# **CS 240 - Data Structures and Algorithms I**

# **Programming Assignment #4**

Due date: 06/04/16 11:59 hrs

# Statement

The aim of this project is to get you conversant with hashing technique. You must develop your own hash table and hash functions instead of using the provided hashTable in Java. You should use the framework provided to implement a HashTable for different names need to print the average score of the name being searched for.

A bunch of files in which each line contains a team name and its score. The objective of this program is to process these files and calculate the average score for each team. Therefore HashTable is used to store the accumulated scores, the number of times the team showed up in the files and of course, the name of the team.

# Details

The whole process has two stages: 1) hashing and 2) testing.

In the **hashing** stage, the framework will extract the name and score from files and use the HashTable you implemented to store it. The scores will be added cumulatively. Which means when the framework is about to add a new pair of name and score into the HashTable, it will check whether the name already exists. If so, it will obtain the score already saved in the HashTable and add up the new score, then update the score to the same entry in the HashTable; otherwise, it will add this pair as a new entry in the HashTable. Your need to implement the HashTable using the Entry<K, V> class to store name and score pair. You need to implement chaining for collision resolution.

Following methods need to be implemented in order to make the whole program work.

* public HashTable(int tableSize)
  + Construct a HashTable in size of tableSize
* public void put(String key, Double value)
  + Put a key-value (name-score) pair into the HashTable
* public Entry<String, Double> get(String key)
  + Get the entry (name-score) from the HashTable if it exists; return null otherwise
* public void delete(String key)
  + Delete an entry matches given key (name)
* public boolean containsKey(String key)
  + Check if the HashTable contains given key (name)
* public boolean isEmpty()
  + Check if the HashTable is empty
* public int size()
  + Return the number of elements in the HashTable

In the **testing** stage, the framework will call a method getAverage() which you need to implement to obtain the average score for the team.

* public Double getAverage(String key) throws Exception
  + Return calculated average score for corresponding key (name) if it exists; return 0.0 otherwise

You may also implement any data structure or private methods necessary.

# Example

Sample output:

"Miami", expected=14.91, received=-1.0 [MISMATCHED]

"Green Bay", expected=27.36, received=-1.0 [MISMATCHED]

"San Francisco", expected=17.45, received=-1.0 [MISMATCHED]

"Baltimore", expected=18.82, received=-1.0 [MISMATCHED]

"New England", expected=26.18, received=-1.0 [MISMATCHED]

"Philadelphia", expected=26.64, received=-1.0 [MISMATCHED]

"Indianapolis", expected=34.55, received=-1.0 [MISMATCHED]

"Houston", expected=20.82, received=-1.0 [MISMATCHED]

Final score: 0 out of 35

Above are sample output once you run run.py. Each line represents a test, for example: “Miami” is the name of the team, “expected=14.91” is the expected (correct) average score for the team, “received=-1.0” is the average score obtained from your method, and lastly “[MISMATCHED]” indicates this test failed. You can see the “Final score” at last line.

# Code

Download PA4.zip from the Blackboard under Assignments → Programming Assignments → PA4.

# Code Structure

data/→ This directory contains test directory which contains all the input files from where you will get to know the names and their scores. test.txt is the input file which contains list of names and their average.

java/→ This directory contains all the JAVA code. You only need to modify HashTable.java; **Please do not modify other code in this folder**. It is mandatory that you use the LinkedList.java provided for implementing chaining for collision detection. Some of the methods have already provided to you. You only need to fill out the highlighted stubs in the code.

run.py­→ A tool to compile, run & test your code.

# Extract and Run

* Download the file PA4.zip. Extract it.
* Copy the directory to your ZFS system by following the basic linux guide provided.
* Let’s say you extracted into ‘PA4’ directory. Now, from the terminal:

$cd PA4

# To run and evaluate your code $./run.py

# Compatibility

Note that, all assignments will be tested under Linux environment with Python and Oracle Java is installed. Given code might work on other platforms (like Windows, etc.) but has not been tested. Hence, it is encouraged to develop and test your code in a Linux based environment.

# Submission

You should only modify and upload HashTable.java to Blackboard. Any change in other files will not be accepted and you will not be evaluated in that case.

# Evaluation

There is some held out data set against which your code will be tested and evaluated. Your main aim is to write a HashTable class following the specifications provided.

# Honor Code

I encourage students to discuss the programming assignments including specific algorithms and data structures required for the assignments. However, students should not share any source code for solution.

￼￼￼￼￼￼￼￼￼￼￼￼￼

Code exists on the web for many problems including some that we may pose in problem sets or assignments. Students are expected to come up with the answers on their own, rather than extracting them from code on the web. This also means that we ask that you do not share your solutions to any of the homework ­ programming assignments, or problem sets ­ with any other students. This includes any sort of sharing, whether face­ to ­face, by email, uploading onto public sites, etc. Doing so will drastically detract from the learning experience of your fellow students.