Problem Definition

This program will simulate a customer service application. The customer will be given a menu with five options. If the user selects new customer arrival, the user must be prompted to enter their name, the service they require, the date, and time. From this, a node must be created containing all the information they entered. If the user selects the serve customer option, the program must seek out the customer in a first-in first-out manner, and must delete the customer that has been in the linked list the longest. If the customer left option is selected, the user must be prompted to enter the name of the customer, once the name is found in the linked list, that customers node must be deleted. If the user selects the list all option, all customers still in the linked list must be displayed. If the user selects the quit program option, the program will be exited. The program must contain a CustomerList class to perform the linked list operations, and a CustomerNode.h file containing the customer node definition. In all there should be four files for the program. These are, CustomerNode.h, CustomerList.cpp. CustomerList.h, and source.cpp.

Analysis

Source.cpp

Header	Why
#include <iostream>	Needed to handle Input\Output
#include <string>	Needed to create string variables from string class
#include “CustomerList.h”	Needed for linked list operations
#include “CustomerNode.h”	Needed for CustomerNode struct that represents a customer

CustomerList.h

Header	Why
#include <string>	Needed to create string variables from string class
#include “CustomerNode.h”	Needed for CustomerNode struct that represents a customer

CustomerList.cpp

Header	Why
#include <iostream>	Needed to handle Input\Output
#include <string>	Needed to create string variables from string class
#include “CustomerList.h”	Needed for linked list operations
#include “CustomerNode.h”	Needed for CustomerNode struct that represents a customer
#include <iomanip>	Needed for the setw() member function for formatting output

CustomerNode.h

Header	Why
#include <string>	Needed to create string variables from string class

Source.cpp Variables and functions

Global Constants

Name	Use
NEW_CUSTOMER_ARRIVAL	Type: Constant int Used in menu to represent choice for adding a customer to the linked list. Value will be the integer 1
SERVER_CUSTOMER	Type: Constant int Used in menu to represent choice for serving the customer. Value will be the integer 2
CUSTOMER_LEFT	Type: Constant int Used in menu to represent a customer leaving prior to being served. Value will be the integer 3
LIST_ALL_CUSTOMERS	Type: Constant int

	Used in menu to represent choice for displaying all customers currently in the linked list. Value will be the integer 4
QUIT_CHOICE	Type: Constant int Used in menu to represent choice to exit the program. Value will be the integer 5

Variables in main()

Name	Use
customers	Type: CustomerList Will be the linked list that stores and performs operations on the customer nodes
user_choice	Type: int Will store the users choice value as an integer
name	Type: string Will store the customers name
service_needed	Type: string Will store the service that the customer needs performed
month	Type: int Will store the month that the customer says they arrived
day	Type: int Will store the day the customer says they arrived
year	Type: int Will store the year the customer says they arrived
hour	Type: int Will store the hour the customer says they arrived
minute	Type: int Will store the minute that customer says they

	arrived
head	Type: pointer to CustomerNode Will store the head of the customer node to be used in the customers.listAll() member function

Variables in get_user_choice()

Name	Use
choice	Type: int Will be used to store users choice from the menu. Will be validated within get_user_choice function

Variables in get_information()

Name	Use
name	Type: string Reference parameter Will store the customers name
service_needed	Type: string Reference parameter Will store the service that the customer needs performed
month	Type: int Reference parameter Will store the month that the customer says they arrived. Will be validated to ensure the month does not fall out of range
day	Type: int Reference parameter Will store the day the customer says they arrived Will be validated to ensure the day does not fall out of range for any given month
year	Type: int Reference parameter Will store the year the customer says they arrived.

hour	Type: int Reference parameter Will store the hour the customer says they arrived Will be validated to ensure the hour does not fall out of range
minute	Type: int Reference parameter Will store the minute that customer says they arrived. Will be validated to ensure the minute does not fall out of range

CustomerList.h

Variables in CustomerList.h

Name	Use
head	Type: pointer to CustomerNode Will store pointer to the most recent customer added to the linked list
last_node	Type: pointer to CustomerNode Will point to the node that has been in the linked list the longest

CustomerList.cpp Variables and Member functions

Variables in CustomerList() constructor

Name	Use
head	Type: pointer to CustomerNode Will set head pointer to nullptr
last_node	Type: pointer to CustomerNode Will set last_node pointer to nullptr

Variables in insertNode() member function

Name	Use
head	Type: Pointer to CustomerNode Will point to the newest value in linked list.
customer_name	Type: string Parameter Will store the customers name
service_needed	Type: string Parameter Will store the service that the customer needs performed
month_arrived	Type: int Parameter Will store the month that the customer says they arrived
day_arrived	Type: int Parameter Will store the day the customer says they arrived
year_arrived	Type: int Parameter Will store the year the customer says they arrived.
hour_arrived	Type: int Parameter Will store the hour the customer says they arrived
minute_arrived	Type: int Parameter Will store the minute that customer says they arrived.
new_node	Type: Pointer to CustomerNode Will be used to store a dynamically allocated CustomerNode into the linked list. Values will be assigned to CustomerNode data members based

	on the parameter variables listed above
temp	Type: Pointer to CustomerNode Will be used in the else clause on the insertNode member function. Used to prevent breaking of the linked list, and to establish a sequence number for the new_node variable

