CSC 2110, Spring 2013
Programming Assignment 2
Assign date: February 20, 2013, Due: March 8, 2013 at class

Skills you will learn

- Creating a templated linked list class
- Building a list
- Searching a list
- Deleting items from a list
- Adding items to a list

Description

From the ACM website:
ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and a profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

You should check out the ACM when you get a chance. You can become a member of the national organization, and/or become a member of our local chapters. The ACM's best known journal is ***Communications of the ACM***, or CACM for short. For this assignment, you will create an in-memory database for some of the articles appearing in CACM since 2000. I have provided you sample text files that contain lines of text with a string key, the name of the author that wrote the article, and the title of the article. The key is a simple unique string that identifies the article. Your program should begin by reading the text file and creating a linked list of articles. Once the list is created, your program will then prompt the user with a menu of operations. The menu will include operations for displaying a particular article, adding a new article, removing an article, printing the entire list of articles, or exiting. Here is a sample execution:

```
$ ./cacmLibrary cacmpub.txt
Loading library, please wait...
Welcome to the CACM Libray!


What would you like to do?

(F)ind an article
(L)ist all articles
(A)dd a new article
(R)emove an existing article
(E)xit

Your choice>F
Please enter a search key: Meeks:2000:EFBa
```

```
------------------------------------------------------------
- Record FOUND:
-
- Key: Meeks:2000:EFBa
- Author: Brock N. Meeks
- Title: Electronic frontier: bugging out: Y2K fallout or business as usual?
------------------------------------------------------------

What would you like to do?

(F)ind an article
(L)ist all articles
(A)dd a new article
(R)emove an existing article
(E)xit

Your choice>L
------------------------------------------------------------
- Key: AAgren:2001:ODE
- Author: Per-Olof AAgren
- Title: Is online democracy in the EU for professionals only?
------------------------------------------------------------
------------------------------------------------------------
- Key: AAstrom:2001:SDO
- Author: Joachim AAstr"om
- Title: Should democracy online be quick, strong, or thin?
------------------------------------------------------------


...


------------------------------------------------------------
- Key: vonAhn:2008:DGP
- Author: Luis von Ahn and Laura Dabbish
- Title: Designing games with a purpose
------------------------------------------------------------
------------------------------------------------------------
- Key: vonLohmann:2004:VVC
- Author: Fred von Lohmann
- Title: Viewpoint: Voluntary collective licensing for music file sharing
------------------------------------------------------------
2502 records shown.

What would you like to do?

(F)ind an article
(L)ist all articles
(A)dd a new article
(R)emove an existing article
(E)xit

Your choice>F
Please enter a search key: asdf adsfhiuaa afsdnasd asdf
-----------------------------------------------------------
-   Sorry, but there are no records matching your query   -
-----------------------------------------------------------
```

```
What would you like to do?

(F)ind an article
(L)ist all articles
(A)dd a new article
(R)emove an existing article
(E)xit

Your choice>A
Please enter the key for your new article: Crawford:2000:EPa
This key already exists!

What would you like to do?

(F)ind an article
(L)ist all articles
(A)dd a new article
(R)emove an existing article
(E)xit

Your choice>A
Please enter the key for your new article: JoeSchmoe:2008
Enter the author's name: Joe Schmoe
Enter the title of the article: Joe Schmoe Rules!
------------------------------------------------------------
- The following record has been added:
-
- Key: JoeSchmoe:2008
- Author: Joe Schmoe
- Title: Joe Schmoe Rules!
------------------------------------------------------------

What would you like to do?

(F)ind an article
(L)ist all articles
(A)dd a new article
(R)emove an existing article
(E)xit

Your choice>R
Please enter the key of the item you wish to remove: asdf sdaf asdf
sadfadsfsd
Article with key asdf sdaf asdf sadfadsfsd not found.

What would you like to do?

(F)ind an article
(L)ist all articles
(A)dd a new article
(R)emove an existing article
(E)xit

Your choice>R
Please enter the key of the item you wish to remove: Lucas:2000:VWB
```

```
------------------------------------------------------------
- The following record has been REMOVED:

-
- Key: Lucas:2000:VWB
------------------------------------------------------------

What would you like to do?

(F)ind an article
(L)ist all articles
(A)dd a new article
(R)emove an existing article
(E)xit

Your choice>E
Thank you for using the CACM Library!
```

If you look carefully at the output above, you will notice that the file is completely loaded when the program starts up. In other words, your program should read the file and create a linked list of all the articles immediately before displaying the menu. Reading the file is simple. Each article consists of three lines. The first line is the unique key, the second line is the author's name, and the third line is the title of the article. To read all articles in the file, simply read each triplet of lines, insert a new node representing an article in your list, and then iterate (loop again). Your program should insert the article into the linked list in such a way that the linked list is in sorted order. Here is some pseudo-code for inserting in sorted order:

```
      method to insert article node n into list:
        find the first node x in list such that x > n
        insert n into list before x
```

After the linked list is constructed, your program should print out the menu and wait for the user's input. Selecting 'F' on the menu (hitting 'F' on the keyboard and then <Enter>) results in the user being prompted for a key. The key is a string of printable ASCII characters. Once the user enters a key, your program should search for the key in your linked list, and, once the key is found, prints out the corresponding key, title, and author. If the key is not found, your program should give the appropriate message.

