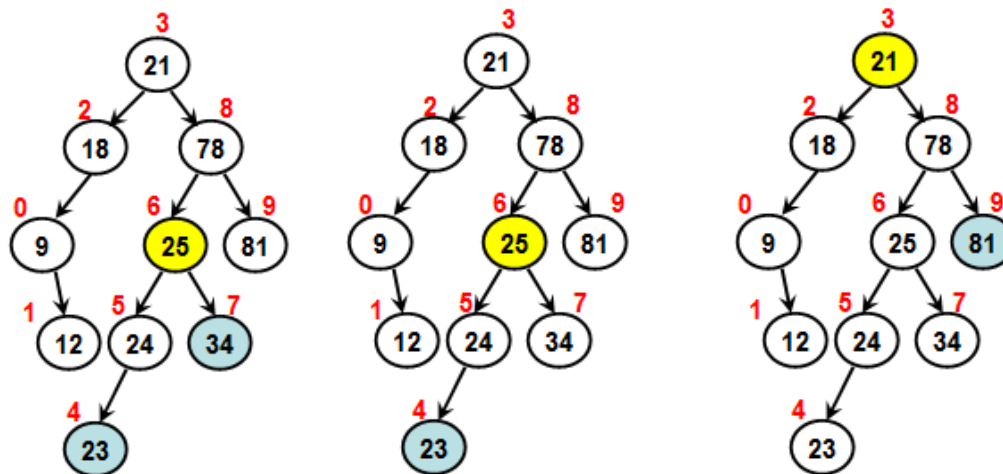


**CSC 228 Data Structures and Algorithms, Fall 2017**  
**Instructor: Dr. Natarajan Meghanathan**

**Project 9: Binary Search Tree: Lowest Common Ancestor Node**

**Due:** November 17, 1 PM (submission through Canvas)

The lowest common ancestor (LCA) for two nodes A and B in a binary search tree (BST) is the node that is the common ancestor for both A and B, and is the farthest away from the root node of the BST. Note that depending on the BST, one of the two nodes A and B could themselves be the LCA of the other node or a third node (that is different from nodes A and B) could be the LCA. The three cases are illustrated in the following figure:



Node '6' with data 25 is the LCA for nodes '4' and '7' with data 23 and 34 respectively

Node '6' with data 25 is the LCA for nodes '4' and '6' with data 23 and 25 respectively

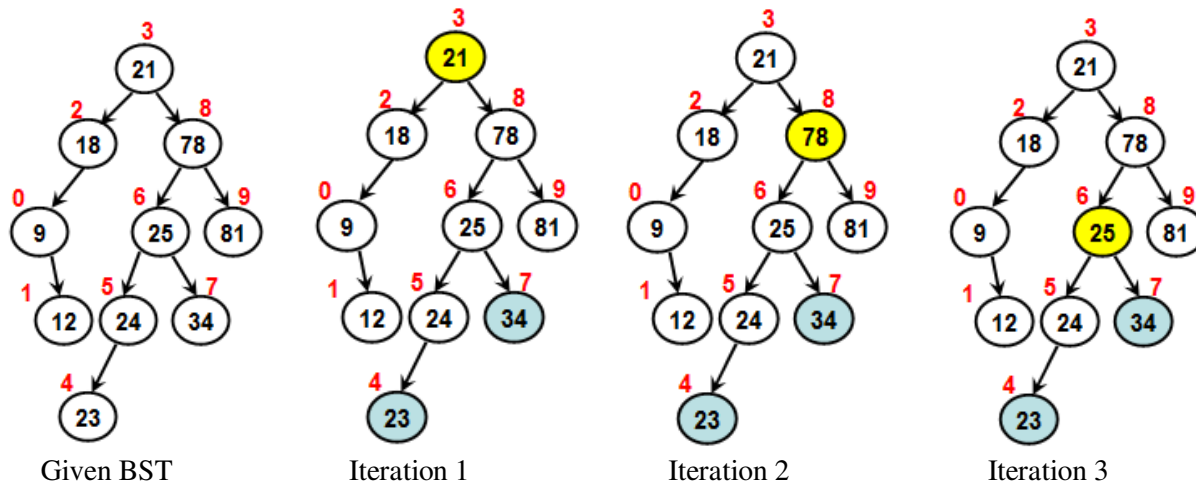
Node '3' with data 21 is the LCA for nodes '3' and '9' with data 21 and 81 respectively

In this project, you will construct a BST based on a randomly generated and sorted array of integers, input the values (i.e., data) for two nodes A and B (that are part of the BST) and find the data corresponding to their LCA. Follow the steps below:

- (1) You will first construct a BST using a randomly generated and sorted array (sorted using Selection sort). The code to construct such a BST is given to you.
- (2) You will then input the data/values for the two search nodes A and B. Between the two nodes, determine the node whose data is relatively lower and call such a node as the leftSearchNode; the other node will be then called the rightSearchNode.
- (3) The third and the major step is to initiate a search process starting from the root node (called say, the prospectiveAncestorNode).
  - If the data for the leftSearchNode and rightSearchNode are both lower than that of the prospectiveAncestorNode, then the two nodes should be located in the left sub tree of the prospectiveAncestorNode. Hence, we set the id of the prospective ancestor node to correspond to the id of its left child and continue the search process in its left sub tree.
  - If the data for the leftSearchNode and rightSearchNode are both greater than that of the prospectiveAncestorNode, then the two nodes should be in the right sub tree of the prospectiveAncestorNode. Hence, we set the id of the prospective ancestor node to correspond to the id of its right child and continue the search process in its right sub tree.

- If the data for the leftSearchNode is lower than or equal to that of the prospective ancestor node and the data for the rightSearchNode is greater than or equal to that of the prospective ancestor node, we stop and return the data/value of the prospective ancestor node as the LCA.

The execution of the LCA search process is illustrated in the example below:



Let nodeA be node '4' with data 23 and nodeB be node '7' with data 34. Between these two nodes, nodeA has the lower data and hence nodeA is the leftSearchNode and nodeB is the rightSearchNode.

Start with the root node '3' with data 21 as the prospectiveAncestorNode; the data for the leftSearchNode and rightSearchNode are 23 and 34 respectively.

Iteration 1: The data for both the leftSearchNode (23) and rightSearchNode (34) are greater than that of the prospectiveAncestorNode (21). Hence, we set the new id of the prospectiveAncestorNode to that of the id of its right child (node id '8' with data 78).

Iteration 2: The data for both the leftSearchNode (23) and rightSearchNode (34) are lower than that of the prospectiveAncestorNode (78). Hence, we set the new id of the prospectiveAncestorNode to that of the id of its left child (node id '6' with data 25).

Iteration 3: The data for the leftSearchNode (23) is less than that of the prospectiveAncestorNode (25) and the data for the rightSearchNode (34) is greater than that of the prospectiveAncestorNode (25). Hence, we stop and return 25 as the data of the LCA for nodes with data 23 and 34.

**You are given** the code for constructing a BST based on a randomly generated and sorted array of integers.

### What to submit:

The complete code (in a word document) with the following additions to the given code.

- (1) An enhanced main function that inputs two integers, corresponding to the data for nodeA and nodeB, and decides which of the two is the leftSearchNode and rightSearchNode.
- (2) A member function findLCA(...) implemented in the class BinarySearchTree, and is called with the leftSearchNode and rightSearchNode as the two arguments. The findLCA function should implement the procedure described above and return the data corresponding to the LCA. Test your code with an array of at least 15 integers. Capture the execution of the code and include as a screenshot (of the contents of the array and the LCA).