CSCE 221 Cover Page
Homework #3
Due October 13th at midnight to CSNet

First Name Raymond Last Name Zhu UIN 923008555

User Name rawrbyte E-mail address rawrbyte@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. According to the University Regulations, Section 42, scholastic dishonesty are including: acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. Disciplinary actions range from grade penalties to expulsion read more: Aggie Honor System Office

Type of sources			
People	Peer Teachers		
Web pages (provide URL)	Stackoverflow		
Printed material	textbook		
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 Raymond Zhu Date October 12th, 2015

1. Program Description

    (a) This lab includes four folders involving the use of doubly linked lists. A simple doubly linked list which incorporates basic functions of a linked list and a doubly linked list of more complexity which makes use of nodes, pointers (prev, next, trailer, etc), classes, and template classes in addition to user-defined functions and overloading operators. These linked lists are sed to analyze the complexity of implementation of the ADT to applicable data in real-life (phone book records).

2. Purpose of the Assignment

    (a) The purpose of this programming assignment is to learn how to make use and implement doubly linked list ADT into code and how to convert regular classes into templated classes for generic purposes.

3. Data Structures Description

    (a) Doubly linked lists were implemented in this assignment. During the implementation of this abstract data type, the most important aspect that stood out for this ADT to me is the use of two reference pointers to allocate and deallocate memory. For each list node, it is connected to a head and tail by using two pointers pointing to the next and last objects of the list respectively. A friend class called DListNode is implemented with three private members which is made use by the pointers, where a doubly linked list class connects all nodes within DListNode.

4. Alogorithm Description

    - insert_before $O(1)$: insert a node at the beginning of the doubly linked list. Create a new node; point the next node to where the header is pointing to and allowing the header pointer to point to it.
    - insert_after $O(1)$: insert a node at the end of the doubly linked list. Create a new node; point the previous node to where the trailer is pointing to and allowing the trailer to point to it.
    - delete_before $O(1)$: delete a node at the beginning of the doubly linked list.
    - delete_after $O(1)$: delete a node ad the end of the doubly linked list.
    - copy constructor$O(1)$: this function will copy every element in another list.
    - assignment operator $O(1)$: the function will clear the linklist first an then copy the whole list from the other list

- output operator $O(n)$: this function will print out every nodes' content in the linklist
- less-than operator $O(n)$: If the record is less than r in terms of last name, first name, and UIN, the function returns TRUE; otherwise, FALSE.
- insert function $O(n)$: This 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 to compare T objects, assuming that the operator $<$ is defined as a object T in the template class.

5. Program Organization and Description of Classes

```
template<class T> class DoublyLinkedList; // class declaration

// list node
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
  const DListNode<T> *getAfterLast() const { 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;
};

// output operator
ostream& operator<<(ostream& out, const Record& r){
    out << r.getLastname()<< endl << r.getFirstname()<< endl
        << r.getUin() << endl << r.getPhone() << endl;
    return out;
}


bool Record::operator<(const Record& r) const {
    if (this->lastName < r.lastName) return true;
    else if(this->lastName == r.lastName){
        if (this->firstName < r.firstName) return true;
        else if(this->firstName == r.firstName){
            if (this->uin <= r.uin) return true;
            else return false;
        }
        else return false;
    }
    else return false;

}

void read(vector<DoublyLinkedList<Record> >& phoneBook){
    ifstream filein("PhoneBook.txt");
    string lastName , firstName, uin, phone;
    string tmp_lastName, tmp_firstName, tmp_uin;

    if (filein.is_open()){
        while ( !filein.eof()&&(filein >> lastName) && (filein
             >> firstName) && (filein >> uin) && (filein >>
            phone)){
            Record newRecord(lastName, firstName, uin, phone);
            phoneBook[(int) lastName.at(0)-(int) 'A'].
                insertOrderly(newRecord);
        }
    }
    else cout<<"Cannot open PhoneBook.txt."<<endl;
    filein.close();
}

// search for type input
void search(vector<DoublyLinkedList<Record> >& phoneBook,
    vector<Record>& r, string input, string type){
    vector<Record> output;
    if (type == "lastName") { // search for lastname
        int index=(int)input.at(0)-(int) 'A';
        if (index<0 || index > 25)  return;
        DListNode<Record>* tmp= phoneBook[index].getFirst();
        while(tmp!= phoneBook[index].getAfterLast()){
            if(tmp->getElem().getLastname()==input){
                r.push_back(tmp->getElem());
            }
            tmp = tmp-> getNext();
        }
        return;
    }
    else if(type == "firstName"){// search for firstname
        for (int i=0; i<r.size(); i++) {
            if (r[i].getFirstname()==input)
                output.push_back(r[i]);
        }
        r.clear();
        r.resize(0);
        r = output;
        return;
    }
    else if(type == "uin"){// search for uin
        output.clear();
        output.resize(0);
        for (int i=0; i<r.size(); i++) {
            if (r[i].getUin()==input){
                output.push_back(r[i]);
            }
        }
        r.clear();
        r.resize(0);
        r.push_back(output[0]);
        return;
    }
    else return;
}

// print out the menu
void menu(vector<DoublyLinkedList<Record> >& phoneBook){
    vector<Record> result;
    cout << "Please enter the last name :";
    string input;
    cin >> input;
    search(phoneBook,result,input,"lastName");
    if (result.size() > 1) {
        cout << "More than one record was found!"<<endl;
        cout << "Please enter the first name :";
        cin >> input;
        search(phoneBook,result,input,"firstName");
        if (result.size() > 1) {
            cout << "More than one record was found!"<<endl;
            cout << "Please enter the UIN :";
            cin >> input;
            search(phoneBook,result,input,"uin");
            if (result.size() >= 1) {
                cout << result[0]<<endl;
            }
            else    cout << "No record!"<<endl;
        }
        else if(result.size() == 1)  cout << result[0];
        else    cout << "No record!"<<endl;
    }
    else if(result.size() == 1)   cout << result[0];
    else    cout << "No record!"<<endl;
}
// a function dump the whole phone book to screen
void show(vector<DoublyLinkedList<Record> >& phoneBook){
    for (int i=0; i<26; i++)
        cout<<phoneBook[i];
}

int main () {
    vector<DoublyLinkedList<Record> > phoneBook(26);
    read(phoneBook);
    show(phoneBook);
    menu(phoneBook);
    return 0;
}
```

