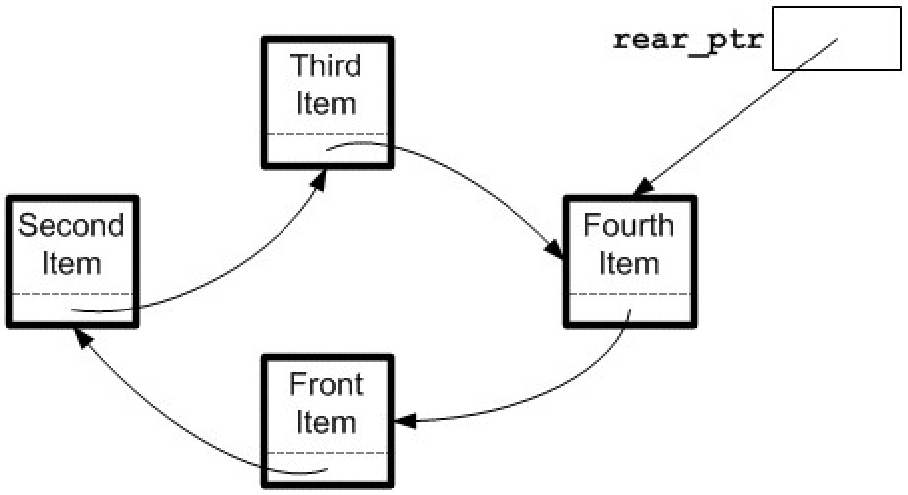
CS 3358 (Data Structures, §255, Spring 2020) by Lee S. Koh

Optional Assignment 1

(posted 04/25/2020, **DUE 05/05/2020 05:00 pm**)
*(I like the **555** in the **due date/time** 😉)*

The Assignment

A queue can be implemented using a linked list using the idea of *circular linked list*. The suggested book talks about the idea in the description of Programming Project 6 on page 434 (page 428 for the third edition, page 416 for the second edition), the relevant portion of which is reproduced below:

> In this chapter we gave a linked-list implementation of a queue. This implementation used two named pointers called **`front_ptr`** and **`rear_ptr`** to point to the front and the rear nodes of the queue (linked list). A ***circular linked list*** is similar to a regular linked list, except that the pointer field in the "last node" points back to the "first node." (Of course, after this change it is no longer clear which node, if any, is intrinsically "first.") If we use a circular linked list, then we need only one pointer to implement a queue, since the front node and the rear node are adjacent nodes, as shown by the following diagram:
>
> 
>
>
> In the diagram we have called the single pointer **`rear_ptr`**, because it points to the last node in the queue. It turns out that this gives a more efficient implementation than having it point to the first node in the queue.

This assignment gives you the opportunity to apply the idea. It also provides an opportunity for you to see one way to use a linked list (and by extension a binary tree) with C++ class to implement an ADT.

Goal

To gain experience designing and implementing a *queue* class using a *circular linked list*.

Supplied files and what you will need to do

- Supplied files.
- You should not make any changes to **QueueCLL.h**.
- You should not remove any functions from **QueueCLL.cpp**.
- You are to fill in the implementations of all the functions in the file **QueueCLL.cpp**.

- The test program **cqueueTest.cpp** has been provided to help you debug your implementation. You should not have to make any changes to it.

Other Tips / Helpers

- One "special" function of the **QeueCLL** class included in this exercise is listed below with a brief explanation of how to implement it:

  `Item peek(size_type n) const` – returns the item on the queue which is at the *n*th position, without altering the queue. For example, using the diagram shown above, if *n* = 2, the second item would be returned and the queue would be unchanged. If *n* = 7, the third item would be returned and the queue would be unchanged. (Note that there are only 4 items on the queue.)
- **`a01test.out`** (one of the supplied files) captures the expected output when **0**, **5**, **9** and **25** are entered at the prompt of the test program.

Deliverables

- *QueueCLL.cpp* (with the implemenation for all functions filled in). (softcopy)

NOTES:

- You can earn ***up to*** 50 points by doing this optional assignment:
  - 660 (= 100 + 100 + 100 + 50 + (50+50) + (50+60) + 100) is the total full points for the 7 required assignments.
  - If you received less than the full total points for the required assignments, this assignment gives you the opportunity to make up for the shortfall.
  - The highest you can receive from this assignment is the shortfall (from full total points for required assignments) or 50, whichever is the *lower*.
    -- This means that ***no student can get more than the perfect score (660 points total)*** for assignments.

  If you don't or don't quite need it for the above "making-up" purpose, it can still be useful (as "good grace") when your overall score is near a borderline between two grades.
- If you discover anything that appears incorrect, please let me know (through email) as soon as possible.
- Be sure to check the homepage often for any updates and/or further information.

Use your browser's **Back** button to go back to previous page. (click here to return to my homepage)