CS 260 Fall 2016 **HW 8: The Joy of Hashing**

Total: 16 points

**Posted 11/22/2016**

**Due 12/01/2016**

**Exercises – Requirements**

1. The main goal of this assignment is to create and exercise a hash table of college students. The key is the student ID (an object of the ID class) and the value is an object of the Student class.
2. Use the following classes: **Student, ID, StudentTable, Application**
3. The ID object represents a **pair** of two 10-digit integers, each integer will be randomly selected between 0 and 2,000,000,000. The reason for the random selection is that we will have to issue IDs for a large number of students (up to 1,000,000) and using a pair of 10 digit numbers allow us reasonably assume that different students will have different IDs.
4. The ID class has two instance fields to store the first and the second element of the pair. Define two constructors, a no-arg constructor randomly initializes the fields, an initializer constructor takes two parameters. The class has an equals( ) method to compare ID objects for equality of their respective fields.
5. The Student class has two instance fields, a String type **name**, and an ID type **id** to store the name and the ID of a student. Define and initializer constructor that takes a String and an ID type parameter to initialize the name and the id field. Define another constructor that takes only a String parameter for name, and calls the ID class constructor to initialize the id field. Use the constructor which fits best an instantiation need.
6. The Student class must have an equals( ) method to compare Student objects.
7. The StudentTable class implements a hash table as described in the class and in your reading (see pp 584 – 594). The elements are Student objects, the keys are ID objects.
8. Considering the way a key is defined, after careful deliberation choose a hash function you like.
9. Generic implementation is optional. In what follows the description assumes a non-generic implementation.
10. Modify the the **put**( ) method of your table class such that it shall take a name (String) and a key (type ID) for parameters. The method instantiates the corresponding Student object (the value) then the value and key will be placed in the table.
11. The StudentTable constructor takes a parameter for the size of the table, and instantiates all the three array fields (keys, data, hasBeenUsed) to that size.
12. The Application class tests and drives your program.
13. In the main method instantiate a StudentTable with size 30,000. Use open address hashing to populate the table with 15,000 students (load factor ½). Measure the time of executing the 15,000 put( ) calls. Use linear probing.
14. You will need names supplied to the Student constructor, therefore you need a name generator method in the Application class. Add a static helper method randomName( ) to the class. The method takes an int parameter for the length of the names (use 7). Every call of the method produces a random name such that the first character is a capital letter randomly chosen, the other characters are alternately random vowels and random consonants. To help the random selections use auxiliary arrays to store the vowels and consonants respectively. A short sample list of 49 such names are shown below:

Yupoyes, Rijaboq, Hipocun, Quvaxim, Uopexod, Facujeq, Vohiyip,

Lipeyib, Bezolul, Cefukon, Cemapen, Risisay, Dijewuk, Bepetoz, Kojimoy, Eaxikar, Xihahef, Uimojuq, Metuver, Hepejah, Yawihuk, Xocasal, Bulekiw, Eejuvas, Fapasaj, Gulobok, Uupinof, Fajonum

Vuyinor, Lopoyas, Qotiqab, Reduder, Iixelup, Hemepuk, Penilir

Piyivaw, Uajaqub, Xugoruq, Yirediy, Zudakup, Xavamow, Ramedaz

Aeriwov, Huqurer, Wifaguf, Qahuloj, Jojewiy, Qomanev, Iipunap

1. In the testing phase check if the ID numbers are created correct.
2. To display the table, find the first 50 non-null Students in the table and print their names and the corresponding array indexes to the console.
3. Select a student randomly from the table and using its key call the remove method to remove the selected element.
4. Repeat steps 13, 16 and 17 for a table of capacity 1,000,000 with 500,000 elements.
5. As non-mandatory but recommended experiments you may want to test other hash function(s), other hashing algorithms, and the large table with greater load factors as well.
6. **Submit**: Name your project <your name>\_HW8. Zip the project folder and upload it to Blackboard.