**CSc 300**
**Assignment #4**
**Gamradt**
**Due: 10-18-21 (Late: 10-25-21)**

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**
- **Queue** consists of 0 or more **QElement** values
  - **QElement** is an exportable standard library **string** 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).

**Exportable Operations: (declared .h file and defined .cpp file)**

| | |
|---|---|
| **Queue** | default constructor function – creates an initialized empty queue (+)<br>default size of 2 |
| **Queue** | overloaded/parameterized constructor – creates an initialized empty queue (+)<br>user specified size |
| **Queue** | copy constructor – creates a duplicate copy of an existing queue (*) |
| **~Queue** | destructor function – destroys the existing queue<br>queue instance state before going out of scope – initialized empty queue |
| **enqueue** | inserts a new element to the back of the queue |
| **dequeue** | removes an existing element from the front of the queue |
| **view** | displays the contents of the queue from the front to the back (*) |
| **isEmpty** | returns true if the current queue instance is empty – false otherwise |
| **isFull** | returns true if the current queue instance is full – false otherwise |

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

**User-Defined Data Types:**
**QElement – QPointer**

**Queue Required Output Format: (view)**
HEAD -> TAIL
HEAD -> CSC -> 300 -> TAIL

**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

typedef string QElement;

class Queue {
        public:                                         // exportable
//  General description of each of the ADT operations/functions – exportable operations only
                Queue( … );
                Queue( Queue & );
                ~Queue();
                void enqueue( const Element );
                QElement dequeue();
                void view();
                bool isEmpty() const;
                bool isFull() const;
        private:                                        // non-exportable
//  No private member documentation – implementation details are hidden/abstracted away
                const short Q_SIZE;                     // must be initializaed
                typedef QElement * QPointer;
                QPointer queue;
                short head, tail;
};

#endif                                                  // Guard
```

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

main.cpp ⟶ Queue.h ⟵ Queue.cpp

**Queue ADT incremental building sequence:** // Using make

1. Place all files in the project folder // I would use Gamradt4
2. make // Process Makefile
3. ./runqueue // 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

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, …

Use **PITA** everywhere possible

- Prefer Initialization to Assignment

| | | |
|---|---|---|
| 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~~ Upload your assignment solution to your ~~GitHub~~ Box account, then send me a shared link to the assignment ~~repository~~ archive file

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

List the class number, your lastname, and assignment number as the e-mail message subject:
SUBJECT: csc300 – Lastname – a4 // I would use "… Gamradt …"

**Notes**:

- Before submitting your project run "make clean" to remove your executable file and object files
  - Also remove you main.cpp file