Queue Implementation Using a Circular Array

Due Tuesday by 11:59pm **Points** 50 **Submitting** an external tool

A circular array is an array-based data structure where the end of the array "wraps around" to the beginning of the array. For example when using an index to iterate through the circular array, once reaching the last element incrementing the index once more will reset the index to 0 and thus reaching the first element immediately after the last element. For example with a circular array of **size** 3 and index **i** starting with 0 and incrementing 4 times:

`i:0 [0][1][2]-> i:1 [0][1][2]-> i:2 [0][1][2]-> i:0 [0][1][2]-> i:1 [0][1][2]`

A queue can be efficiently implementing using this data structure by using two indicies: one for the **front** item in the queue (next to be dequeued) and one for the **back** item in the queue (most recently enqueued). When enqueuing an item the **back** index is incremented according to the circular array and when dequeuing an item the **front** index is incremented. The initial state of the queue you will be empty, this empty state is indicated by the **back** index being immediately before the **front** index; upon the first item being enqueued, the **front** and **back** index are equal since the only (and most recent item enqueued) is also the first item to be dequeued. Additionally since the circular array is of a fixed size the queue can be full; this occurs when there is one less item in the queue than the size of the array. The queue is full when the **back** index is two positions before the front index.

Download the starter code and use **integerArrayQueue.h** for the definition of the **IntegerArrayQueue** class. The constructor functions, destructor and the **printArrayQueue()** functions are already implemented and should not be changed but you will need to implement the **enqueue()** and **dequeue()** functions.

The **enqueue()** function returns a **bool** value: **true** if the queue is non-full prior to enqueuing the value, **false** if the queue is already full in which case the value is not enqueued. The **dequeue()** function returns the dequeued value if the queue is non-empty, or 0 if the queue is empty. These two functions should be implemented in **integerArrayQueue.cpp**. A main function is not needed for submission but can be useful for testing your implemented functions.

Queue Implementation Using a Circular Array Starter Code.zip
(https://clemson.instructure.com/courses/198274/files/17721137?wrap=1) ↓
(https://clemson.instructure.com/courses/198274/files/17721137/download?download_frd=1)