CSc 300
Assignment #3
Gamradt
Due: 10-18-23 (Late: 10-25-23)

Create a user-defined Abstract Data Type (ADT) named **Queue**

- Use an appropriate set of C++ header/implementation files as discussed in class
- **Queue** is implemented as a **dynamically allocated Array**
    - Implemented as a **circular queue**
    - See **C++ Pointers** under D2L Lecture Notes
- **Queue** consists of 0 or more **Element** values
    - **Element** is an exportable standard library **int** data type

The **Queue** ADT must define and implement the following data types and operations.

- Do not add to or modify the public interface (exportable components – public components).
- Do not add to or modify any attributes or data types (storage components).

<u>Exportable Operations: (declared .h file and defined .cpp file)</u>

Queue	default constructor function – creates an initialized empty queue – size 3 (+)
Queue	parameterized constructor – creates an initialized empty queue – size user specified (+)
Queue	copy constructor – creates a duplicate copy of an existing queue (*)
~Queue	destructor function – removes all elements from the queue queue instance state before going out of scope – initialized empty queue
enqueue	inserts a new element to the tail of the queue
dequeue	removes an existing element from the head of the queue
view	displays the contents of the queue from the head to the tail (*) view function uses a non-destructive implementation

(+) Implement a minimum number of constructor functions
(*) Before an element can be accessed and processed it must first be removed from the head of the queue

<u>Exportable Operations: (declared .h file and defined .cpp file)</u>

isEmpty	returns true if the current queue instance is empty – false otherwise
isFull	returns true if the current queue instance is full – false otherwise

<u>User-Defined Data Types:</u>
Element
ElementPtr

Queue Required Output Format: (view)

```
HEAD -> TAIL                          // Output for an empty Queue instance
HEAD -> 5 -> -3 -> TAIL               // Output for a populated Queue instance
```

Required header file (.h). // **only partially specified**

```
//  General description of the ADT and supported operations – exportable operations only
//  Do not include any implementation details

#ifndef _QUEUE_H                                        // Guard
#define _QUEUE_H

typedef int Element;
typedef Element * ElementPtr;

class Queue {
        public:                                         // exportable
//  General description of each of the ADT operations/functions – exportable operations only
                explicit Queue( … );                    // replace … with required arguments
                Queue( Queue & );                       // reuse enqueue & dequeue
                ~Queue();                               // reuse dequeue
                void enqueue( const Element );
                void dequeue( Element & );
                void view();                            // reuse enqueue & dequeue
        private:                                        // non-exportable
//  No private member documentation – implementation details are hidden/abstracted away
                const short QUEUE_SIZE;                 // requires initialization
                ElementPtr queueArray;
                short head, tail;
                bool isEmpty() const;
                bool isFull() const;
};

#endif                                                  // Guard
```

Queue ADT include sequence: // Never include .cpp files

$$\begin{tikzcd}\text{main.cpp} \arrow[r] & \text{Queue.h} & \text{Queue.cpp} \arrow[l]\end{tikzcd}$$

Queue ADT incremental building sequence: // Using make

1. Place all files in the project folder // I would use Gamradt4
2. make // Process Makefile
3. ./output // Run project – make generated executable

Make sure that you completely document the header/implementation files

- The header (.h) file tells the user exactly how to use your ADT
    - General descriptions only – do not include implementation details
- The implementation file (.cpp) tells the implementer/programmer exactly how the ADT works
    - Detailed descriptions – include implementation details
- See **Documentation Requirements** – D2L Handouts Folder

I will write a test program that will include your **Queue** ADT so all header/implementation files tested must use common names. You **MUST** use:

- the **EXACT** same names for each data type and function in the header/implementation files
- the **EXACT** same function argument sequence in the header/implementation files

Use **PITA** everywhere possible

- Prefer Initialization to Assignment

Remember that a queue uses the basic operations of **enqueue** and **dequeue** to support all additional operations.

- Apply function **Reuse** wherever possible.
    - E.g., copy constructor, destructor, view, …

Project Folder:	Lastname4	// I would use Gamradt3
Queue.h	**Queue** class header file	
Queue.cpp	**Queue** class implementation file	
main.cpp	driver program file	// I will use my own
Makefile	appropriate set of incremental build rules	// “1” module

Push your assignment solution to your GitHub account, then send me a shared link to the assignment repository

- E.g., CSc300 // CSc300
    - Remember that a 20% reduction is applied for not using GitHub
    - See **Assignment Requirements** – D2L Handouts Folder

List the class number, your lastname, and assignment number as the e-mail message subject:
SUBJECT: csc300 – Lastname – a3 // I would use “… Gamradt …”