Conrad Horton
CSC382 - September 2016
9-25-2016
Hash Table

My hash table maintains an array of Linked List references where each Linked List is assigned to certain element by being "hashed". The hash method takes the Node's key value (a string assigned by the user), and calculates its integer value then mods (modulo) it by the max length of the table. The user can retrieve data by key, find the length of the table, and get the current number of Nodes in all Linked Lists. Furthermore the user may insert new Nodes, delete Nodes, print the whole table, and even print a histogram of the table's data distribution.

Public Methods:

HashTable(int tableLength) - Creates an instance of a HashTable. The max length of the table is that of the value passed in by the user.
Node * retrieve (string key) - Returns a Node with the same key as that which was passed in.
int getLength() - Returns the max length of the table.
int getNumberOfNodes() - Returns the sum of all Nodes in each table element's Linked List.
void insert(Node * newNode) - Inserts the passed in Node into one of the Linked Lists depending on its hash value. If there is no Linked List in that element, this will be the root Node of a new Linked List.
bool removeNode(string key) - Removes the Node associated to the passed in key.
void printHistogram() - Prints a histogram of the table's data distribution.
void printTable() - Prints each element's Linked Listand their contents.

```
#include <iostream>
#include "HashTable.h"

int main()
  {
  Node* a = new Node {"Abe Lincoln",             "555-856-5555", NULL};
  Node* b = new Node {"Barry White",             "555-954-5568", NULL};
  Node* c = new Node {"Charles Manson",          "555-887-9956", NULL};
  Node* d = new Node {"Doc Brown",               "555-654-1234", NULL};
  Node* e = new Node {"Eli Manning",             "555-987-4567", NULL};
  Node* f = new Node {"Frank Zappa",             "555-888-7898", NULL};
  Node* g = new Node {"Gregg Ory",               "555-474-4747", NULL};
  Node* h = new Node {"House Brick",             "555-654-1234", NULL};
  Node* i = new Node {"Inny Outy",               "555-898-1251", NULL};
  Node* j = new Node {"Jay Sahn",                "555-639-9636", NULL};
  Node* k = new Node {"Killer Tofu",             "555-147-1478", NULL};
  Node* l = new Node {"Lemmi Kilmister",         "555-258-2589", NULL};
  Node* m = new Node {"Mac N. Cheese",           "555-852-8520", NULL};
  Node* n = new Node {"Noodle Hut",              "555-848-8484", NULL};
  Node* o = new Node {"Open Says Me",            "555-484-4848", NULL};
  Node* p = new Node {"Pottery Shed",            "555-656-4564", NULL};
  Node* q = new Node {"Q",                       "555-654-1234", NULL};
  Node* r = new Node {"Ringo Starr",             "555-987-9878", NULL};
  Node* s = new Node {"Snakes and More",         "555-000-0000", NULL};
  Node* t = new Node {"Tiny Tim's Crutch Hutch", "555-000-0001", NULL};
  Node* u = new Node {"Und",                     "555-000-0002", NULL};
  Node* v = new Node {"Volcano Tom's",           "555-000-0003", NULL};
  Node* w = new Node {"Why",                     "555-000-1010", NULL};
  Node* x = new Node {"XXX",                     "555-101-1100", NULL};
  Node* y = new Node {"Yellow Submarine",        "555-111-1011", NULL};
  Node* z = new Node {"Zed Zed Top",             "555-111-2135", NULL};

  HashTable table;

  table.insert(a);
  table.insert(b);
  table.insert(c);
  table.insert(d);
  table.insert(e);
  table.insert(f);
  table.insert(g);
  table.insert(h);
  table.insert(i);
  table.insert(j);
  table.insert(k);
  table.insert(l);
  table.insert(m);
  table.insert(n);
  table.insert(o);
  table.insert(p);
  table.insert(q);
  table.insert(r);
  table.insert(s);
  table.insert(t);
```

```
table.insert(u);
table.insert(v);
table.insert(w);
table.insert(x);
table.insert(y);
table.insert(z);
table.printTable();
table.printHistogram();

Node* result = table.retrieve("Lemmi Kilmister");
cout << "Name: " << result->key << "\tTN: " << result->data << endl;

system("pause");
return 0;
}
```