Data Structure Lab Assignment (CS 2172) Assignment 3: Queue

Time: 2 weeks

## 1. Implementation of an Integer Queue

In this assignment you are required to implement a queue where integer data can be en-queued and de-queued. You should define (**typedef**) an appropriate type called **queue** such that multiple variables of type **queue** can be defined.

Your implementation should support the following functions (interface) for the queue.

- 1. **Queue createIntegerQueue(int queueSize)** This allocates space for the queue to hold maximum "**queueSize**" number of integers and initializes that space. Its return type is "**queue**". The function returns **NULL** if creation of the queue fails.
- 2. **int enqueueInteger(queue q, int d)** It en-queues the data **d** in the queue **q**. It returns 1 if the operation is successful. If the operation fails (say, when queue **q** is full and **d** cannot be en-queued), the function returns 0.
- 3. **int dequeueInteger(queue q, int \*dp)** It de-queues from the queue **q** and stores the dequeued element at address **dp**. It returns 1 if the operation is successful. If the operation fails (say, when queue s is empty and **dequeueInteger()** is attempted), the function returns 0.
- 4. *int freeIntegerQueue(queue q)* It frees the space allocated for queue *q*. It returns 1 if the operation is successful. If the operation fails (say, *q* does not refer to a valid queue), the function returns 0.
- 5. *int isIntegerQueueFull(queue q)* It returns 1 if the queue associated with *q* is full. The function returns 0 otherwise. If *q* does not refer to a valid queue then too the function returns 1.
- 6. *int isIntegerQueueEmpty(queue s)* It returns 1 if the queue associated with *q* is empty. The function returns 0 otherwise. If *q* does not refer to a valid queue then too the function returns 1.

Note: You should implement queue functionality using the circular queue concept. This will be done in two stages

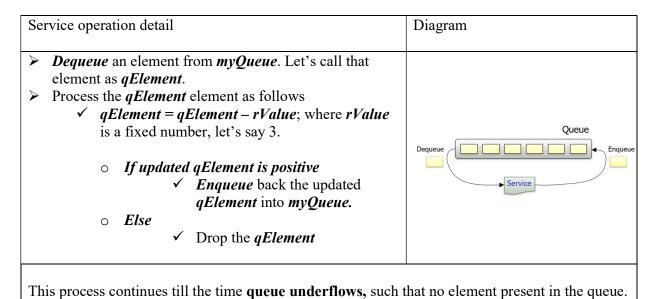
Stage-1: Initially avoid having a *count* variable that represents the number of elements in the queue. This is for better understanding of the difficulties,.

Stage-2: Then finally, introduce the *count* variable to make it simple.

Write suitable main() function to demonstrate that your functions are working as desired.

## 2. Using above Queue implementation, simulate the following

Assume there is one queue created as *myQueue* of size *N*. Also, consider that a series of **positive** integers are read from the user and queued into *myQueue*. The process of populating into the queue continues till the **queue overflows**. Now perform the following steps of operation on each element of the queue.



**Advancement**: Modify the problem where *rValue* is a randomly generated positive integer between (1-9) and repeat testing.

Note: Please ask to the instructor for more testing and observations of this problem.