CSCE 221 Assignment 3 Cover Page

First Name　　Last Name　　UIN

User Name　　E-mail address

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People				
Web pages (provide URL)				
Printed material				
Other Sources				

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.
On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Your Name　　Date

CSCE 221 Assignment 3

September 27, 2013

Due Oct 8 at 11:59pm

Hardcopy Report: Turn in to your TA in labs that week

Demonstration of Part I on Sep 30 and Oct 1

Objective

This is an individual assignment which has two parts.

1. Part 1 involves implementing a simple Doubly Linked List and a Doubly Linked List template ADT, and analyzing the complexity of your implementation.
2. Part 2 involves writing applications of Linked List and writing a report.

Part I: Implementing Doubly Linked List

- **Program Instructions**

Download the program `221-A3-13c-code.tar` from the course website. Use the 7-zip software to extract the files in Windows, or use the following command in Linux.

```
tar xfv 221-A3-13c-code.tar
```

Three programs in separate folders are included.

1. "Simple Doubly Linked List"for integers
   (a) Contains a list node sturcture and associated functions. Doubly linked lists of integers can be constructured using just the structure of list node.
   (b) You need to complete the following functions in the `SimpleDoublyLinkedList.cpp`.
      i. `insert_before`
      ii. `insert_after`
      iii. `delete_before`
      iv. `delete_after`

      The above functions insert an integer or remove a node around the current list node.
   (c) Type the following commands to compile the program.

   ```
   make clean
   make
   ```

   (d) The main program includes an example of contructing a doubly linked list, and demonstrates how to use it and display it. Type the following command to execute.

```
./SimpleDoublyLinkedList
```

2. "Doubly Linked List"for integers

   (a) Most code is extracted from the lecture slides. An exception structure is defined to complete the program.

   (b) You need to complete the following functions in the `DoublyLinkedList.cpp`.

      i. copy constructor
      ii. assignment operator
      iii. output operator

      Make sure the i. and ii. functions do a deep copy of the input list, that is, copying each node one by one.

   (c) Type the following commands to compile the program.

   ```
   make clean
   make
   ```

   (d) The main program includes examples of creating doubly linked lists, and demonstrates how to use them. Type the following command to execute.

   ```
   ./Main
   ```

3. Template "Doubly Linked List"for general type

   (a) Convert the doubly linked list in program 2 to a template, so it creates lists of general types other than integer.

   (b) Read C++ slides, page 16-22 at http://www.stroustrup.com/Programming/19_vector.ppt

   (c) Follow the instructions below:

      i. Templates should be defined in a `.h` file. Move the content of `DoublyLinkedList.cpp` and `DoublyLinkedList.h` to `TemplateDoublyLinkedList.h`
      ii. Add `template <typename T>` before each class and member function, so the general type `T` can be used in the class and function.
      iii. In each member function signature, replace `DoublyLinkedList::` by `DoublyLinkedList<T>::`
      iv. Replace `int` by general type `T` except for the `count` variable.
      v. Replace member variable declaration, input type and output type `DListNode` by `DListNode<T>`
      vi. Replace variable declaration, input type and output type `DoublyLinkedList` by `DoublyLinkedList<T>`, including the friend class declaration
      vii. Set initial value of `T obj` to `T()`

   (d) Type the following commands to compile the program.

   ```
   make clean
   make
   ```

   (e) The main program includes examples of creating doubly linked lists of "strings", and demonstrates how to use them. Type the following command to execute.

   ```
   ./TemplateMain
   ```

- **Complexity Analysis**

  Comment each class member function you implemented with its time complexity using big-O notation. Specifically, comment on the loops.

Part II: Application of Doubly Linked List

You will implement a phone book. The phone book stores the last name, first name, 9-digit UIN, one phone number of each student. Students may have the same first name & last name. To speed up the search in the phone book, the data will be stored in a vector of 26 sorted doubly linked list. Each element of the vector, i.e. each doubly linked list, corresponds to an alphabet letter. For example, the first element of the vector, i.e. the first doubly linked list, v[0], corresponds to the letter 'A'. Each doulbly linked list must be maintained in sorted order by last name, first name and UIN. For examples, records "Leyk, Teresa, 123456789", "Lee, Bill, 000000000" and "Lee, Bill, 000000001" should be stored in the linked list at v[11], which corresponds to letter 'L'. In that linked list, Lee people must be stored before Leyk, and "Lee, Bill, 000000000" must be store before "Lee, Bill, 000000001".

Your phone book program will read from a file of student data and provides an interface for users to search in the phone book. The user will be asked to input the last name to search. If the program finds people with the same last name, the user will be asked to input the first name. Again if more than one people have that last and first name, the user will be asked to input the UIN for search. Finally, the program will print the phone number on screen.

- **Program Instructions**

Please create the following files and complete them.

1. `Record.h`: You will create a `Record` class with:
   (a) Member variables for a person's last name, first name, 9-digit UIN, and phone number.
   (b) Overloaded output operator << to print the record on screen
   (c) Overloaded less-than operator `()` to compare to another record by the last name, first name and UIN.

   ```
   bool operator()(const Record& r) {
     /* complete this function */
   }
   ```

   If `this` record is less than `r` in terms of last name, first name and UIN, the function returns TRUE; otherwise, it returns FALSE.

   Please see p.729-738 of Dr. Stroustrup's "Programming Principles and Practice Using C++"

   NOTE: Because the program might ask user to input only partial data, such as only the last name, you should consider the case that the `Record r` object contains empty first name and empty UIN.

2. `PhoneBook.txt`: The file contains unsorted phone data and has the format below. The file will be published on the course website.

   Leyk

   Teresa

   123456789

   9798454456

3. `TemplateDoublyLinkedList.h`: Add a function to the template doubly linked list class:

   `ListNode* insert(const T& obj)`

   The function inserts an object to the correct position assuming the linked list is sorted. After the object is inserted, the linked list should remain sorted. The function should utilize the less-than operator `()` compare `T` objects, assuming that operator `()` is defined for the object `T`.

4. `Main.cpp`:
   (a) Declare a data structure for the phone book

   ```
   vector<TemplateDoublyLinkedList<Record>> phoneBook;
   ```

   (b) Write a display function to dump the whole phone book to screen

```
void display(vector<TemplateDoublyLinkedList<Record>>& v)
```

(c) Implement the search interface as described at the top.

- **Complexity Analysis**

Assume you have `n` input records. The records are distributed evenly into the 26 linked list. Provide a running time function `F()` for the `insert` function. Provide the Big-O notation of the `F()`. Comment beside the `insert` function the Big-O notation of its running time.

Report

Follow the report instructions at http://courses.cs.tamu.edu/teresa/csce221/pdf/CSCE221_Report_Instructions.pdf with extra requirements below:

- In the algorithm description section,
  - briefly describe the 7 functions you implement in Part I and their complexity;
  - and for Part II, describe the less-than operator () and the `insert` function and the compexity of the `insert` function.