Kang-hee Cho

CSCI 323 Hashing Assignment

**\*\*\*IMPORTANT\*\*\***

Files included inside submission:

hash\_element.java: my own implementation of the hash element linked list class

hash\_table.java: my own implementation of the hash table class

hash\_main.java: the main class: includes the main function for testing

to run my code:

if using a mac: get onto terminal and into the directory these files are located in

javac hash\_main.java

java hash\_main

if using a windows machine:

you should run it on a text editor/eclipse

1. I used the division method for hashing for the implementation of my hash function.
2. With mu = 0.8, we assume the table’s length is equivalent to 80% of the number of keys we plan to store in the hash table. Therefore, the hash table size for mu = 0.8 was 80.

For this input, using my hash function, we got 70 collisions. Although this seemed abnormal, remember that we are using the method described in pg 263 of the textbook to get the ASCII value for each word (key) value we are putting into our hash table. Since most of these words have common letters, it is clear to see why so many collisions happened.

1. With mu = 0.5, we assume the table’s length is 50% of the number of keys we plan to store in the table, which is size = 50.

For this table size, we got 72 collisions.

1. Output:

Check the file I submitted. I included screenshots in the file in the folder names:

Mu= 0.8

Mu= 0.5

LookupTests

1. I have tested the lookups for 50 random keys for the two different hash tables fairly. I tested 50 random lookups for each table 100 times and averaged the values. Unexpectedly, we ran into 34 average collisions during lookup for each table.

I have put the screenshots of this output and my test code inside the folder called LookupTests.