CSCE 221 Cover Page
Homework Assignment # **3**

First Name: **Peng** Last Name: **Li** UIN: **822003660**

User Name: **abc** E-mail address: **billlipeng@tamu.edu**

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 in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People	**Peer Teacher**				
Web pages (provide URL)			**http://www.cplusplus.com/reference/vector/vector/**		
Printed material					
Other Sources		**Slides from lecture**			

I certify that I have listed all the sources that I used to develop the solutions/code 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 (signature)  Date **Oct 10, 2014**

Program Description

This lab including two parts: part one involves implementing a simple Doubly Linked List and a Doubly Linked List template ADT, and analyzing the complexity of the implementation; part two involves writing applications of Linked List and writing a report.

Purpose of the Assignment

The purpose of this programming assignment is to learn about doubly linked list data structures and how to transfer a class into a template.

Data Structures Description

We used doubly linked list data structure in this assignment. During the implementation of the doubly linked list, I learned how the doubly linked list structure. For each list node, it is connected to a front node and a back node by using two pointers previous and next. A friend class called DListNode is also implemented, which has three private members, hence T obj, prev pointer and next pointer. The doubly linked list class is a lists that connecting all the DListNodes.

Algorithm Description

i. insert_before O(1): insert a node at the beginning of the doubly linked list. Create a new node; point the next to where the header pointing to and then let the header pointer pointing to it.

ii. insert_after O(1): insert a node at the end of the doubly linked list. Create a new node; point the pervious to where the trailer pointing to and then let the trailer pointer pointing to it.

iii. delete_before O(1): delete a node at the beginning of the doubly linked list.

iv. delete_after O(1): delete a node at the end of the doubly linked list.

v. copy constructor O(n): this function will copy every element in another list.

vi. assignment operator O(n) : the function will clear the linklist first(O(n)). Then copy the whole list from the other list (O (n)). O (n)+ O (n) = O (n)

vii. output operator O(n): this function will print out every nodes content in the linklist.

iix. less-than operator O(1): If this record is less than r in terms of last name, first name and UIN, the function returns TRUE; otherwise, it returns FALSE.

ix. insert function O(n): The function inserts an object to the correct position assuming the linked list is sorted. After the object is inserted, the linked list remains sorted. The function utilizes the less-than operator< compare T objects, assuming that operator< is defined for the object T.

Program Organization and Description of Classes

```
template<class T>
class DListNode{
private: T obj;
    DListNode<T> *prev, *next;
    friend class DoublyLinkedList<T>;
public:
    DListNode(T e=T(), DListNode<T> *p = NULL, DListNode<T> *n = NULL)
    : obj(e), prev(p), next(n) {}
    T getElem() const { return obj; }
    DListNode<T> * getNext() const { return next; }
    DListNode<T> * getPrev() const { return prev; }
};

// doubly linked list
template<class T>
class DoublyLinkedList{
protected: DListNode<T> header, trailer;
public:
    DoublyLinkedList() : header(T()), trailer(T()) // constructor
    { header.next = &trailer; trailer.prev = &header; }
    DoublyLinkedList(const DoublyLinkedList<T>& dll); // copy
constructor
    ~DoublyLinkedList(); // destructor
    DoublyLinkedList<T>& operator=(const DoublyLinkedList<T>& dll);
// assignment operator
    // return the pointer to the first node
    DListNode<T> *getFirst() const { return header.next; }
    // return the pointer to the trailer
    DListNode<T> *getAfterLast() { return &trailer; }
```

```
    // return if the list is empty
    bool isEmpty() const { return header.next == &trailer; }
    T first() const; // return the first object
    T last() const; // return the last object
    void insertFirst(T newobj); // insert to the first of the list
    T removeFirst(); // remove the first node
    void insertLast(T newobj); // insert to the last of the list
    T removeLast(); // remove the last node

    DListNode<T>* insertOrderly(const T& obj);//inserts an object to
the correct position assuming the linked list is sorted
};

class Record {
private:
    string lastName;
    string firstName;
    string uin;
    string phone;
public:
    Record():lastName(""), firstName(""), uin(""),phone("") {}

    Record(string lastName, string firstName, string uin, string
phone)
    :lastName(lastName), firstName(firstName), uin(uin),phone(phone)
{}
    string getLastname() const { return lastName; }
    string getFirstname() const { return firstName; }
    string getUin() const { return uin; }
    string getPhone() const { return phone; }

//    ostream& operator<<(ostream& out, const Record& r);
    bool operator<(const Record& r) const;
};
```

The record class keeps all the information for each person, including last name, first name and UIN and phone number.

Instructions to Compile and Run your Program

$ cd ./part1/SimpleDoublyLinkedList
$ make

```
$ ./SimpleDoublyLinkedList

$ cd    ./part1/DoublyLinkedList
$ make
$ ./Main

$ cd    ./part1/TemplateDoublyLinkedList
$ make
$ ./TemplateMain

$ cd    ./part2
$ make
$ ./a.out
```

Input and Output Specifications

The input format of the phone book should be the same as shown in "PhoneBook.txt".

The first letter of last name and first name should use upper case.

The search keyword is case sensitive.

Logical Exceptions

No logical error has been found in testing.

C++ object oriented or generic programming features

In this lab, both object oriented and generic programming features are used. The classes in this lab are shown in Program Organization and Description of Classes section of this lab report. The generic feature is the doubly linked list template.

Tests

Invalid inputs were tested. People with same last name and first name was tested.