CS 2401 Assignment #9

Due Date: Wednesday, November 28, 07:00AM (See the syllabus for late policy).

Objective: The goal of this assignment is to practice Queue operations.

Assignment: The following class is written for a standard array-based `Queue`, which we went over during the lecture session. Blackboard contains the slides. The `Queue` class from the lecture is copied below.

```
public class Queue{
	private int QUEUE_SIZE = 50;
	private Object[] items;
	private int front, back, count;

	public Queue() {
		items = new Object[QUEUE_SIZE];
		front = 0;
		back = QUEUE_SIZE -1;
		count =0;
	}

	public boolean isEmpty(){
		return count ==0;
	}

	public boolean isFull(){
		return count == QUEUE_SIZE;
	}

	public void enqueue(Object newItem){
		if (!isFull()){
			back = (back+1) % QUEUE_SIZE;
			items[back] = newItem;
			count++;
		  return;
		} else
			System.out.println("Trying to enqueue into full queue");
	}

	public Object dequeue(){
		if (!isEmpty()){
			Object queueFront = items[front];
			front = (front+1) % QUEUE_SIZE;
			count--;
		  return queueFront;
		}else
			System.out.println("Trying to dequeue from empty queue");
		return null;
	}

	public void dequeueAll(){
		items = new Object[QUEUE_SIZE];
		front = 0;
		back = QUEUE_SIZE -1;
		count =0;
	}
```

```
	public Object peek(){
		if (!isEmpty()) {
			return items[front];
		}
		else
			System.out.println("Trying to peek with empty queue");
		return null;
	}

	public int size(){
		return count;
	}

}// End of class Queue
```

Your tasks in this assignment are outlined below.

1. Change the `QUEUE_SIZE=50` to `QUEUE_SIZE=5` in the given code. This will make analyses of your code easier.
2. Change the `enqueue` method of the `Queue` class in such a way that if the array is full then the array-size will become double. Obviously, the new item will be added in the expanded new array in that case. That is, `enqueue` will never fail due to the size-limitation of the array. For this assignment, enqueue only numbers (either int, or double). In reality, you could enqueue any object but "any object" is the out of scope of this assignment.
3. Write a different class named `Runner.java` from which you will create a queue object and demonstrate that your `Queue` class works. In `Runner.java`, in addition to the `main` method, write the following methods and demonstrate that these methods work too.
    - **(a)** **`public static void printQueue(Queue q):`** Print all the elements of a queue. **`q`** must have the same numbers in the same sequence after printing all the numbers.
    - **(b)** **`public static void findMaxInQueue(Queue q):`** Print the largest of all the numbers in **`q`**. The queue **`q`** must have the same numbers in the same sequence after entering the method and before leaving it.
    - **(c)** **`public static void reverseQueue(Queue q):`** Reverse the content of the queue.

The use of object cloning is strictly prohibited in this assignment. A sample terminal output of `Runner.java` is provided below.

```
My queue is as follows:
10 20 30 40 50
I am going to dequeue one element.
My queue is as follows:
20 30 40 50
I am going to reverse my Queue.
My queue is as follows:
50 40 30 20
I am going to enqueue 60.
My queue is as follows:
50 40 30 20 60
I am going to enqueue 70.
Queue is full. Doubling the size.
New max. size is: 10
Entered the new item.
My queue is as follows:
50 40 30 20 60 70
I am going to reverse my Queue.
My queue is as follows:
70 60 20 30 40 50
The largest number in the queue is: 70
My queue is as follows:
70 60 20 30 40 50
```

Deliverables: `Queue.java` and `Runner.java`. You must use Blackboard to submit. Talk to your TA for further instructions.