Queues

Datastructure for handling dynamic data where the element is pre-specified. A queue can be implemented with a linked list, or an array, and each presents interesting implementation details.

In problems we've looked at so far, we could add and delete from any point in our array or linked list. eg. remove item from middle of message board, add city to list middle

With a queue, data is always treated first-in, first-out (FIFO). This type of structure useful for applications where integrity in data ordering is critical.

We say enqueue to add to the queue and dequeue to remove from the queue.

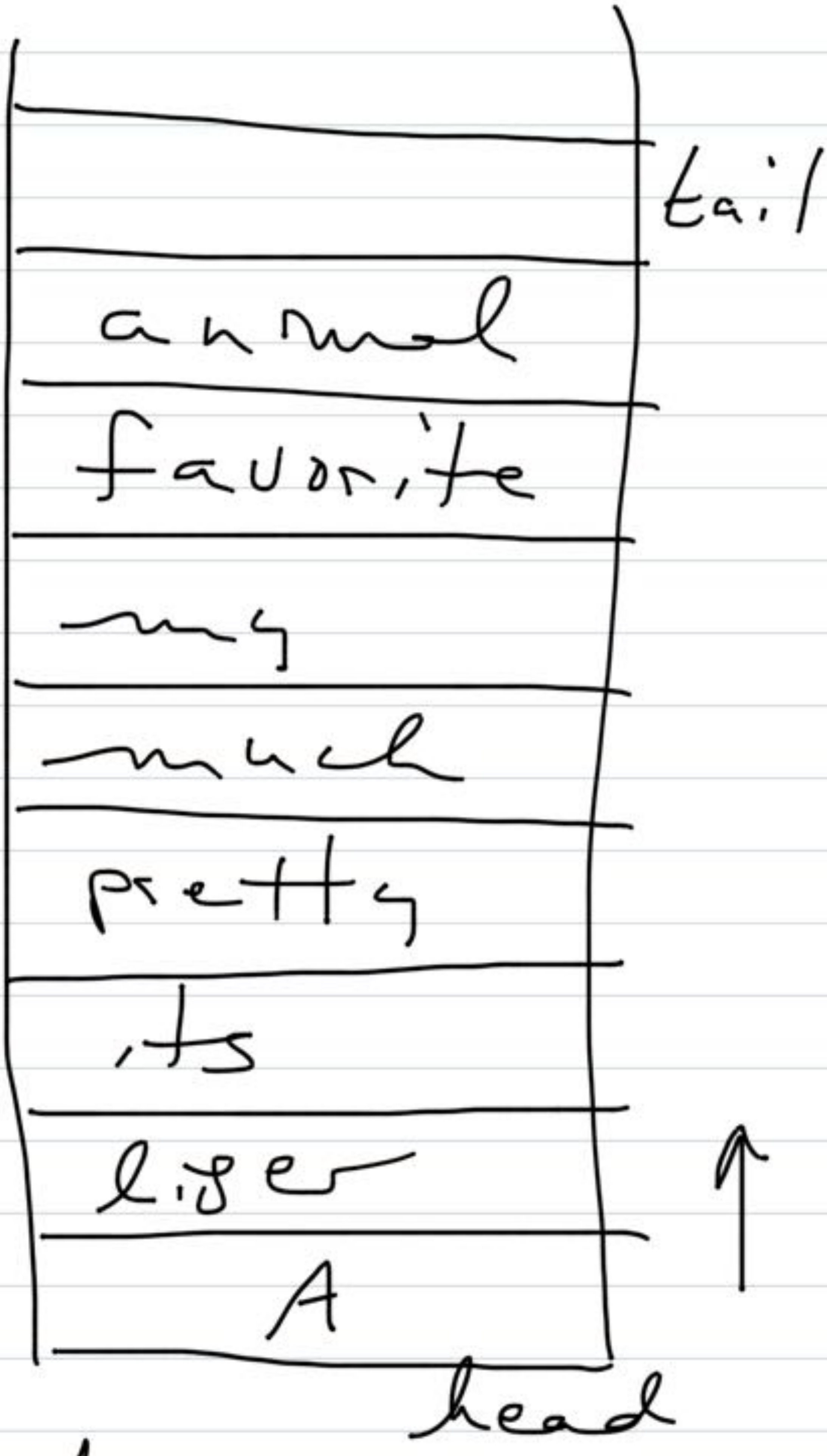
Example:

Read in:

"A liger its pretty much my favorite animal."

