Instructions

You may work in a group of any size. Ask your TA lots of questions, and feel free to make lots of noise. Next week, the pre-tutorial and in-tutorial exercises will start being worth marks.

Two interfaces

Below is the Queue interface from the lecture notes, plus a new interface for stacks.

```java
public interface Stack{
    // Push o onto me.
    // Precondition: I am not full.
    public void push (Object o);

    // Pop off and return my top element.
    // Precondition: I am not empty.
    public Object pop ();

    // Return whether or not I am empty.
    public boolean isEmpty ();

    // Return whether or not I am full.
    public boolean isFull ();

    // Empty me out.
    public void clear ();
}
```

```java
public interface Queue {
    // Append o to me.
    public void enqueue(Object o);

    // Return my front Object.
    public Object head();

    // Remove and return my front Object.
    public Object dequeue();

    // Return the number of Objects in me.
    public int size();
}
```

Your task

Remember that we have written a class ArrayQueue that implements Queue. On the other side of this page, is the outline for a class that implements Stack. You are going to fill in the outline, using an instance of ArrayQueue to store the stack. (Wierd!)

1. In the comments on the back of this page, write a representation invariant that describes how exactly q will represent a stack. Where will the top of the stack be?

2. Figure out how push() and pop() will work, given your representation invariant.

3. Write bodies for methods push() and pop().
public class ArrayStack implements Stack {
    // My contents, stored in a queue.
    private ArrayQueue queue;
    private int capacity;

    // Representation Invariant:

    // Construct me with capacity for n elements.
    public ArrayStack(int n){
        queue = new ArrayQueue(n);
        capacity = n;
    }

    // Push o onto me.
    // Precondition: I am not full.
    public void push(Object o){
    }

    // Pop off and return my top element.
    // Precondition: I am not empty.
    public Object pop (){}

    }

    // Return whether or not I am empty.
    public boolean isEmpty (){   
        return (queue.size() == 0);
    }

    // Return whether or not I am full.
    public boolean isFull (){   
        return (queue.size() == capacity);
    }

    // Empty me out.
    public void clear (){   
        // Optional: Write this method too.
    }
}