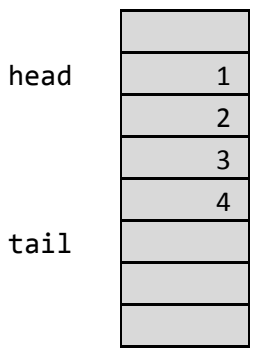
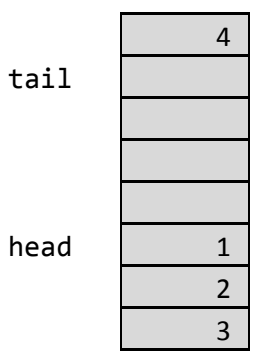
# CIRCULAR ARRAY QUEUES

**Part 1**: You will implement a queue as a *circular array* as follows: Use two index variables `head` and `tail` that contain the index of the next element to be removed and the next element to be added. After an element is removed or added, the index is incremented – as shown below:




After a while, the `tail` element will reach the top of the array. Then it "wraps around" and starts again at 0 – as in the next illustration:




For this reason, the array is called "circular".

```
public class CircularArrayQueue
{
      public CircularArrayQueue(int capacity) { . . .}
      public void add(Object x) { . . .}
      public Object remove() { . . .}
      public int size() { . . .}
      private int head;
      private int tail;
      private int theSize;
      private Object[] elements;
}
```

This implementation supplies a "*bounded*" queue – it can eventually fill up. That's OK – you'll see how to remove that limitation in the next part.

---

**Part 2**: The queue in Part 1 can fill up if more elements are added than the array can hold. Improve the implementation as follows: When the array fills up, allocate a larger array, copy the values to the larger array, and assign it to the `elements` instance variable. (*Hint*: You can't just copy the elements into the same position of the new array. Move the head element to position 0 instead.)

The tester for both parts is included below:

```
public static void main(String[] args) {
      CircularArrayQueue a = new CircularArrayQueue(10);
      a.add(1);
      a.add(2);
      a.add(3);
      a.add(4);
      a.add(5);
      a.add(6);
      a.add(7);
      a.add(8);
      a.add(9);
      System.out.println(a);
      System.out.println("NEXT: " +a.remove());
      System.out.println("NEXT: "+a.remove());
      System.out.println("NEXT: "+a.remove());
      System.out.println(a);
      a.add(10);
      System.out.println(a);
      a.add(11);
      System.out.println(a);
      a.add(12);
      System.out.println(a);
      System.out.println("NEXT: "+a.remove());
      System.out.println(a);
      a.add(13);
      System.out.println(a);
      a.add(14);
      System.out.println(a);
      a.add(15);
      System.out.println(a);
      System.out.println("NEXT: "+a.remove());
```

```
        System.out.println(a);
}
```