Programming Assignment 5 - Hash Tables and Binary Search Trees

Submit Assignment

Due May 24 by 11:59pm **Points** 100 **Submitting** a file upload
File Types java and cs **Available** May 12 at 8:50pm - Jun 18 at 11:59pm about 1 month

Hash Tables and Binary Search Trees

In the last programming assignment you were to implement a hash table that resolved collisions with chaining. Each non-empty bucket in the hash table array contained a reference to a Java LinkedList<Integer> obect. As collisions were encountered, items that collided were added to a linked list referred to by the bucket the item hashed to.

For this assignment, each non-empty bucket in the bucket list will hold a reference to a Binary Search Tree. You are to implement the Binary Search Tree from the pseudo code provided in chapter 6 of the zybook for this course

The hash table must meet the following requirements:

- the underlying data structure, hashArray, is a fixed size Java array (not an ArrayList) of type BinarySearchTree
- The data field for the Node in the BinarySearchTree must be of type Integer
- When an item is added to an empty bucket, instantiate a BinarySearchTree and store a reference to it in the bucket
- when the loading factor reaches 1.0, resize hashArray to the next prime number greater than twice the hash Array's current size, and then rehash
- the hash function is

```
hashValue = item % hashArray.length
```

- the initial length of the hash array is passed to the constructor as an int parameter
- HashTable, the class that defines the hash table must have the following minimal set of public methods:
  - HashTable(int sz) //a constructor that takes a single int parameter
  - void insert(Integer key) //takes an Integer parameter and stores it in the list
  - void remove(Integer key) // find an item in the hash table and remove it
  - Integer search(Integer key) // searches for an item in the hash table and if found, return it. Otherwise, return null
  - String toString() // outputs the contents of the hash table, one row per element in hashArray

    - int size() // return the number of elements in all of the chains
    - int size(int index) // return the number of elements in the indexed chain
    - double loadFactor() // a method the calculates and returns the loading factor to the caller
- and the following private method(s):
    - rehash() - resize hashArray to the next prime number greater than twice the hash Array's current size. Then for each element in the original hash array, rehash it into new array (don't forget that the hash function is based on the size of the hash array)

When you are finished, upload the java files to canvas.