Selecting 'L' lists all articles in your linked list. In other words, your program will iterate through the linked list nodes, printing out the key, title, and author in each node. After all articles in the linked list have been printed to the screen, your program should display the total number of records shown.

Selecting 'A' adds a new article to the list. To add a new article, your program must first prompt the user for the key, title, and author. Note that after the user enters the key, your program should check to see if that key already exists and, if so, inform the user that that key already exists. Once the user enters the article's data, your program should put the item into the list such that the list remains in sorted order.

Selecting 'R' removes an existing article from the list. To remove an article, your program must first prompt the user for the key. Once the key is entered, your program searches for the key, and if found, removes that node from the list and de-allocates its memory. If the key is not found in the list, your program should print out the corresponding message that the article was not found.

Lastly, selecting 'E' exits the program.

Implementation:

Part I

You must implement a templated linked list class. Your linked list class should have, *at least*, the following methods:

`template<class T>` `LinkedList<T>::LinkedList()`	A constructor that initializes the empty list.
`template<class T>` `LinkedList<T>::~LinkedList()`	A destructor that deallocates all memory occupied by list elements.
`template<class T>` `void LinkedList<T>::PutItem(T item)`	Inserts an element into the list in sorter order.
`template<class T>` `void LinkedList<T>::DeleteItem(T item)`	Removes an element from the list and deallocates its memory. List remains in sorted order after the item has been deleted.
`template<class T>` `T LinkedList<T>::GetItem(T& item, bool& found)`	Finds an element in the list with a key matching the key in item. If the item is found, a copy of the item is returned and found is set to true. Found is false otherwise.

Although these are the only required member functions of our LinkedList class, it may be wise to write different functions to simplify other LinkedList tasks. The choice and implementation of any other functions is left up to you.

As we have seen in our lectures, the LinkedList class will be composed of NodeType objects. Therefore, our NodeType struct must also be templated to allow the info component to be changed with each declaration of a list containing a different data type.

After you have finished implementing your templated LinkedList class, test the creation of lists of different types in a driver file named listTestDriver.cpp. In this driver file, you can create LinkedLists of different types as follows:

```
LinkedList<int> intList;            // creates a list of integers
LinkedList<double> doubleList;      // creates a list of doubles
```

```
LinkedList<string> stringList;          // creates a list of strings
```

Test adding and removing items from these lists until you are confident that you have a valid implementation of a templated LinkedList before you move onto the next part.

Part II

Now that we have a working LinkedList class, we are going to create a new object that we can use to store the Article information for each article in our CACM database. We will in turn create a LinkedList of our Article objects to be used as our database. The Article class should have three private data members (one to store the article's key, one to store the author's name, and one to store the article's title). All of these data members should be of type string. Also, the Article class should have various public member functions. Our Article class needs functions to create new instances of Article objects (constructors) as well as functions to get and set the private data members of the class (e.g. getKey() and setKey(string k)).

More importantly, we need a way to compare our Article objects. Therefore we must overload the comparison operators > and ==. Although you are welcome to overload any other comparison operators that you like, for the operations of a LinkedList these two should be sufficient. These comparison operators should compare the values of two Article object's key data members and return the result. For example, say we have two Article objects a1 and a2 with keys "abcd" and "wxyz" respectively. The comparison a1 > a2 will result in a false and the comparison a2 > a1 will result in a true.

After our definition and implementation of the Article class is complete, a list of Article objects should be used to store all information read in from our input file. Each entry in the input file should be used to create a new Article object that will be inserted into a LinkedList of type Article. Declaring such a linked list can be done as following in another driver program called cacm.cpp:

```
LinkedList<Article> cacmDB;
```

After we have a LinkedList of Article objects, the tasks described above and provided by the menu must be implemented using said LinkedList.

Deliverables:

- You must submit a README file containing information on how to compile, run, and use your program.
- You need to zip all your source code (all header and cpp files only, no executable or data files) and submit it to iLearn. The name of your zip file should be yourlastname_prog2_2110.zip (for example my filename would be mcdaniel_prog2_2110.zip).

Your program will not be graded if it does not compile

Grading:

This assignment is worth 100 points, distributed as follows:

Functionality	**80 pts**
(program works correctly, and meets the specifications)	
Templated LinkedList class Implementation	15 pts
Implementation of PutItem function	10 pts
Implementation of overloaded > and == function	10 pts
Implementation of DeleteItem function	10 pts
Implementation of GetItem functions	10 pts
Implementation of LinkedList Destructor	5 pts
Populating the list correctly	5 pts
Implementation of menu choices	15 pts
Program Style	**10 pts**
Meaningful variable names	2 pts
Proper indentation	2 pts
Correct program header	2 pts
Comments throughout the program	4 pts
README file	**5 pts**

Late submission:

There will be a 5% penalty/day for up to 2 days. For delays more than two days a score of zero will be assigned.

*** Start working on your assignment early; and don’t hesitate to contact me or the TA if you need help.