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.

For each topic, 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)
3. 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")
4. Review (e.g., your thoughts about how helpful this site is)
5. Rating (1-5 stars – 1 being not very useful, 5 being very useful)

Data Structures: Write a C++ program that implements and uses a **<u>table</u> abstract data type using a hash table (with chaining)** to add a new website's information and your review, 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!).

a. This means that retrieve will need an array of websites as an argument (you may use a class or struct for an individual website) for the ADT to fill. Retrieve needs to also know how many at most it can find (and integer argument) and return 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);

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

Evaluate the performance of storing and retrieving items from this table. Monitor the number of collisions that occur for a given set of data that you select. Make sure your hash table's size is a prime number. Try different table sizes, and evaluate the performance (i.e., the length of the chains!). **Your efficiency writeup must discuss what you have discovered.**

In summary, the required functions for your TABLE ADT are:

1. **Constructor**
2. **Destructor** *(deallocate the hash table)*
3. **Insert a new website by topic** *(add website's information into the hash table)*
4. **Retrieve** *(retrieve all websites based on the topic keyword supplied)*
5. **Edit** (*modify the review and rating for a particular topic and website match)*
6. **Remove** *(remove all websites with a 1 star rating)*
7. **Display** *(only displaying matches, based on the topic keyword)*
8. **Display all**

The data must originally be retrieved from an external data file. This is required for this assignment. In order to properly evaluate the hash function chosen, we need a large data set! **However:** your program will not be removing or modifying the data from the file.

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.