# Lab #10

CS-2050

April 5, 2024

## 1 Requirements

In this lab, you are tasked with implementing a *time-priority* Queue ADT. In this context, "time-priority" indicates that the oldest item on the queue (IE: that which has been on the queue the longest) is the next to dequeue. You may consider it to be the same as FIFO.

Note the complexity requirement specified on each function. If your implementation of a function does not meet the complexity requirement specified, you cannot get credit for that function.

### 1.1 newQueue

```
// O(1)

Queue * newQueue(void)
```

**Info:** This function creates and returns a new Queue. If creation was successful, it returns a pointer to the new Queue, otherwise NULL.

### 1.2 getSize

```
// O(1)

int getSize(Queue *q)
```

**Info:** This function takes a Queue, and returns the number of items on the queue.

### 1.3 enQueue

```
// O(1)

int enQueue(Queue *q, void *data)
```

**Info:** This function takes a Queue, and a data pointer. It inserts the data onto the queue and returns 1 on success, or 0 on failure. You may assume that data will never be NULL.

### 1.4 peek

```
// O(1)

void * peek(Queue *q)
```

**Info:** This function takes a Queue, and returns the item at the front of the queue without removing it. If the queue is empty, it returns NULL.

### 1.5 deQueue

```
// O(1)

void * deQueue(Queue *q)
```

**Info:** This function takes a Queue, and removes and returns the item at the front of the queue. If the queue is empty, it returns NULL.

### 1.6 destroy

```
// O(n)

void destroy(Queue *q)
```

**Info:** This function takes a Queue, and frees all memory allocated to it. Remember that the data on the queue belongs to the *user*, not to your implementation.

**Grading: 30 points**

1. Write required *newQueue* function
   * 5 points
2. Write required *getSize* function
   * 5 points
3. Write required *enQueue* function
   * 5 points
4. Write required *peek* function
   * 5 points
5. Write required *deQueue* function
   * 5 points
6. Write required *destroy* function
   * 5 points

**Notice:**

1. All of your lab submissions **must** include documentation to receive full points.
2. All of your lab submissions must compile under GCC using the $-Wall$, $-Werror$, and $-Wpedantic$ flags to be considered for a grade.
3. You are expected to provide proper documentation in every lab submission, in the form of code comments. For an example of proper lab documentation and a clear description of our expectations, see the lab intro PDF.