CS 211 – Data Structures
Lab for Week #09

For this lab exercise, you will work either in 2-person teams or individually. You are HIGHLY ENCOURAGED to work in 2-person teams! The point here is to have teams discuss the lab and offer each other support in making sure the IDE of your choice is working.

Lab Exercise

(NOTE: some of the code as specified below is already supplied!)
Into a C++ Application project, download and copy the contents of the following files:

ArrayQueue.cpp **ArrayQueue.h** **week09Lab.cpp**

NOTE that ArrayQueue.h and testQueue.cpp should not be modified for this lab!

For this lab, define and implement a class called **ArrayQueue** that creates a simple queue data structure of fixed size. The queue should a capacity for exactly 10 items of type **int**. You MUST use an array to implement your queue (sorry, NO vectors!). Keep in mind, though, that the size of the **Queue** is limited to 10 items. All methods should be able to handle the special cases when the **Queue** is empty and when the **Queue** is full.

The queue should be circular, and you should also consider using local private **bool** variables in the **ArrayQueue** class that helps keep track of whether the queue is empty or full. The relative positions of the **first** and **last** (or **front** and **end**) index values can be used to determine whether the **Queue** is full or empty – see IMPORTANT HINTS below!

Do **not** use pre-existing methods for the **ArrayQueue** class library. You must write the C++ code for the methods yourself! Also, a constructor, a destructor, and a **printArray()** are pre-written for your convenience.

Your **ArrayQueue** class needs to support the following methods:

bool isFull() – returns **true** if the queue is full, **false** if not
bool isEmpty() – returns **true** if the queue is empty, **false** if not
int firstEl() – returns the **int** element at the front of the queue, without removing from the queue!
void clearQueue() – empties the queue completely
bool enQueue(int el) – adds element **el** to back of the queue – returns **true** if successful, **false** if not
int deQueue() – returns the **int** value at the front of the queue, and removes it from the queue

For the **deQueue** and **firstEl** methods, if the queue is empty, the method should return **-99999**.

At the top of **each** file, document these files as follows based on your 2-person team:

```
# CS 211 Fall 2024 – Week 09 Lab
# <Student_Name1> and <Student_Name2>
```

Substitute your own names for the **<Student_Name>** placeholders above.

IMPORTANT HINTS

Your **enQueue** should check the following cases:

- If the Queue is full, you can't add any elements!
- If the Queue is empty, adding an element means you'll need to set **queueEmpty** to **false**

Your **deQueue** should check the following cases:

- If the Queue is empty, you can't remove any elements!
- If the Queue has exactly one element, removing it means you'll need to set **queueEmpty** to **true**

Note the following code examples for incrementing **frontIndex** and **backIndex**!

```
// It turns out C++ has a thing about the % (modulo) function:
// Some C++ compilers will evaluate -7 % 3 as -1 instead of 2
// This means the following computations as seen here are
// needed to correctly increment frontIndex and backIndex

frontIndex = (frontIndex + 1 + QUEUE_SIZE) % QUEUE_SIZE;
backIndex = (backIndex + 1 + QUEUE_SIZE) % QUEUE_SIZE;
```

NOTE that you CANNOT check to see if the queue is empty based on simply comparing **frontIndex** and **backIndex**. You need to set **queueEmpty** to **true** when *removing the only item in the queue* (and NOT update the index values), and you need to set **queueEmpty** to **false** when *inserting into an empty queue* (again, NOT updating the index values!).
To check whether there's exactly one element in the Queue, you can use the following:

```
// Expression below is true is Queue has exactly one element
(frontIndex == backIndex) && !queueEmpty
```

To check to see whether the queue is full, you can use the following:

```
int difference = backIndex – frontIndex;
// Below is a Boolean expression that is true if Queue is full
((difference + QUEUE_SIZE) % QUEUE_SIZE == QUEUE_SIZE - 1)
```

ONCE FINISHED, ALL STUDENTS SHOULD SUBMIT THE ArrayQueue.cpp FILE ONLY VIA CANVAS.
THE ArrayQueue.h AND testQueue.cpp FILES DO NOT NEED TO BE SUBMITTED, AS THEY SHOULD HAVE NOT BEEN MODIFIED!