BST)** that inserts a number into the appropriate place of a binary search tree. (Assume that the number x is different from all current values stored in the tree).

5. Write the lists obtained from different traversal schemes on the binary tree:



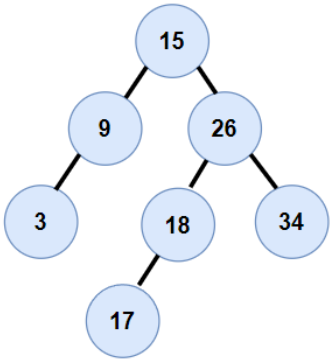
   i. Breadth first

   ii. Depth first, preorder

   iii. Depth first, inorder

iv. Depth first, postorder

6. Write the lists obtained from different traversal schemes on the binary search tree:



i. Breadth first

ii. Depth first, preorder

iii. Depth first, inorder

iv. Depth first, postorder

7. The BST in Q6 is not of minimum height/depth for storing all 7 values. Demonstrate the process of building a binary search tree from it with minimum height/depth.

8. Write a recursive algorithm called **isBST(T)** that determines if a binary tree is also a binary search tree or not.

9. Consider the algorithm search(x, T) in Lecture 7 slides. We aim to look for x in the binary tree by comparing the search key to each node values. Which traversal scheme is used there?

10. Write a recursive algorithm called **inorder(BST)** that returns a list of values stored in a binary search tree by the inorder traversal scheme (LNR scheme).

    Following algorithms might be helpful, and you can directly use any of them as sub-algorithm here:

    - merge(L1,L2)
    - mergeSort(L)
    - concat(L1,L2) (Problem Sheet 5, Q2)