


# National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Data Structure BSCS	Course Code:	4th 4A, 4B
	Program:		Semester:	
	Name:		Section:	
	Registration #:		Assessment	
	Due Date:	13 <sup>th</sup> April 2020	Assignment 2	

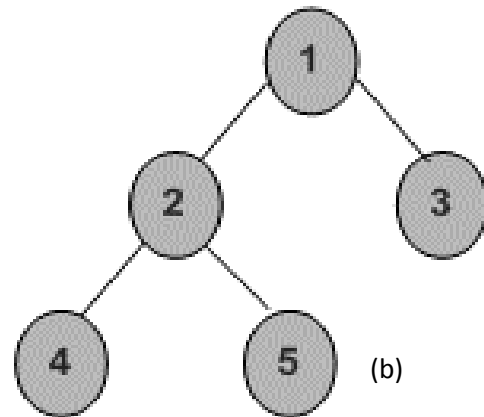
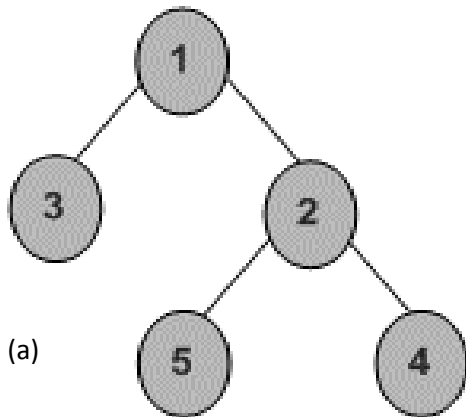
## Instruction/Notes:

1. The term **Efficient** means in terms of Time Complexity and Memory Requirements, the code should be better
2. Write down algorithms first to avoid any ambiguity during assessment of your assignment
3. **Under any circumstances, late submissions will not be entertained. It wastes plenty of precious time.**
4. Don't use containers and iterators to solve this assignment

## Q1:

[4+4+2=10]

Given a Binary Tree, construct a mirror tree such that the nodes are in reverse order. We are assuming there are no duplicate entries involved. For example for the given tree in (a), the mirror tree generated is shown in (b)



The simplest algorithm to construct this tree is perform a pre order traversal and construct a new tree with inverse properties i-e all the values less than the root should be inserted to the right side and all the values greater than the root should be inserted on the left side.

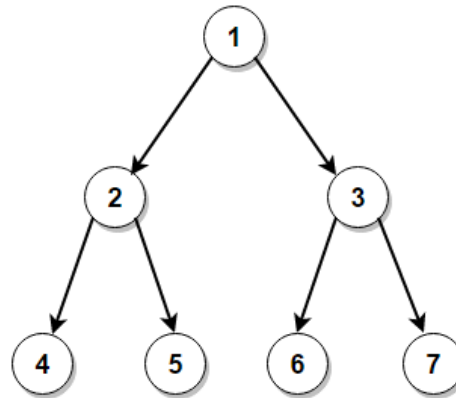
- a) Write down a code to perform the above mentioned task (Iterative solution). What is the time complexity and memory requirements?
- b) Write down an efficient code (with respect to memory or execution time complexity) to perform the said task.
- c) What will be the advantage/ disadvantage if we use recursive instead of iteration? (Answer in terms of memory and time complexity requirements)

## Q2:

[3+2=5]

[Spiral Order Traversal]

Given a binary tree, prints all its nodes level by level in spiral order. All the nodes present at level 1 should be printed left to right and all the nodes present at level 2 should be printed right to left. Then all the nodes in the level 3 should be printed left to right and all the nodes in level 4 right to left and so on. For example the spiral order traversal for the given tree is **1, 2,3,7,6,5,4**



For this, you need to find the depth of the tree first and then pick appropriate traversal method.

- Write down a C++ code to print spiral order traversal of the binary tree.
- Write down as much as efficient code to perform the said task.

**Q3:**

**[5]**

**Diameter of Binary Tree:** Given a binary tree, you need to compute the length of the diameter of the tree. The diameter of a binary tree is the length of the longest path between any two nodes in a tree. This path may or may not pass through the root.

Example:

Given a binary tree

```

      6
     / \
    4   8
   / \
  2   5
  
```

```

/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class MySolution {
public:
    int diameterOfBinaryTree(TreeNode* root) {
        //Write your code here
    }
};
  
```

**Return 3**, which is the length of the path [2,4,6,8] or [5,4,6,8]. You have to write a code with linear time complexity. Can you improvise this code further? (Yes/No) Justify your answer in either case.

**Q4:**

**[1+4=5]**

Given a singly linked list where elements are sorted in ascending order, convert it to a height balanced BST. For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of *every* node never differ by more than 1.

Example:

Given the sorted linked list: [-10,-3,0,5,9],

One possible answer is: [0,-3,9,-10,null, 5], which represents the following height balanced BST:



a. How will you select root of the binary tree?

b. Give a **recursive** solution for constructing such tree. What will be the memory and time complexity?

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */

/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */

class Solution {
public:
    TreeNode* sortedListToBST(ListNode* head) {

    }
};
```