The record class classifies all the information(last name, first name, uin, phone number) for each individual that is taken from the text file database.

The main function calls functions(read(), show(), menu()) with a single argument. From there the void functions implement doubly linked list functions, etc.

6. Instructions to Compile and Run your Program

   - cd SimplyDoublyLinkedList
   - make
   - ./SimplyDoublyLinkedList
   - cd ..
   - cd DoublyLinkedList

- make
- ./Main
- cd ..
- cd TemplateDoublyLinkedList
- make
- ./TemplateMain
- cd ..
- cd Record
- make
- ./a.out

7. Input and Output Specifications

   - The input format of the phonebook should be the same as shown in "phonebook.txt."
   - The first letter of last name and first name should be capitalized.
   - The search keyword is case sensitive

8. Logical Exceptions

   (a) No logical error has been found in testing from the program itself

   (b) During the process of implementing the code, logical errors were discovered that had to be fixed in order for the code to compile and run properly. For example, the implementation of <T> in templated classes within functions and variable calling.

9. C++ object orientated or generic programming features

   (a) In this lab, both object oriented and generic programming features were 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 templated version of the doubly linked list which is also implemented in the phonebook application of doubly linked lists. Within the doubly linked class, there are mutliple functions defined under the class which inherit the functions of the class and this can be observed with the syntax of "::" when the functions are implemented.

10. Tests

    (a) Invalid inputs were tested. People with the same last name and first was tested

```
[rawrbyte]@sun ~/CSCE221/lab3/Record> (11:45:01 10/12/15)
:: ./a.out
Andrews
Edna
528320876
4352514822
Arenas
Edward
239924731
2525976612
Autry
Richard
527646269
6028236739
Latham
Mary
174485583
2156902061
Lawrence
Adelle
606880340
4154068779
Leblanc
Fred
253174074
6782391146
Lee
Rolando
680385098
7752476489
Lester
Mozell
173503264
5706145308
Lile
Francis
237818062
7044943520
Lopez
Felecia
524085043
3036105461
Weaver
Andrew
460497442
2817601558
Whitney
Floyd
367196315
2313697993
```

```
Wilham
Bernice
008641309
8027701514
Wiseman
Kristi
311708896
8122398910
Wiseman
Mary
150521209
7324463703
Wiseman
Mary
224376947
7037213439
Wiseman
Mary
464996747
3253740973
Wiseman
Paul
347075568
2179340158
Please enter the last name :Wiseman
More than one record was found!
Please enter the first name :Mary
More than one record was found!
Please enter the UIN :150521209
Wiseman
Mary
150521209
7324463703

[rawrbyte]@sun ~/CSCE221/lab3/Record> (11:50:31 10/12/15)
:: ./a.out
Please enter the last name :.
No record!

[rawrbyte]@sun ~/CSCE221/lab3/Record> (11:51:21 10/12/15)
:: ./a.out
Please enter the last name :Kyle
No record!

[rawrbyte]@sun ~/CSCE221/lab3/Record> (11:51:57 10/12/15)
:: ./a.out
Please enter the last name :12368916
No record!
```