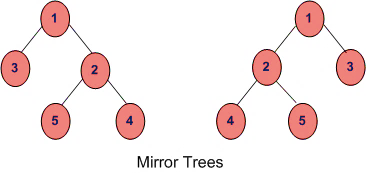
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| C:\Users\saif\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\final design.jpg | **Course:** | **Data Structure** | **Course Code:** | **CS231** |
| **Program:** | **BSCS** | **Semester:** | **3rd** |
| **Name:** |  | **Section:** | **3E, 3F** |
| **Registration #:**  **Due Date:** | **21st Nov 2020** | **Assessment** | **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. You can use containers and iterators to solve this assignment

**Q1: [(4+1) + (4+1) =10]**

1. 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)



(b)

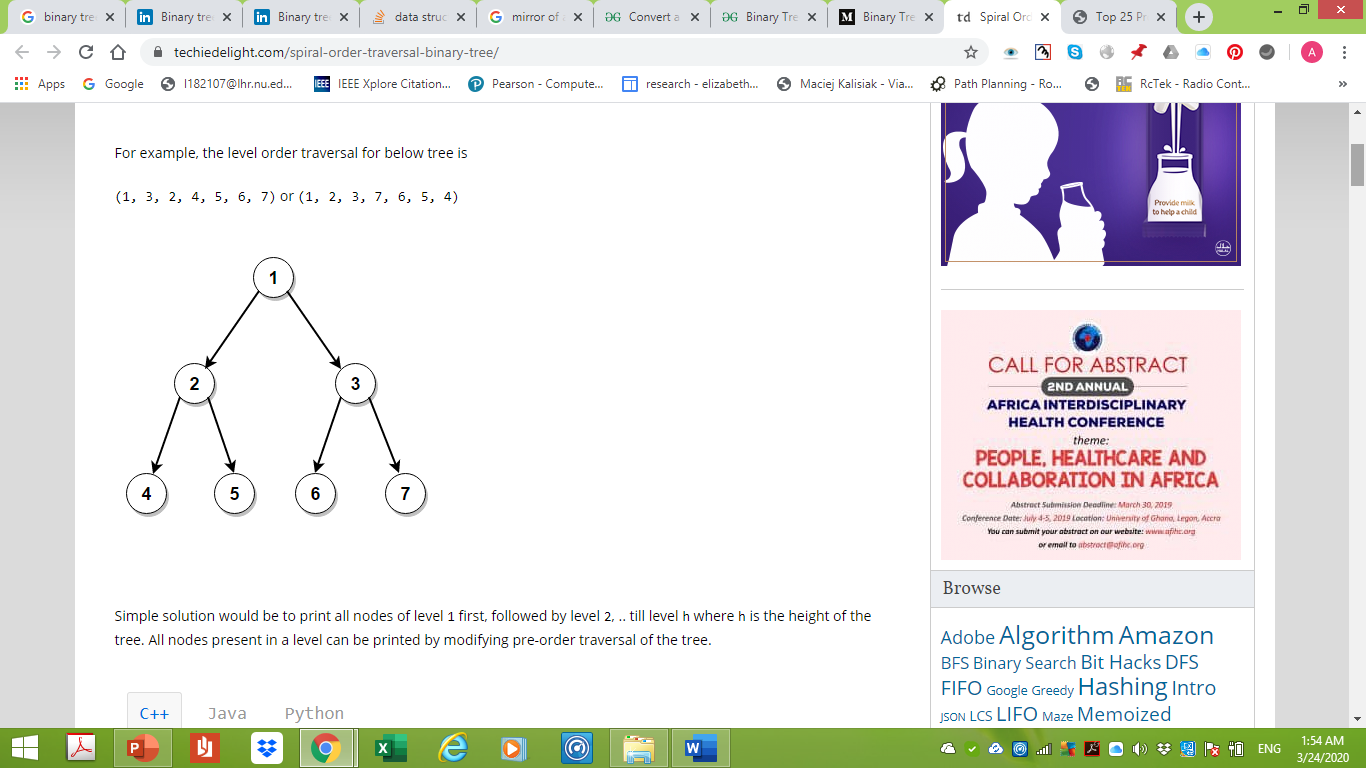
(a)

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.

* Write down a code to perform the aforementioned task (Iterative solution is required).
* What is the time complexity and memory requirements?

1. [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.

1. Write down a C++ code to print spiral order traversal of the binary tree.
2. What is the time complexity of your algorithm?

**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.

/\*\*

\* Definition for a binary tree node.

\* class 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

}

};

Example:

Given a binary tree

6

/ \

4 8

/ \

2 5

**Return 3**, which is the length of the path [2,4,6,8] or [5,4,6,8]. Write a code with linear time complexity.

**Q4: [4+1 =5]**

1. Write down a C++ implementation to flatten a binary tree to a linked list. For example, given the following tree:

10

/ \

5 15

/ \ \

3 7 16

The flattened tree should look like: **10 -> 5-> 3->7->15->16**

/\*\*

\* Definition for a binary tree node.

\* class TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* public:

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

void flatten(TreeNode\* root) {

}

};

1. Write down time and memory complexity of your algorithm

**Q5: [4+1 = 5]**

Given two AVL trees T1 and T2, where the largest key in T1 is less than the smallest key in T2, Join(T1, T2) returns an AVL tree containing the union of the elements in T1 and T2.

1. Give an algorithm (in pseudocode) for Join that runs in linear time at most.
2. What is the overall complexity of your code?