Programming Homework Assignment #5 (Updated)
Due Date: Thur., March 3, 11:55 PM
Upload the source files (.java files) with output (copied and pasted to the end of the main file) ZIPPED into ONE .zip file (submit under Week 9)

Problem: Write a Java program which changes the previous versions of HashSC AND HashQP classes and practices using Hash Tables with pointer to objects data.

Use the Color class given in the HW5_JavaCodeFile. You'll have to include the Node.java file from the Linked List Code folder in Catalyst (under Week 1), but also USE the UPDATED LList.java class given in the Hash Table Files folder under Week 6 in Catalyst (how to use is indicated in the HW5-CodeFile). Note that the Comparator<E> interface is in java.util.*

Update the HashSC AND HashQP class member methods given in the HW5-Code file (NOT in the Hash Table Files folder) (I'm calling "code file" below):

- make HashSC and HashQP SUBCLASSES of the abstract HashTable class (given in the Code File)
- add 2 interface parameters to the constructor, one for a Hasher<E> and another for a Comparator<E>, and pass them to their corresponding parameters in the superclass' constructor
- override the **getEntry**() method, like remove(), but assigns to returnedItem if found (need to write WHOLE method!)
- CHANGE in the HashSC class the **insert**, **contains**, and **getEntry** so any comparisions each method MUST traverse a linked list using its iterator (see **remove** and **rehash** for examples) AND will use the compare method from the Comparator (this is similar to the BinarySearchTree's use of the compare method) (DON'T call a linked list's contains() !)
- change in the HashQP class, the **findPos** so any comparision with Object will use the compare method
- add code to HashSC's **insert** and HashQP's **findPos** so they will increment the collisionCount (declared in the superclass) ONLY when a collision occurs (see answers to Lab Ex.8.2), and will update the longestCollisionPath when there is a longer linked list (in HashSC) OR the while loop in HashQP's findPos has iterations. In HashQP, make sure you reset the counters to 0 if before rehashing.
- change **myHash()** in HashSC AND HashQP so it doesn't call x.hashCode, but the Hasher<E> instance variable's **hash** method
- write the **traverseTable**() member method (in HashSC and HashQP) so each Object (template for item) will be visited by calling a method through the visit pointer to method parameter in the array or vector order (make sure to not display an item if the location isn't used)

Define the following classes using the Visitor<E> or Hasher<E> interfaces given in the "code file":

- **class ColorCodeVisitor implements Visitor<Color>**, so when you override the visit method, it displays the Color's code value (in hexadecimal),

then its String (you may use System.out.printf with %06X to display in hex)

- **class ColorNameVisitor implements Visitor<Color>**, so when you override the visit method, it displays the Color's String, then its code value in hexadecimal (see example runs for format)
- **class ColorCodeHasher implements Hasher<Color>** so when you override hash(Color) it returns the code value of the parameter
- **class ColorNameHasher implements Hasher<Color>** so when you override hash(Color) it returns an int hashing the String of the parameter the same way as shown in Lesson 8, or your own algorithm
- **class ColorCodeComparator implements Comparator<Color>** so it overrides only the **compare** method, return the difference between the first parameter's code value - second parameter's code value
- **class ColorNameComparator implements Comparator<Color>** so it overrides only the **compare** method, return the result of first parameter's String's compareToIgnoreCase (passing the second parameter's String)

Write main so it has 2 HashTable<Color> variables. Assign to one of the HashTable variables a new **HashSC<>** passing a new ColorCodeHasher and a new ColorCodeComparator, and the other to a new **HashQP<>** passing a new ColorNameHasher and a new ColorNameComparator. In main, do the following (each bullet should be a method):

- call a method (you write) to fill the a HashSC and HashQP (both are parameters in ONE method because they will refer to the SAME data). Open an input file the same way you did in Prog. HW#1 (using the openInputFile method, also given in the code file). If the file doesn't open, return false. If it opens:
  - read the file which has several sets of Color, which has an int in hexadecimal, a space, and a string for the name (read to the end of line, then trim()), create a new Color with data from the line of input, **insert** the same Color to each hash table (through its pointer). Read until the end of file (use the Scanner's hasNext() method). Close the file at the end of the method, then return true.
  - Hints: to read (from a Scanner) an int in hexadecimal, use Scanner instance method nextInt(16), and to read the name, READ THE REST OF THE LINE using the Scanner nextLine() instance method AND trim it (the name may have more than one word)!
- if the file-reading method wasn't successful, display "ending program" and end the program.
- call the *traverseTable* method for your HashSC and HashQP tables, passing the ColorCodeVisitor for HashSC and ColorNameVisitor for HashQP (make sure you display a description of which table is being displayed first)
- call the *displayStatistics* for each hash table
- call the *testHashTables* method (in the code file)
- AGAIN, call the *traverseTable* method for your HashSC and HashQP tables, passing the Visitor corresponding its Hash Table (make sure you display a description of which table is being displayed first)

Extra Credit Problem (due the last day of the quarter!): Change the HashQP class so it doesn't use quadratic probing, but a Random object for rehashing (hints will be given in a separate file upon request)