National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structures and Algorithms	Course Code:	CS2002
Program:	BS (Electrical Engineering)	Semester:	Fall 2021
Assessment Tool	Programming Assignment # 2		
Total Marks:	50		

Submission Guidelines:
Archive your files (.h and .cpp) in a zip file named as your roll number(s). Upload this file on slate under the assignment submission section till Monday 29th Nov 2021 till 4pm.
Code should be documented and well written in C++. It must compile.

VERY IMPORTANT

Academic integrity is expected of all the students. Plagiarism or cheating in any assessment will result in negative marking or an **F** grade in the course, and possibly more severe penalties.

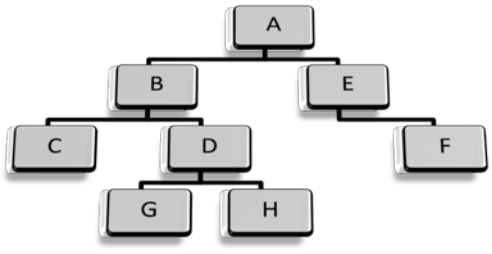
PROBLEM 1 — CLO [4]

a. Please implement the following member functions for a Binary Tree class. Please note that you have to write a client (main) function that demonstrates the use of these functions.

i. `bool BinaryTree<DT>::isComplete()`
// returns true if the binary tree is complete
//your implementation should be **recursive**

ii. //returns the number of nodes present at the level number passed in as //parameter
`int BinaryTree<DT>::numberofNodesAtLevel(int level)`
//your implementation should be **iterative**

iii. (Practice problem 8) Given two distinct nodes in a binary tree, there will always be exactly one path between them: from the starting node … to the lowest common ancestor of both nodes … to the second node. For example, in the tree below, the lowest common ancestor of nodes C and H is node B.



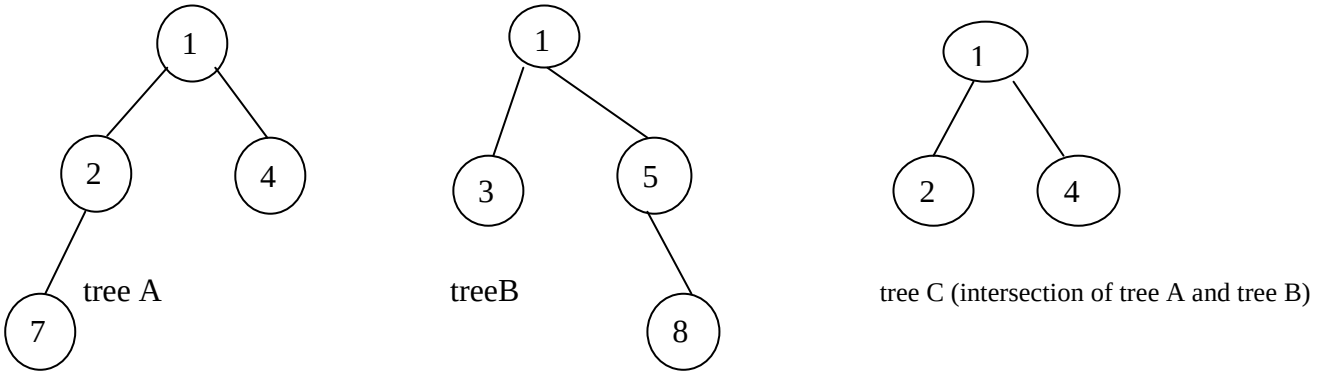
Given two nodes of a binary tree, where each node has a pointer to its parent, write a function

```
BNode<DT>* BinaryTree::lowest_common_ancestor(BNode<DT>* a,BNode<DT>* b)
```

which returns the lowest common ancestor of nodes a and b. Also give the time complexity of your code.

b. (Practice problem 10) **Please implement the following non-member functions for a Binary Tree class. Please note that you have to write a client (main) function that demonstrates the use of this function**

Write a function that constructs a binary tree resulting from the structural intersection of two binary trees i.e. only the nodes that are present in both trees are created in the third one, but the values of the keys are not important and can differ. For example, if the root nodes of the two trees both have a left child then a left child is added to the root of the resulting tree, otherwise it is set to NULL and so on. When a new node is created in the resulting tree for two corresponding nodes of the input trees, the minimum value is copied to the newly created node. An example is given below:



The function should take root node pointers (of tree A and tree B) as input parameters and return the root pointer of the newly constructed tree (tree C). If you use any helper functions then implement those as well. Also a node DOES NOT have a pointer to parent node and there is no function to return the depth of the tree. The overall time complexity of your function must not be worse than O(n).