CS216: Introduction to Software Engineering Techniques (Fall, 2018)
Lab Assignment 10
(20 points)

Today's Date: Monday, November 05

Due date: Friday, November 09

The purpose of this lab assignment is

- To work on the definition of class named `Autocomplete`
- To review binary search algorithm and a few sorting algorithms
- To get to know generic programming using templates
- To create your own makefile for building executable program for Lab10
- To work on the second part of your Project Assignment 2

Part 1:

1. In the terminal window, make the `CS216` directory, which you created in `Lab1`, your current working directory.

2. Create a directory underneath the `CS216` directory named `Lab10`, and make the `Lab10` directory, your current working directory.

3. Use command curl to download a zip file named `Lab10source.zip` from the link (http://www.cs.uky.edu/~yipike/CS216/Lab10source.zip) and save the file into your current working directory **`~/CS216/Lab10`**:

```
$ curl –O http://www.cs.uky.edu/~yipike/CS216/Lab10source.zip
```

4. Unzip the file you downloaded from step 3 using the command:

```
$ unzip Lab10source.zip
```

The zip file contains SIX files: actors.txt, autocomplete.h, autocomplete.cpp, SortingList.h, SortingList.cpp, and Lab10.cpp. Please note that the definition of the class, named `SortingList`, is exactly the same as that of `Project 2`; and the declaration of the class, named `Autocomplete`, is exactly the same as that of `Project 2`, however, the implementation of one member function of class `Autocomplete`, named Search(), is slightly different from Project 2: instead of generating the first and last index numbers of all prefix-matched Term objects in the sequence as in Project 2, this member function in Lab 10 only generates at maximum THREE prefix-matched Term objects, by trying to match the one index number lower than, and one index number higher than, the returned index number of calling binary_search() member function. If either of them prefix-matches the user input, then add it/them to the list to display. Other than this one member function (which you need to modify to fit the Project 2 requirement), all other member function implementations of this class can be directly used (without modification) for Project 2.

5. Copy the definition of the class named `Term`, which you have finished in Lab9, to your current working directory:

```
$ cp ~/CS216/Lab9/term.h ./

$ cp ~/CS216/Lab9/term.cpp ./
```

6. Open `autocomplete.cpp with` your preferred text editor and take a look at the description of each function and provide the implementation of each member function. In this Lab assignment, beside providing the definition of the `Autocomplete` class in **`autocomplete.cpp`**, you also need to provide the implementation of three functions: `operator []` overloading, `selection_sort()` member function and `bubble_sort()` member function, in **`SortingList.cpp`**. (Hints: the implementation should be quite similar to that of TermSortingList class in Lab9, however you do need to modify it to a template class).**(Please do not modify Lab10.cpp!)**

After you finished `autocomplete.cpp`, and `SortingList.cpp`, compile the source files using the command:

```
$ g++ term.cpp SortingList.cpp Lab10.cpp -o Lab10
```

7. Write your `makefile` for `Lab 10` to help you efficiently generate the executable program. Each time after you modify some code, save the file and run `make`, you may need to fix some errors and run make again. make will help you efficiently rebuild the program every time you make a change.

The following are some examples of running your program named `Lab10`:

```
$ ./Lab10

Usage: ./Lab10 <filename>

$ ./Lab10 actors.txt

Time for sorting all terms: 0.356714 seconds.
Please input the search query (type "exit" to quit):
Tom H↵
Time for searching the maximum three of matched terms: 0.124783 seconds.
Data itmes in the list:
1351430588    Tom Hollander
342551365     Tom Hopkins (VII)
38189270      Tom Holland (X)

Please input the search query (type "exit" to quit):
```

```
Zv↵
Time for searching the maximum three of matched terms: 0.156479 seconds.
Data itmes in the list:
79711678        Zviad Sokhadze
54617761        Zvonimir Hace

Please input the search query (type "exit" to quit):
Emma↵
Time for searching the maximum three of matched terms: 0.119014 seconds.
Data itmes in the list:
148775460       Emma Dukes
127509329       Emma Degerstedt
30363732        Emma Dewhurst (I)

Please input the search query (type "exit" to quit):
Z↵
Time for searching the maximum three of matched terms: 0.134753 seconds.
Data itmes in the list:
114274386       Zachary Chitwood
40073766        Zachary Browne
30651422        Zachary Boyt

Please input the search query (type "exit" to quit):
Zvon↵
Time for searching the maximum three of matched terms: 0.285376 seconds.
Data itmes in the list:
54617761        Zvonimir Hace

Please input the search query (type "exit" to quit):
Aa↵
Time for searching the maximum three of matched terms: 0.196584 seconds.
Data itmes in the list:
886365780       Aaron Pearl
247207184       Aaron Paul (I)
56800325        Aaron Patrick Freeman

Please input the search query (type "exit" to quit):
Aak↵
Time for searching the maximum three of matched terms: 0.401731 seconds.
```

```
Data itmes in the list:
168336709     Aakomon Jones

Please input the search query (type "exit" to quit):
Yi↵
Time for searching the maximum three of matched terms: 0.14446 seconds.
Data itmes in the list:
176663527     Yi Zhao (I)
128067808     Yi Shih
58183966      Yi Lu Wei

Please input the search query (type "exit" to quit):
Yi Pike↵
Time for searching the maximum three of matched terms: 0.198547 seconds.
No matched query!
Please input the search query (type "exit" to quit):
exit↵
```

Note that the blue part is what you type from the keyboard, ↵ represents the "return" key. Please note that the time measurement in the above sample output may not match your output.

8. Then zip together: **`makefile`**, **`term.h`**, **`term.cpp`**, **`autocomplete.h`**, **`autocomplete.cpp`**, **`SortingList.h`**, **`SortingList.cpp`**, **`actors.txt`**, and **`Lab10.cpp`** into one file named **`Lab10.zip`**. (Note your TA will use your makefile to build your program)

Submission

Open the link to Canvas LMS (https://uk.instructure.com/), and log in to your account using your linkblue user id and password. Please submit your file (Lab10.zip) through the submission link for "Lab 10".

Grading (20 points + Bonus 3 points)

1. Attend the lab session or have a documented excused absence. (5 points)
2. You create a correct makefile. (1 point)
3. Your program correctly solves the problem.
   - The implementation of FOUR member functions of Autocomplete class are correct.
     - `insert()` member function is correct in `autocomplete.cpp` (2 points)
     - `sort()` member function is correct in `autocomplete.cpp` (2 points)
     - `binary_searchHelper()` function is correct in `autocomplete.cpp` (2 points)
     - `search()` member function is correct in `autocomplete.cpp` (2 points)
   - The implementation of three member functions of SortingList class are correct. (3 points* 2 = 6 points)

Bonus: Demonstrate your program (including to build your executable program using your own makefile) to your TA and answer TA's questions. (3 points)

(Late assignment will be reduced 10% for each day that is late. The assignment will not be graded (you will receive zero) if it is more than 3 days late. Note that a weekend counts just as regular days. For example, if an assignment is due Friday and is turned in Monday, it is 3 days late.)

Enjoy programming... with an awesome feeling of accomplishment...