# CSC 212 Homework # 4
# BT & BST
# **Due date: 11/12/2016 at 5:00 AM (Morning)**

| | |
|---|---|
| Guidelines: | This is an individual assignment.<br>The homework must be **submitted electronically through LMS**.<br>**Hard copy submissions are not accepted**. |

**Remark.** *Consider the following when answering questions:*

1. *If a question specifies the type of programming to be used (recursive or iterative), you have to respect that requirement, otherwise you are free to choose the technique you deem appropriate.*

2. *When writing a recursive method* ***do not use*** *class data members (static or not) to save partial results. All newly defined variables must be local.*

you fail to do so, I.

## Problem 1

1. Write the method `private boolean areMirror(BTNode<T> t1, BTNode<T> t2)`, member of $BT$, which checks if two subtrees $t_1$ and $t_2$ are the mirror image of each other.

2. Write a **recursive** method *swap*, member of the class $BT$ (binary tree), that accepts a node $t$ as input. Starting from $t$, it swaps the data of every node with the data of its left child if it exists, otherwise it swaps with the data of its right child. If both children do not exist, the data of the node is unchanged. The signature of the method is `private void swap(BTNode<T> t)`.

## Problem 2

1. Write the method *collectLeaves*, user of the ADT Binary Tree, which returns a list of the data contained in all leaf nodes (the data must be inorder). Signature: `public static <T> LinkedList<T> collectLeaves(BT<T> bt)`.

2. Rewrite the method *collectLeaves* as a member of the class *BT*.

   Signature: `public LinkedList<T> collectLeaves()`.

## Problem 3

1. Write the method `public static isBST(BT<Integer> bt)` that takes a binary tree and returns true is the values in the tree are those of a BST, otherwise returns false.

2. Suppose that `BT<Integer> bt` satisfies the BST order. Write the method `public static boolean find(BT<Integer> bt, int k)`, which searches for the value $k$ in the tree. The method *find* has the same specification as that of *BST*.

## Problem 4

1. Write the method `private void swapData(int k)`, member of the class *BST*, that swaps the data associated with the key $k$ with the data of its parent. If the key $k$ does not exist or the corresponding node has no parent, the tree is unchanged. **Do not call any methods of the class *BST*.**

2. Write a member method that prints the keys of a BST in reverse order (note that in-order traversal of a BST visits the nodes in increasing order, so how can we visit the largest keys first?).

## Problem 5

1. Write the method `public int nbInRange(int k1, int k2)`, member of the class *BST*, which returns the number of keys $k$ in the *BST* that statisfy: $k_1 \leq k \leq k_2$. Make sure that the method does not visit any unncessary nodes.

2. Write the method `private int deepestKey(BSTNode<T> t)`, member of *BST* which returns the key associated with the deepest node in the subtree $t$. In case of a tie (two keys at the same depth), the smallest key should be returned. As precondition, $t$ must not be empty.