Queue Worksheet

Here is an interface that defines a generic queue.

```java
public interface Queue<T> {
    public void enqueue(T item);   // Add an item to end
    public T front();               // Get item waiting the longest
    public T dequeue();             // Remove item waiting the longest
    public boolean isEmpty();       // Are any items waiting?
    public int size();              // How many items are waiting?
}
```

Assume that you want to store the elements that are in the queue in a LinkedList:

```java
private LinkedList<T> values = new LinkedList<T>();
```

How would you implement each of the methods in the interface?
What is the O() cost of each of these methods?
Next, let’s suppose that we store the queue items in a circular array using these instance variables:

```java
int first;  // index of front of queue
int last;   // index of last element of queue
int numItems;  // how many in queue
int capacity;  // how many queue can hold
T[] values;   // array for data
```

If our circular array contains:

![Circular array with 3, 4, 5, 6, 7, 8](image1)

When we call enqueue(9), the circular array should then contain:

![Circular array with 3, 4, 5, 6, 7, 8, 9](image2)

How would you implement the enqueue method?
Suppose the circular array is full, like this:

If the user calls enqueue again, we could reallocate the circular array, doubling its size, similar to what we did for ArrayList. What values should be where in the reallocated circular array? What are the values of first and last?

How would you write a reallocate method to get this effect?

```java
private void reallocate() {
    // Your code here
}
```
Assume that the next and prev fields in a LinkedList’s nodes each take 4 bytes of memory and the reference to the Object that contains the node’s value also takes 4 bytes of memory.

If a Queue contains 100 entries, how much memory does the Queue use if represented as a LinkedList? What percent of that memory holds the actual values that are in the queue?

Suppose you represent a queue with a circular array that is large enough to hold 128 values and it currently contains 100 values. What percent of the memory is holding actual values?

In general, how would you decide which representation to use for a particular application?