# COL106 Lab Week 5 Questions

## Problem 1: Reconstructing a Binary Tree

As discussed in class: you are given two integer arrays *inorder* and *preorder*, representing the IDs of the nodes in the inorder and preorder.

From these traversals, try to reconstruct the binary tree.

You can run and test your code here.

## Problem 2: Lowest Common Ancestor

You are given two binary trees and references to two nodes *n1* and *n2*. Find the element which is the lowest common ancestor of the two nodes.

The lowest common ancestor in a binary tree T is defined between two nodes p and q as the *lowest node* in T that has both p and q as descendants (where we allow a node to be a descendant of itself).

You can run and test your code here.

## Problem 3: kth-Smallest Element of Binary Search Tree

Given the root of a binary search tree, and an integer k, return the kth smallest value (1-indexed) of all the values of the nodes in the tree.

You can run and test your code here.

## Challenge Problem: Speeding up LCA Queries

You have found the lowest common ancestor of two nodes in Problem 2. Think about the time complexity of your algorithm. Suppose we want to do better. For one-time uses, the solution of problem 2 seems alright. But suppose we want to make a large number of queries - i.e. find the LCA of a large no. of pairs (n1,n2).

In such scenarios, when we are designing a data structure for answering a large number of queries, some *preprocessing* helps - i.e. a one-time computation (which may be expensive) but makes answering queries much faster. *Preprocess* the tree such that you can answer LCA queries in O(*logn*) time, where n is the number of nodes in the tree.

*Hint:* Think about your algorithm for Problem 2. Would it be faster if you could skip over more than 1 element at a time? (i.e reach the grandparent, or the grandparent's grandparent quickly?)

If you implement this algorithm, you can run and test it by solving this problem based on it.