Variables in searchNode() member function

Name	Use
name	Type: string Will store the name being searched for to be deleted.
current_node	Type: Pointer to CustomerNode Will establish a starting point for each recursive call to searchNode() member variable

Variables in deleteNode() member function

Name	Use
name	Type: string Will store the name being searched for to be deleted.
node_ptr	Type: Pointer to CustomerNode Will be assigned the value of the node being searched for if node is found, or nullptr if node is not found
current_node	Type: Pointer to CustomerNode Will be used to search the linked list for the node that contains the name being searched for
previous_node	Type: Pointer to CustomerNode Will be assigned to the node that comes before

	the node being searched for so that it's next_node data member can be assigned to node_ptr next_node data member to prevent breaking of the linked list.
head	Type: Pointer to CustomerNode Will contain the value of the newest node in the linked list
last_node	Type: Pointer to CustomerNode Will contain the value of the oldest node in the linked list

Variables in serveCustomer() member function

Name	Use
current_node	Type: Pointer to CustomerNode Will be used to search the linked list for the node at the end of the linked list, which is the oldest node
previous_node	Type: Pointer to CustomerNode Will be assigned to the node that comes before the node being searched for so that it's next_node data member can be assigned to nullptr to prevent breaking of the linked list.
head	Type: Pointer to CustomerNode Will contain the value of the newest node in the linked list
last_node	Type: Pointer to CustomerNode Will contain the value of the oldest node in the linked list

Variables in listAll() member function

Name	Use
node_ptr	Type: Pointer to CustomerNode Parameter

	Will be used in recursive call to listAll() member function until node_ptr points to a nullptr. Then will be used to display each of node_pointers data members to the console

Variables in ~CustomerList() destructor

Name	Use
head	Type: pointer to CustomerNode Points to the node at the beginning of the list
last_node	Type: pointer to CustomerNode Points to the node at the end of the list
temp	Type: Pointer to CustomerNode Will temporarily store each node as the linked list is iterated through and destroyed
current_node	Type: Pointer to CustomerNode Will temporarily store each node that is destroyed, and be reassigned after each destruction

CustomerNode.h variables

Name	Use
sequence_number	Type: unsigned int Will store the sequence number of the customer the node represents
name	Type: string Will store the name of the customer the node represents

service_required	Type: string Will store the service required by the customer the node represents
month	Type: int Will store the month of arrival of the customer the node represents
day	Type: int Will store the day of arrival of the customer the node represents
year	Type: int Will store the year of arrival of the customer the node represents
hour	Type: int Will store the hour of arrival of the customer the node represents
minute	Type: int Will store the minute of arrival of the customer the node represents
next_node	Type: Pointer to CustomerNode Will store a pointer to the next node in the linked list if there is a next node, or nullptr if there is not a next node.

Design

Algorithm

1. Create a customer list object
2. Do-while loop
    a. Display menu to user
    b. Get users choice from menu
    c. If user chooses new customer arrival
        i. Get customer name, service needed, date and time of arrival
        ii. Pass to information insertNode

```
            1. Assign user to appropriate position in linked list
d. If user chooses serve customer
    i. Call serveCustomer
            1. Find customer at the end of the linked list and remove
e. If user chooses to remove a customer from the waiting list
    i. Get Customers name
    ii. Pass name to deleteNode
            1. Find name in linked list and delete if found
f. If user chooses list all current customers
    i. Call listAll
            1. Recursively process linked list until the end is found
            2. Display all customers starting at the last node
g. If the user chooses quit program
h. Exit do-while loop
i. Exit program
```

Implementation

This program was written in Microsoft Visual Studio Community edition. The Desktop used is running Windows 7 professional 64 bit, an Intel i5 4690k processor, and has 16Gb of ram.

I ran into several problems. When adding items to the linked list, I would occasionally get an infinite loop. This was caused by mixing getline(cin, var) and cin>>var, this was resolved by strategic placement of cin.ignore() in main() and get_information() to eliminate unwanted assignment of variables.

The insertNode(), deleteNode(), and listAll() member functions appear to be operating as they should be as of the writing with no known issues.However, the serveCustomer() member function initially caused crashes when called. This was fixed by putting in three if statements that handled cases where there were zero nodes, one node, and two or more nodes respectively. Also, within the while loop of the last if block, the statement previous_node->next_node = current_node was changed to previous_node = current_node. This change prevented the break in the linked list that was originally causing problems.