CS 215: Introduction to Program Design, Abstraction and Problem Solving

Final Lab Exam

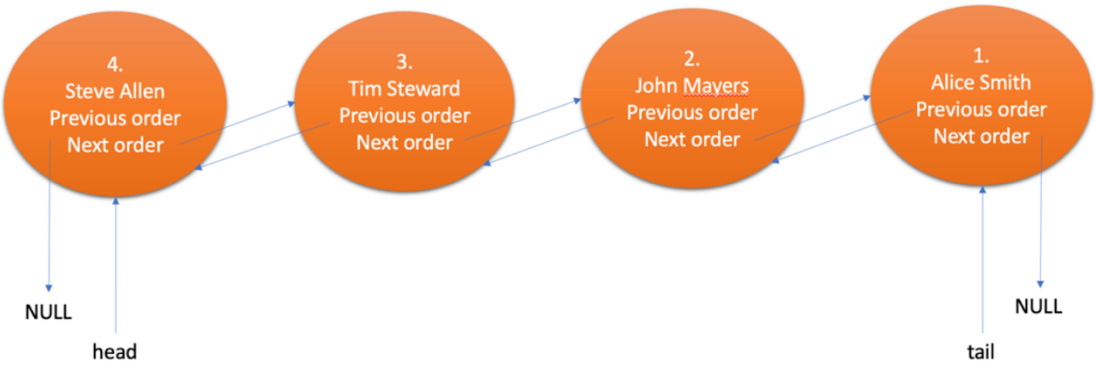
Queue as a Doubly Linked List (DLL)

The purpose of this exercise is to verify that every student can:

- Create queue structure using DLL
- implement class functions
- user input verification
- reuse written functions
- use pointers as private data members (reference, dereference)

Solve the following problem:

Help restaurant manager to create a queue of customer orders and fulfill those orders in the order they were received.



Class Order

NOTE: You are not allowed to modify this class interface!

```
class Order
{
    public:
        Order(int number, string name);
        string getCustomerName();
        void printOrder();
        friend class OrderQueue;
        ~Order();
    private:
        Order* next_order;
        Order* prev_order;
        int* number;
        string* name;
};
```

This class should have the following functions:

1) `Order(int number, string name)` – overloaded constructor. This constructor should dynamically allocate space for *number* and *name* pointers.
2) `string getCustomerName()` – returns full name of a customer
3) `void printOrder()` – prints order number and customer's full name
4) `~Order()` – destructor, frees memory allocated by the constructor

And four data members:

`next_order, prev_order, number, and name`

Class OrderQueue

NOTE: You are not allowed to modify this class interface!

```
class OrderQueue
{
    public:
        OrderQueue();
        void addOrderToQueue(Order* order);
        void removeOrderFromQueue();
        Order& getFirstOrder();
        bool empty();
        void printAllOrders();
    private:
        Order* head;
        Order* tail;
};
```

This class should have the following functions:

1) `OrderQueue()` – default constructor
2) `Void addOrderToQueue(Order* order)` – adds customer order to DLL according to queue rules (FIFO ordering)
3) `Void removeOrderFromQueue()` – removes customer order according to queue rules (FIFO ordering)
4) `Order& getFirstOrder()` – gets the order that is first in the queue to be fulfilled. FYI: returning by reference (Order&) means that you are returning an address of the original object. The caller function can use this direct reference to continue modify the object or for a simple access. Returning by reference is fast when we deal with structs and classes.
5) `bool empty()` – checks if DLL is empty.
   NOTE: this function MUST be used in the main function as you are fulfilling orders until there is no more orders to fulfill.
6) `printAllOrders()` – prints DLL. You must reuse function from class Order here.

And two data members:

`head, tail`

Main Function

1) gets as many customer orders as user is going to enter (We will assume that user enters correct information, no check necessary).
2) you need to use "q" as the indication of end of user input
3) if order queue is not empty, print all customer orders, then fulfill all the orders by using FIFO queue ordering.
   NOTE: This is done automatically by the program (with intermediate result print outs, see Sample Run) without user interaction.
4) After all orders have been fulfilled, display corresponding message and quit the program.

You should not:

- Modify Order and OrderQueue class interfaces
- Use global variables
- Have header or comments (optional)

Sample Run (normal execution)

```
Anastasias-MacBook-Pro-5:FinalLab2019 nastiakazadi$ ./prog
Enter custormer full name (q to quit): Alice Smith
Enter custormer full name (q to quit): John Mayers
Enter custormer full name (q to quit): Tim Steward
Enter custormer full name (q to quit): Steve Alen
Enter custormer full name (q to quit): q

Customer queue:
1. Alice Smith
2. John Mayers
3. Tim Steward
4. Steve Alen

>> Fulfilling order for: Alice Smith

Customer queue:
2. John Mayers
3. Tim Steward
4. Steve Alen

>> Fulfilling order for: John Mayers

Customer queue:
3. Tim Steward
4. Steve Alen

>> Fulfilling order for: Tim Steward

Customer queue:
4. Steve Alen

>> Fulfilling order for: Steve Alen

--------------------------------
All orders have been fulfilled!
```

Submission
Zip the following files into CS215FinalLab.zip and submit to Canvas by 2:45pm.
NO LATE SUBMISSIONS WILL BE ACCEPTED.

- Order.h
- Order.cpp
- OrderQueue.h
- OrderQueue.cpp
- CS215FinalLab.cpp

NOTE: if you have incorrect file names, you will lose points!