Prelab 7

For this prelab you are to implement six functions for a Queue ADT:

```
/* This function returns the error code from the most
   recently executed queue operation. 0 implies success,
   1 implies out-of-memory error. Some functions may
   document additional error conditions. NOTE: All
   queue functions assign an error code.   */
int getQueueErrorCode(Queue *)

/* This function returns an initialized Queue variable.
   Every queue variable must be initialized before
   applying subsequent queue functions. */
Queue * queueInit()

/* This function enqueues an object into the queue.
   For convenience, error code is returned directly
   (and also can be obtained via getQueueErrorCode) */
int enqueue(void*, Queue *)

/* This function performs dequeue and returns
   object at front of queue. NULL is returned
   if queue is empty and error code is set to 2.
   NOTE: User should check error code if null
   objects are permitted in the queue. */
void* dequeue(Queue *)

/* This function returns the number of objects
   in the queue. */
int getQueueSize(Queue *)

/* This function uninitializes a queue and frees all
   memory associated with it. NOTE: value of Queue
   variable is undefined after this function is
   applied, i.e., it should not be used unless
   initialized again using queueInit. */
void freeQueue(Queue *)
```

Your underlying representation should be a simple (or circular) linked list with tail pointer so that all functions take only O(1) time except for freeQueue, which obviously must take time proportional to the size of the queue.

This is the first time you've been given a specification that includes a performance constraint. From now on your documentation – *both for prelabs and labs* – should include the computational complexity for every function. That's easy for this prelab because all functions other than freeQueue are *required* to have O(1) run-time complexity.