Enqueue at tail

Dequeue at the head



Like being at head or end of line.

initially head = tail

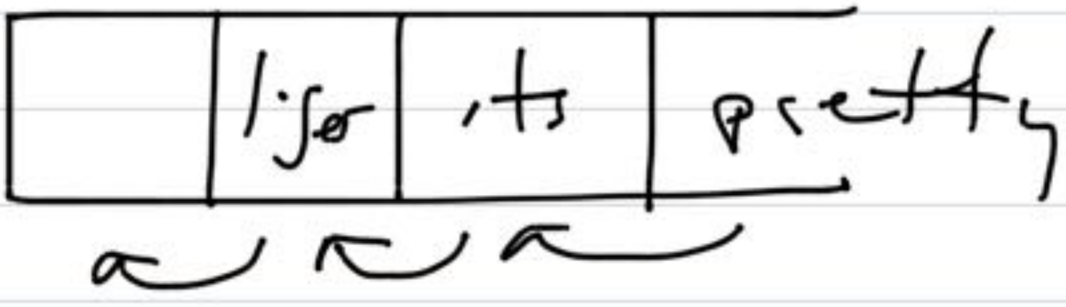
Simplest, but least efficient is to use an array, and shift items when head removed

A	liger	its	pretty	

Dequeue

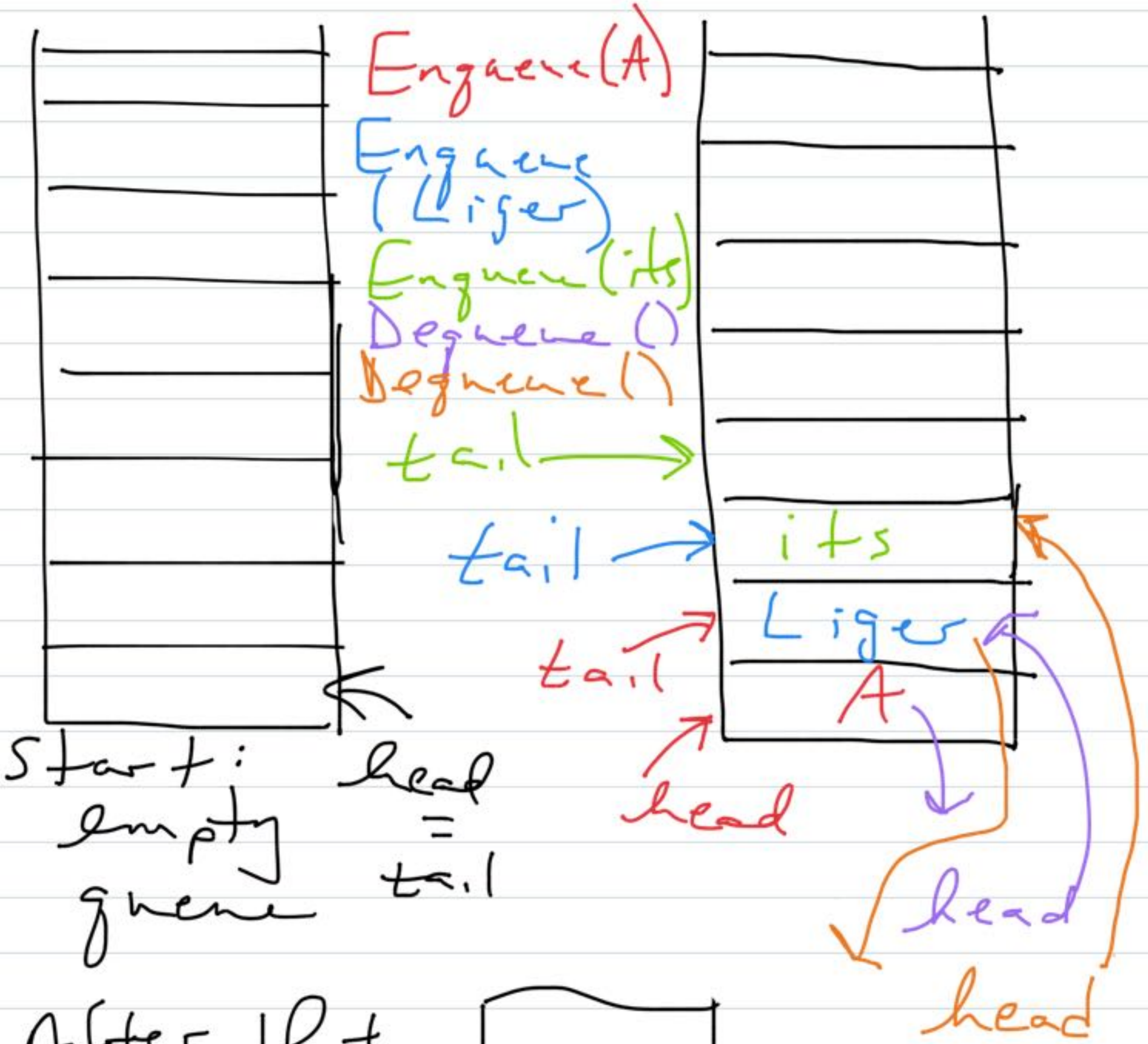
removes "A"

