LAB 5 - Implementation of Binary Search Trees

Due Date: L01: Mar.21; L02: Mar.28; L03: Mar.22; L04: Mar.29; L05: Mar.23; L06: Mar.30; L07: Mar.24; L08: Mar.31; L09: Mar.25; L10: Apr.1

Assessment: 4% of the total course mark.

Description:

In this assignment you are required to write a `Java` class `BSTree` representing binary search trees, where each tree node stores an integer. Additionally, you will have to implement `Java` classes `MyStack` and `MyQueue` that you will use to perform a non-recursive inorder traversal, respectively a level-order traversal of a `BSTree`. You must also write a test class `TestBSTree`. To implement the tree nodes you have to use the `Java` class `TNode` provided in this assignment. You are not allowed to use any predefined `Java` methods or classes from `Java API`, other than for input and output.

Specifications:

- You have to use the `Java` class `TNode` given below, to implement the tree nodes. Classes `TNode` and `BSTree` must be contained in the same package.

```
public class TNode{
        int element;
        TNode left;
        TNode right;
        TNode(int i, TNode l, TNode r)
            { element =i; left = l; right = r; }
}//end class
```

- Class `BSTree` must contain **only** a `private` field named `root`, of type `TNode`, which is a reference to the root of the tree. No other fields are allowed. When the `BSTree` is empty you should have `root==null`.
- Class `BSTree` must contain **at least** the following constructors:
  - `public BSTree()` - creates an empty `BSTree`.
  - `public BSTree(int[] sortedInput)` - creates a `BSTree` of **minimum height**, storing all elements from array `sortedInput`. Note that this array does not contain repetitions and is sorted in increasing order.
- Class `BSTree` must contain **at least** the following methods (pay attention that you may need to write some additional `private` methods as "helpers", for instance, when using recursion):
  1) `public boolean isIn(int v)`: Returns `true` if integer `v` is stored in some node of `this BSTree`. It returns `false` otherwise.
  2) `public void add(int v)`: Adds `v` to `this BSTree` if `v` was not already an element of `this BSTree`. It does nothing otherwise.
  3) `public void remove(int v)`: Removes `v` from `this BSTree` if `v` was an element of `this BSTree`. It does nothing otherwise.

4) `public int size()`: Returns the number of integers stored in `this BSTree`.
5) `public int height()`: Returns the height of `this BSTree`.
6) `private void printRec(TNode t)`: Prints the integers stored in the subtree rooted in `t`, in increasing order. This method is **recursive**. Note that this method is a "helper" for the next `public` method.
7) `public void printRec()`: Prints the integers stored in `this BSTree` in increasing order. This method invokes `printRec(root)`.
8) `public void printNonRec()`: Prints the integers stored in `this BSTree` in increasing order. This method is **nonrecursive** and uses a stack to implement the inorder traversal (use the class `MyStack`). See tutorial notes for the algorithm.
9) `public void printLevelOrder()`: Prints the integers stored in `this BSTree` in level order, using a queue (use the class `MyQueue`). See tutorial notes for the algorithm.

NOTES:

- You may use the code from the lecture notes on binary search trees, stacks and queues. Of course, you will need to adapt the code. Additionally, you will have to be able to explain the code/algorithm to the TA at the demo.
- NO REPORT IS NEEDED. Submit the source code for each of the `Java` classes in a separate text file. Include your student number in the name of each file. Submit the files in the Dropbox on Avenue by 11:59 pm the day of your designated lab session.
- To get credit for the assignment you have to demonstrate your code in front of a TA during your lab session. A 50% penalty will be applied for late demo. A 25% penalty will be applied if the demo is on time, but the electronic submission is late.