Problem 1. Getting starting with Queue Implementation

In Python, implement and test a Class that implements the Queue ADT as a circular array. The idea is as follows: the data is stored in a list. You keep track of the index where the data begins and the index where the data ends. You enqueue by inserting at the end, and dequeue by taking the element from the beginning.

```
index    0    1    2    3    4    5
              10   12

Begin at index: 2
End at index: 3
-----------------------------------
After enqueue 7:
index    0    1    2    3    4    5
              10   12   7

Begin at index: 2
End at index: 4
-----------------------------------
After dequeue:
index    0    1    2    3    4    5
                   12   7

Begin at index: 3
End at index: 4
```

The indices "wrap around" when reaching the end of the list:

```
index    0    1    2    3    4    5
              10   12



Begin at index: 2
End at index: 3
-----------------------------------
After enqueue 7:
index    0    1    2    3    4    5
              10   12   7

Begin at index: 2
End at index: 4
-----------------------------------
After dequeue:
index    0    1    2    3    4    5
                   12   7

Begin at index: 3
End at index: 4
```

```
----------------------------------
After enqueue 1:
index    0    1    2    3    4    5
                   12   7    1

Begin at index: 3
End at index: 5
After enqueue 6:
index    0    1    2    3    4    5
         6         12   7    1

Begin at index: 3
End at index: 0
```

Start with the Queue implementation we had in class. Change it to start with a fixed-size list of None's and modify it so that the indices wrap around when needed.

Write an `__str__` method for your CircularQueue class. It should output a string that represents the contents of the queue.

Test your CircularQueue class.

Problem 2. Sorting CircularQueues

Suppose we would like to sort CircularQueues, assuming that they contain integers as data. The Queues would be sorted lexicographically: a sequence of number s_1 comes before sequence s_2 if in the first number that differs between s_1 and s_2, the number in s_1 is smaller than the one in s_2. For example, the following is an example a list of sorted queues:

```
1->5->5
2->1->5
2->5->10->12
11->2
```

Write code that sorts CircularQueues. Do this by implementing the `__lt__` method.

Problem 3. Queue Implementation in C

Write code in C to implement a Queue, similarly to Question 1.

You can look up the implementation of ArrayList from Lab 5. Note that you should first implement a Queue without considering the issue of having to realloc when you run out of space.

Problem 4. Queue with Dynamic Memory

Modify your code from Question 3 to handle the situation when you run out of space an need to resize the block used to store the data.