CSCE 221 Assignment 3

Daniel Bueso-Mendoza 117005010

October 8, 2013

1 Program Description

The program in this assignment was a phonebook that kept a record of various person's information. A person's record involved their last name, first name, UIN, and phone number. Upon executing the application, it will query the user for a person's last name. If the last name matches one in the phonebook then it will notify the user and ask for a first name. If the first name returns a hit then that person's phone number is displayed. There are cases where a certain contact is listed multiple times. For these situations the program asks for the UIN before retuning that contact's phone number.

2 Purpose of the Assignment

This assignment had 2 parts. Part 1 involved implementing a SimpleDoublyLinkedList, DoublyLinkedList, and a TemplateDoublyLinkedList. Part2 utilized the TemplateDoublyLinkedList and a Record class to realize the phonebook program. The overall goal was to become more proficient with doubly linked-lists and template classes.

3 Data Structures Description

The major data structure incorporated in this assignment was a doubly linked-list. Doubly linked-lists are similar to linked-lists in where nodes of a certain type are linked together with a pointer to the next node. The main difference is that doubly linked-lists have a pointer from a node to its previous node.

Initial construction of a doubly linked-list begins by providing header and trailer nodes with "links" between them through the use of pointers. Other member functions involved with a doubly linked-list are a copy constructor, copy assignment, insert first, insert, insert last, remove first, remove last, destructor, first node, last node, list length, and output operator. When a new node is inserted in the list, pointers need to be adjusted carefully. For removal, pointers need to be re-assigned and also a new node object is created which gets assigned to the target node that later is deleted.

4 Algorithm Description

No fancy algorithms were employed in this assignment. Simple while loops were utilized for the more "complex" functions. These loops used a node that starts at the element next to the header and gets "incremented" at the end of the loop. This node iteration continues until the node next to the trailer is found.

In Part 1: the insert and remove element functions all run in $O(1)$ time. This is because elements neighboring the insert/deletion element simply have their pointers re-allocated. Insert before places a new node before the current "this" node. Insert after is where you embed the node after the current node. Delete before removes a node before the "this" node. Delete after erases the node after the current node. These four methods have their pointers re-assigned. In the case of the delete methods a new node is created that gets assigned to the target node for removal.

Copy constructor, copy assignment operator, and ostream operator all perform in $O(n)$ time. This is because you must iterate through the entire list of n elements to copy them over or output their data. Copy constructor iterates through another list and inserts its elements into the current list. Copy assignment operator assigns another list to a temporary object. The current list obtains elements from the temporary object. The ostream operator iterates through an entire list and prints out the data of each node.

In Part 2: the less-than operator () runs in O(1) time and the insert function performs in O(n) time for n input records. The less-than operator begins by completing a comparison of last names from the current Record object and another. The function then compares first names and then UINs. True is returned if the current object has parameters less than the other Record object, else the result is false. The insert function creates a node for a new Record object and an iterating node. The new Record object gets compared with each indexed element from the iterating node to see where it should be inserted. The less-than operator is called for the new node to find its appropriate location based on its data. The function runs in O(n) time because it is directly related to how many n input entries exist at its indexed location.

5 Program Organization and Description of Classes

Class definitions are in header files and their implementations are in the .cpp.
Main Class - creates an object that tests the member functions of a class
DoublyLinkedList and DListNode - are the two classes that create the main data structures
EmptyDLinkedListException - is called when a list is empty
Templates - used to create Doubly Linked-Lists of any type

6 Instructions to Compile and Run

Be sure to have the text input file in the same directory as where you extracted the program. Then you will need to update the directory location in main.cpp of phonebook. Compile by typing "make". The executable will be called Main. Do ./Main to execute.

7 Input and Output Specifications

The text input file should have entries listed line by line. The first line is the last name, the second is the first name, third is the UIN, and fourth is the contact's phone number. Empty spaces must exist between contacts in order for the program to distinguish between different Record objects.

