**Programming Assignment #3**

**CS 260 Data Structures**

**Please follow submission instructions in style sheet.**

**Background:** As we begin the next phase of CS260, we will be working with “Table Abstractions”. A Table ADT allows us to search on the “value” of the data without requiring a particular location to be known. It allows us to store the data using non-linear techniques. A hash table comes to mind as a good alternative for programs that require working by “value” but that don’t require that the data be sorted.

**Goal:** The goal of the third program is to create a **hash table using chaining**. Hash tables are very useful in situations where an individual wants to quickly find their data by the “value” or “search key”. The client program uses a “key” instead of an “index” to get to the data. The key is then mapped through a **hash function** which supplies an index!

**Specifics:** Have you ever wanted to find information on the web just to have to go through so many websites and posts that were irrelevant. Once you find a great website with valuable (and correct) information, making it a favorite is nice but at some point how many favorites do you have and it becomes difficult to filter through. In program#3, you are going to have the client program search by topic (e.g., “Hash Functions” or “Finance”) and retrieve all of the website information that you have collected about that topic.

Part I: The Table ADT

The Table ADT is implemented using a hash table with chaining (array of linked lists).

For each website, keep track of the following information

1. Topic name (e.g., “Data Structures”)
2. Website address

(e.g., <https://www.pearson.com/us/higher-education/product/Carrano-Data-Abstraction-Problem-Solving-with-C-5th-Edition/9780321433329.html>)

1. Summary of what you can find at this address (e.g., “The classic, best-selling Data Abstraction and Problem Solving with **C++**: Walls and Mirrors book provides a firm foundation in data structures”)
2. Review (e.g., your thoughts about how helpful this site is)
3. Rating (1-5 stars – 1 being not very useful, 5 being very useful)

**The required functions for your Table ADT are:**

**Constructor**

**Destructor** *(deallocate the hash table)*

**Insert a new website by topic** *(add website’s information into the hash table)*

**Retrieve** *(retrieve all websites based on the topic keyword supplied)*

retrieve will need an array of websites and the number of matchings as arguments for the ADT to fill. It returns a success/fail flag for whether or not it found a match. Retrieve’s arguments might be something like:

**bool retrieve(char \* topic\_keyword, website all\_matches[], int & num\_found);**

Retrieve should not correspond with the user (i.e., it should not prompt, echo, input, or output data).

**Edit** (*modify the review and rating for a particular topic and website match)*

**Remove** *(remove all websites with a 1 star rating)*

**Display** *(only displaying matches, based on the topic keyword)*

**Display all**

**Monitor the performance of hashing function by displaying each chain length**

Part II: The driver or the test program

The test program needs to first load the test data set from external file at the beginning of the program.

The test program needs to allow user to add a new website, retrieve all websites for a given topic, edit a website to modify the review and rating, remove all websites with a 1 star rating, and display all matches for a given topic, and then a display all (not in order!). It should also allow user to monitor the performance of the hash table (the monitor function in the ADT)

The menu-based user interface should allow user to use/test ALL the functionalities of the program. Try to make the user interface easier to use.

1. Always prompt user when you need input data.
2. The prompt needs to be meaningful. Example works great. E.g. “Enter the rating (e.g 1-5): “
3. When asking user to choose some existing data, index works great. You can display the data with index preceding each one first.

**Things you should know...as part of your program:**

1. Do not use statically allocated arrays in your classes or structures. All memory must be dynamically allocated and kept to a minimum!
2. All data members in a class must be private
3. Never perform input operations from your class in CS260
4. Global variables are not allowed in CS260

**5)** **Do not use the String class! (use arrays of characters instead and the cstring library!)**

**6)** Use modular design, separating the .h files from the .cpp files. Remember, .h files should contain the class header and any necessary prototypes. The .cpp files should contain function definitions. You must have at least 1 .h file and 2 .cpp files. **Never "#include" .cpp files!**

**7)** Use the iostream library for all I/O; do not use stdio.h.

**8)** Make sure to define a constructor and destructor for your class. Your destructor must deallocate all dynamically allocated memory.