American University of Armenia, CSE
CS 121 Data Structures A, B, C
Fall 2019

Homework Assignment 6

Due Date: Wednesday, November 27 by 23:55 electronically on moodle

Any submissions containing cheating and/or plagiarism, by university policy, will be reported and will result in the grade F for the entire course.

Please solve the programming tasks either in Java or C++, following good coding practices (details posted in moodle).

You should submit full tested programs for all questions.

1. **(40 points)** Write a class `BSTMap` that extends the `AbstractSortedMap` class using a `LinkedBinaryTree` *tree* as the underlying data structure. Note that it should maintain *tree* as a binary search tree. All the external nodes in a BST are sentinel nodes, i.e. they store `null` instead of a real entry. This implies that when the map is empty, the *tree* BST should contain only a single sentinel node at the root.

   Your class should support all of the following functionality:

   (a) two constructors that create an empty tree: a no-arg constructor that relies on the default comparator, and another constructor that receives a comparator argument for the key-type;

   (b) `size` method (note that `isEmpty` is inherited from `AbstractMap`);

   (c) a utility method `expandExternal`;

   (d) a utility method `treeSearch`;

   (e) the fundamental methods `get`, `put`, and `remove` from the Map ADT;

   (f) the `firstEntry, lastEntry, floorEntry, ceilingEntry, lowerEntry`, and the `higherEntry` methods from the Sorted Map ADT;

   (g) the `entrySet` method from the Map ADT (note that `keySet` and `values` methods are inherited from `AbstractMap`);

   (h) the `subMap` method from the Sorted Map ADT.

   You may want to add more utility methods to your class to simplify the implementation of the above functionality.

2. **(15 points)** Write a method that, given the root node of a binary search tree and a comparator for strings, checks if the tree is balanced (for every internal node v, the heights of the children of v differ by at most 1). The entries have string keys and integer values. To test your method, write a comparator class for strings that orders the strings based on the ASCII ordering of their first characters.

3. **(15 points)** Write a generic method that, given the root node (this is the `Node` class for linked binary trees) of a BST or AVL tree, returns the key of the median entry.

4. (15 points) Write a program that reads a sequence of integers and prints these integers in the sorted order of their sums of digits. If multiple integers have the same sum of digits, only the first such number is printed. Your program should use a map. Test your program with a `SortedTableMap`, a `BSTMap`, and an `AVLMap`; compare their performance on the same sequence of 1000 integers, recording actual execution times.

5. (15 points) *A problem on hash tables coming soon.*