CS0048 DATA STRUCTURES AND FILES

Assignment 8 – Separate Chaining in Hash Tables

Basic Idea

Use separate chaining to resolve collisions in a hash table. Your program should use a hash table of size 887 and the data from **animals.csv** which contains the following field entries separated by commas:

ID,animalName,kingdom, phylum,subphylum,animalClass,animalOrder,family,genus

The **key** will be the name of the animal and the **value** will be the whole **Animal** record as an object. The key is not unique and the hash table will store different values that have the same key – do not replace any entries. Create the following classes

Animal to store information about an animal
AnimalNode to store a node with key, value, and next attributes
AnimalChain to store a chain of **AnimalNode**s
Hashing to set up and perform the hashing and print results

Information on Hashing

Hash Table Contents: The hash table will store instances of the class **Animal**. All records from the file must be hashed to the table.

Calculating the Hash Code: The key to be used is the **name** of the animal. The key must be converted to a hashcode as follows:

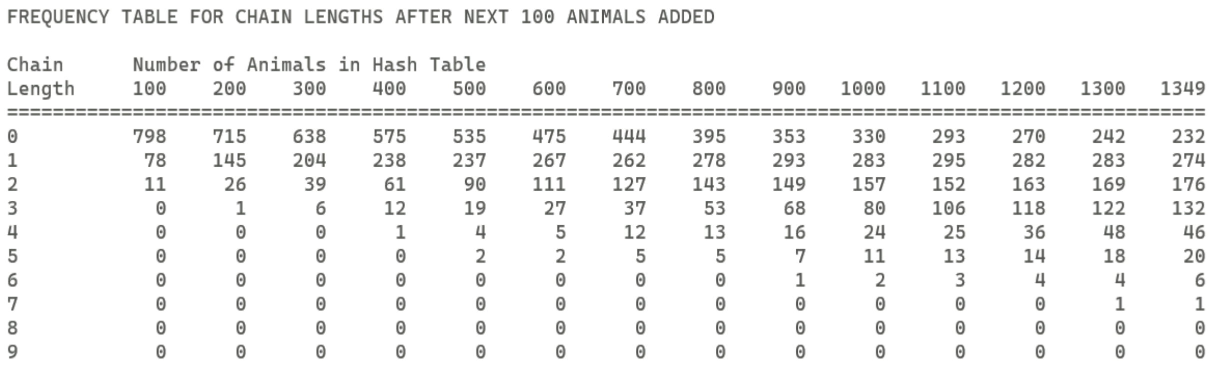
(1 * int equivalent of first character in name) + (2 * int equivalent of second character in name) +
(3 * int equivalent of third character in name) + (4 * int equivalent of fourth character in name) +
(5 * int equivalent of fifth character in name) and so on

Example: The first animal in the file is Arctic Wolf which will give a hash code of 6294.

Calculating the Hash Function: The hash function will take the hash code and mod it with the size of the hash table

Collision Resolution Method: Separate chaining is used – there will be no probing. Your hash table should have an **AnimalChain** at every index. Any object hashed to that index is added to the chain. No check for duplicate keys is necessary.

Results: Your program will successively place animals into the table. After each interval of 100 insertions, do a frequency analysis on the lengths of the chains. All the frequency results must be stored in a 2 dimensional array and printed after all values have been written to the hash table. The output should look something like this (results not correct):



```
FREQUENCY TABLE FOR CHAIN LENGTHS AFTER NEXT 100 ANIMALS ADDED

Chain      Number of Animals in Hash Table
Length     100    200    300    400    500    600    700    800    900   1000   1100   1200   1300   1349
==========================================================================================================
0          798    715    638    575    535    475    444    395    353    330    293    270    242    232
1           78    145    204    238    237    267    262    278    293    283    295    282    283    274
2           11     26     39     61     90    111    127    143    149    157    152    163    169    176
3            0      1      6     12     19     27     37     53     68     80    106    118    122    132
4            0      0      0      1      4      5     12     13     16     24     25     36     48     46
5            0      0      0      0      2      2      5      5      7     11     13     14     18     20
6            0      0      0      0      0      0      0      0      1      2      3      4      4      6
7            0      0      0      0      0      0      0      0      0      0      0      0      1      1
8            0      0      0      0      0      0      0      0      0      0      0      0      0      0
9            0      0      0      0      0      0      0      0      0      0      0      0      0      0
```

After all animals have been added and the above table printed, your program should do the following

- print the hash code and index for Red-Bellied Black Snake
- print the contents of the list (**Animal** objects) at index 851 in the hash table
- print all the indexes where the chain length is 5
- find and print the hash table index and value (**Animal** object) for the following keys:
    - Brown Rat
    - Marsh Deer
- delete all the records from the hash table designated by the following keys and print the index and **AnimalList** at that index after deletion:
    - Sunda Flying Lemur
    - Lesser Flamingo

General

Write methods where you can. Include comments for methods or code that need explanation.

Although the methods are all fairly simple, it is very easy to make mistakes by confusing hash codes, hash functions, keys, indexes, nodes, chains. Be very careful when coding.

IMPORTANT: You cannot use any global variables. All variables in Hashing must be declared inside methods and be passed when necessary. Methods in Hashing will be static.

IMPORTANT: Except for arrays, do not use any existing data structures from the Collections Framework in Java. You must create your own.

Using a 2 Dimensional Array

Think of this as a matrix of rows and columns. Or an array of arrays.

You access the entry at row i and column j using `frequencies[i][j]`

You can iterate through one row or one column or the whole matrix (the last is done using a nested for loop).

Turning in the Assignment

Take screenshots of your results. Upload the screenshots, java files and class files by the start of class on April 4. You have two weeks to work on this assignment and it is worth ~~24~~ 20 points.