General Instructions

The TA's handling this assignment are below. Contact them by email at ____**@mail.mcgill.ca**.

- David Bourque (**david.bourque**) He will handle discussion board questions.
- James Bodzay (**james.bodzay**)
- Florence Clerc (**florence.clerc**)
- Antoine Soulé (**antoine.soule**)
- Howard Huang (**howard.huang**)

The general instructions are the same as in previous Assignments, in particular, 20 point late penalty per day up to 3 days late. We will deduct 20 points for any student who has to resubmit after the due date (i.e. late) because the wrong file or file format was submitted. This policy will hold regardless of whether the student can provide proof that the assignment was indeed "done" on time.

See the submission instructions at the end of this PDF.

Question 1 Hash Table (70 points)

You are given a **MyHashTable** class with code stubs, a **Song** class that describes song objects which will be stored in the hash table, and a **HashTableTester** class which uses a hash table to store a list of songs.

Your task:

Implement the following methods in the **MyHashTable** class:

- the constructor for the class **MyHashTable**
- the constructor for the inner class **HashIterator**
- **put()**, **get(), remove(), rehash()**, **keys(), values()**

For the specification of what these methods must do, see the comments in the code.

Question 2 Graph (30 points)

You are given a **Graph<T>** class, and a **Vertex<T>** class, with generic argument **T** that represents the type of object that the vertex will contain. Inside the **Graph** class, the vertices are represented using a java **HashMap** object that is referenced by the **vertexMap** field. The **HashMap** keys are labels for the vertices. All methods of the **Graph** class are implemented for you.

You are also given a class **ShapeGraph** which extends the **Graph** class. The generic type is **Shape** which is itself an abstract class. The classes **Triangle, Rectangle, Circle** which extend **Shape** are also given to you. These classes compute the shape properties defined in the comments. (All the material you need for this question will be covered by the end of lecture 31 on Wed. Nov 23.)

Your task:

Implement the **traverseFrom(String key, float threshold)** method in the **ShapeGraph** class. This method finds the paths to all vertices that are reachable from the vertex defined by the **key** parameter, subject to the following condition: you must only return paths such that the total area along the path (the sum of the areas of the vertices) is greater than the **threshold** parameter.

For example, if there is a path 1->2->4->6->7->9, with all vertices having area 5, then the list returned by **traverseFrom(1,16)** should contain the paths

1->2->4->6
1->2->4->6->7
1->2->4->6->7->9

The paths 1, 1->2, and 1->2->4 do not have total areas above 16, and thus are not returned.

Your solution must use a **recursive, depth first** implementation. You must visit nodes within the adjacency lists in the order defined by the list. This will ensure that all students get the same paths (since for some graphs there may be multiple ways to go from vertex A to vertex B).

<u>Submission instructions</u>

- Submit your java file **MyHashTable.java** to the A4 Q1 folder
- Submit your java file **ShapeGraph.java** to the A4 Q2 folder.

Do not submit directories. Do not zip files. Just submit the java files this time.

Check the Announcements FAQ for latest updates.

Good luck and have fun! You are almost done!