Data Structures and Algorithms, Spring '23 section 1 Dr. Hristescu

Lab 11

Assigned: Thursday, April 13
Due: 10pm Monday, April 17

Solutions to problems different than specified in the description or not following the requirements of this lab will not receive any credit. Follow the instructions carefully.

Consider the reference based implementation of the Binary Search Tree (**BST**) ADT BinarySearchTree, TreeException, KeyedItem, TreeNode, BinaryTreeBasis. The keys in the BST need to be unique.

Problem 1: Design a class called **MyBinarySearchTree** using this frame:
Implement the following 3 methods using the **iterative** approaches discussed in class:

- **retrieve**
- **findLeftmost**
- **deleteLeftmost**
- **(Extra Credit) implement other methods than the ones listed above using an iterative approach if they do not need to be recursive**

Problem 2: Define a subclass of **MyBinarySearchTree** called **MyBinarySearchTreePlus** that implements additional functionality specified in **BSTPInterface .**

Use the following class header:

```
public class MyBinarySearchTreePlus <T extends KeyedItem<KT>,KT extends Comparable<?
super KT>> extends MyBinarySearchTree<T,KT> implements BSTPInterface<T,KT>
```

Implement the new functionality specified in **BSTPInterface** using **recursive** approaches for:

- **toStringInorder** - returns String representation of Tree with items in Inorder
- **toStringPostorder**- returns String representation of Tree with items in Postorder
- **toStringPreorder**- returns String representation of Tree with items in Preorder
- **hasCharacteristic -** returns true if the node has the characteristic that all even levels have nodes with either 0 or 1 children and all odd levels have either 0 or 2 children; false otherwise (root is on level 0(=even), root's children are on level 1, and so on)

Problem 3: Develop a test application with the following menu options:

0. Exit
1. Search for item in BST
2. Insert item in BST
3. Delete item from BST
4. Display content of BST in-order
5. Display content of BST in post-order
6. Display content of BST in pre-order
7. Check if BST has required characteristic

Use an **Item** class structured like the textbook's Person class (page 596) that is a subclass of the abstract class **KeyedItem**. Use as the **Comparable** key an Integer (passed to the super constructor as a parameter) just like in Person and add two more data fields one of type **boolean** named **assocboolean** and the other one of type **String** named **assocstring** but with appropriate getters and setters associated with them as well as a toString method that returns the information in the format **{key/assocboolean/assocstring}.**

Use the following class header for Item:

```
public class Item <KT extends Comparable<? super KT>> extends KeyedItem<KT>
```

Note: **In the driver** make sure that duplicate keys are NOT inserted, i.e. Menu option 2 does not insert an item in the tree if the item's key is already in the tree. [Driver reads all item data before rejecting item based on duplicate key]. A sample run of the program on this input data can be found here.

Problem4: Lab11Conclusions: Summarize in a few sentences what you have learned from working on this lab.

Submit: Follow the instructions to turn in the lab electronically using **Lab11** for the name of your directory. Submit all the code developed for this lab and all **sample runs on the provided input data** as well as any additional testing suite that you developed.