CS163 Lab Session #4 – Hash Tables

Please complete this to become familiar with Hash Tables.
Submit code to the CS199 D2L dropbox. Limit the time invested to 1 hour and 50 minutes maximum.

Arrays of Linked Lists: The purpose of hash tables is to allow for direct access without actually knowing the position of the data unlike ordered lists. We use a **combination of arrays and linked lists**. A hash function will let us know what element of the array to access and then all items that collide with that index are stored at the beginning of a "chain" which is a linear linked list. *This lab will have us practice building a hash table using arrays of linear linked lists.*

Getting Set Up: If you have questions with any of these, post them on D2L, the scribblar link for Lab #2 or contact karlaf@cs.pdx.edu:

_____1. Since the data structure still uses linear linked lists, we will continue using nodes. Assume that we will be storing a journal entry in each linear linked list's node, create the **node** structure:

```
struct node {
  char * title;
  char * information;
  node * next;
}
```

_____2. **Next,** the hash table itself is an array of head pointers. Create that array, statically allocated of size 101:

```
node head[101];
```

_____3. What is the underlying type of each element? node

_____4. Now, create a dynamically allocated array of N elements of that type:

```
node * head = new node[N];
```

_____5. Before we can begin using the data structure, **what value** should each head be? NULL;

a. When working with arrays of size N, indices range from 0 to N-1.

b. Write the code for a **constructor** to initialize this data structure. Watch your boundary conditions.

```
table::table(void)
{
  node * head = new node[N];

  for (int i = 0; i < (N-1); ++i)
```

Coding: *Now that we are set up, we will be working with an existing class implementing a table ADT for a journal. The data structure is an* ***array of linked lists*** *of journal entries (each array has at most 5 entries). You have access to the .h class interface in D2L's online "locker" to see what data members and member functions are available. Your job will be to implement functions to experience manipulating the hash table.*

Develop an ADT: Create a stack for your journal entries. Pushing, popping, and peeking journal entries

_____Step 1. Design a hash function and compare with other students in the Lab

_____Step 2. Begin implementing the member functions, in a .cpp file and upload these to D2L's CS199 dropbox:

a. Constructor table();
b. The insert function (insert at the head of a linear linked list)
c. The retrieve function (to find an item by title)

_____Step 3. Compile and run:

a. Modify main to call your functions (or double check main is correct for your implementation)
b. Download the .h and .o files from D2L's online locker
c. Compile: g++ *.cpp *.o
d. Run: ./a.out

_____Step 4. Now think about what the destructor should be like and write the algorithm:

_____Step 5. **Develop the test plan:** *For each member function that you plan to write, think about how to test it – what flow of control exists in the member function and how would you test out all conditions:*

Test Case(s)	Expected Result	Verified? (yes/no)
Enter no items, try to find an item	No result found	Yes
Enter no items, do nothing	Nothing	Yes
Enter 1 entry, try to find a different item	No result found	Yes
Enter 1 entry, try to find that item	Display result	Yes
Enter 2 entries, try to find both	Display both of the entries	Yes

Verify correctness: Using the above test plan, create a test program that tests the interactions of all functions together.

Self-Assessment: *What could you do to improve for next time?*

I would work to make my hash function more unique in the way that it approaches data