Format of data from keyboard should be entered on the next line after the request for input. When typing in a search input, be sure to *capitalize* the first letter. For example if it is asking for the last name simply enter the last name below the request and capitalize the first letter. Same procedure for first name and UIN, enter the data on the line directly below the request for information. The UIN must be an integer. Almost all incorrect input suggestions are caught as long as you input a capital letter first for the last name. The program will then ask you again for the last name as long as an uppercase letter is detected and the name not found. All other cases return a segmentation fault. This is because capital letters are used by the program to index the vector of doubly linked-lists.

8 Logical Exceptions

The program will crash if the user inputs a query and does not *capitalize* the first letter for the last name. If first name is not a string and if the UIN is not an integer the program will also crash. Other exceptions are caught. An empty list and the input file not opening are also caught.

9 C++ Object Oriented or Generic Programming Features

This assignment incorporated:
Inheritance
Polymorphism
Templates

10 Tests

Could not find invalid and random cases that do not involve testing the menu. If for example incomplete data is provided then the program autofilled that field with an empty

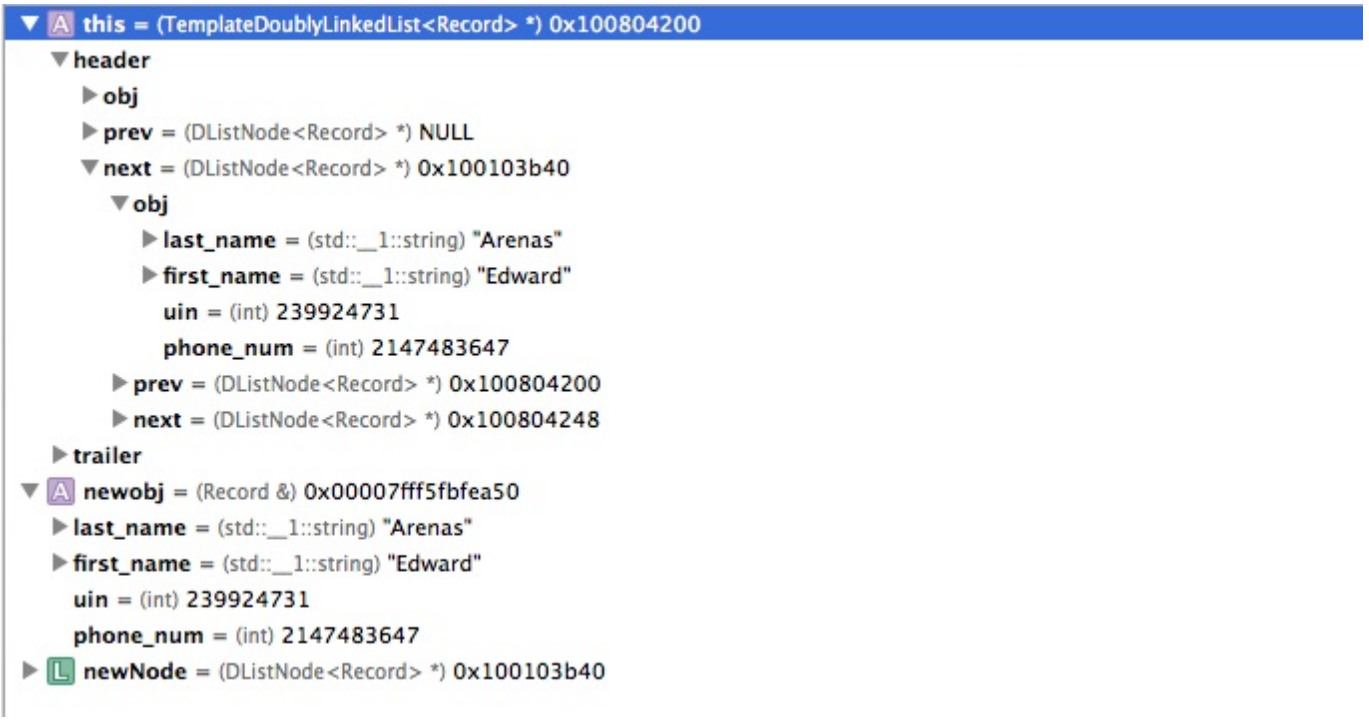

Figure 1: Valid Case of InsertLast: Header.next is equal to new object

string or 0 for integers. Valid cases of course exist and is demonstrated in Figure 1.