University Of Central Punjab

Faculty of Information Technology

Final Exam

Summer 2022

Data Structures and Algorithms – Lab

Time: 1 hour 30 minutes **Total Marks :30**

Instructions for Invigilators:

1. Students will have 1 hour 20 minutes to finish the whole exam plus ten minutes for submission. It is up to the students to manage their time.

Instructions for Students:

1. Please create file with appropriate name.
2. Submit only .h and .cpp files with output **SCREENSHOTS** on portal.
3. Late submissions will **NOT** be accepted.
4. Create as many classes and functions as required. Remember one function for one functionality.
5. Take care, Plagiarism will not be tolerated in any case.
6. The paper is closed book closed notes, NO cheat sheets allowed.

You are supposed to attempt any 2 questions out of three.

Question 1 – 15 Marks:

Write a menu-based program to implement the following operations on two circular linked lists L1 and L2:

1. Press 1 to append L1 to L2
2. Press 2 to append L2 to L1
3. Press 3 to merge L1 and L2 in sorted order.

<u>Sample Inputs:</u>

L1: 23 -> 56-> 78->13 -> pointing to 23
L2: 45-> -2->8 -> pointing to 45

<u>Sample Outputs:</u>

<u>Option 1:</u>

45-> -2->8 ->23 -> 56-> 78->13 -> pointing to 45

<u>Option 2:</u>

23 -> 56-> 78->13 ->45-> -2->8 ->pointing to 23

<u>Option 3:</u>

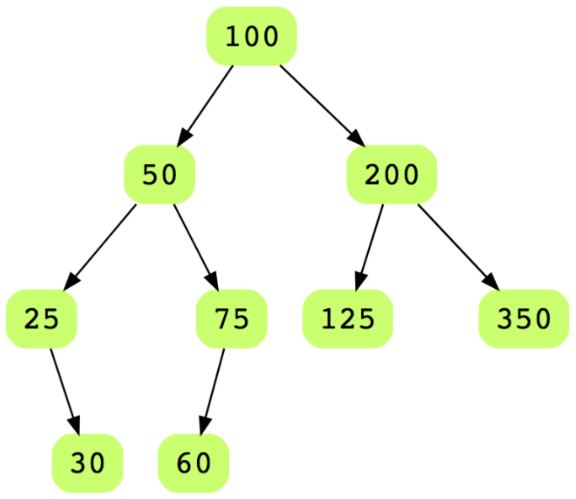
-2-> 8 -> 13 ->23 -> 45-> 56-> 78 -> pointing to -2

Question 2 – 15 Marks:

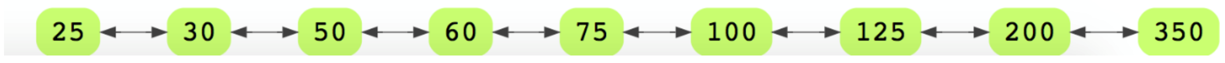
Given a binary Search tree, write a function in C++ to convert a BST into a doubly linked list using in-order traversal.

The conversion should be done such that the left and right pointers of binary tree nodes should act as previous and next pointers in a doubly-linked list, and the doubly linked list nodes should follow the same order of nodes as in-order traversal on the tree.

Consider the tree below:



Its in-order traversal will be 25, 30, 50, 60, 75, 100, 125, 200, 350. So the output doubly linked list should look like so:



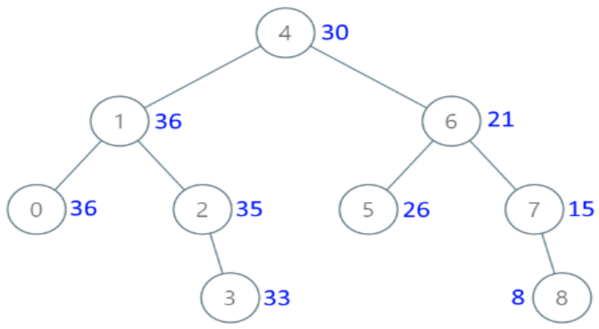
Question 3 – 15 Marks :

Given the root of a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus the sum of all keys greater than the original key in BST.

As a reminder, a *binary search tree* is a tree that satisfies these constraints:

- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

Example 1:



```
Input: root = [4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]
Output: [30,36,21,36,35,26,15,null,null,null,33,null,null,null,8]
```

The above output shows a level order traversal of BST.

Prototype:

TreeNode* bstToGst(TreeNode* root) { }

Note: Your program should be menu based, and only exit on pressing 4

- Press 1 to insert values to the tree
- Press 2 to display the inorder/preorder/post order/level order traversal of tree (You are free to choose one form of traversal).
- Press 3 to generate greater sum tree.
- Press 4 to exit