then shift



liger	its	pretty	

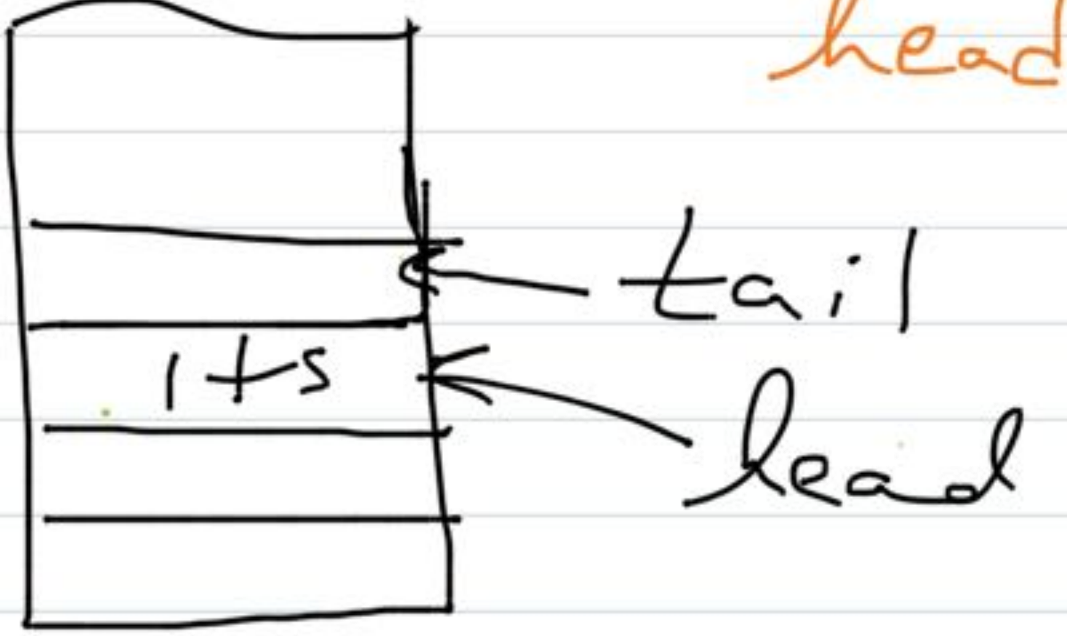
Shifting is costly. Use circular



Start: empty queue

After that sequence:

Queue has one item



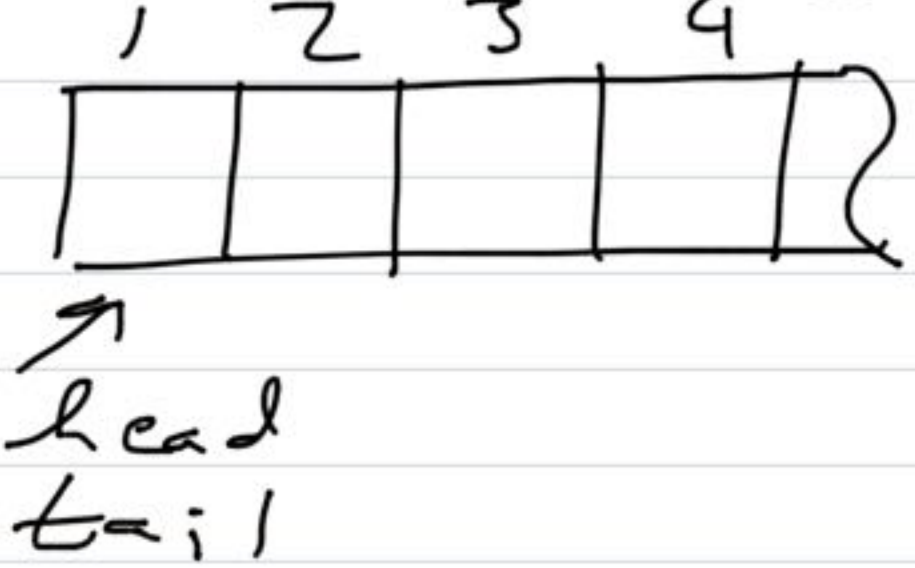
Different ways to implement a queue

Array - circular or
non-circular

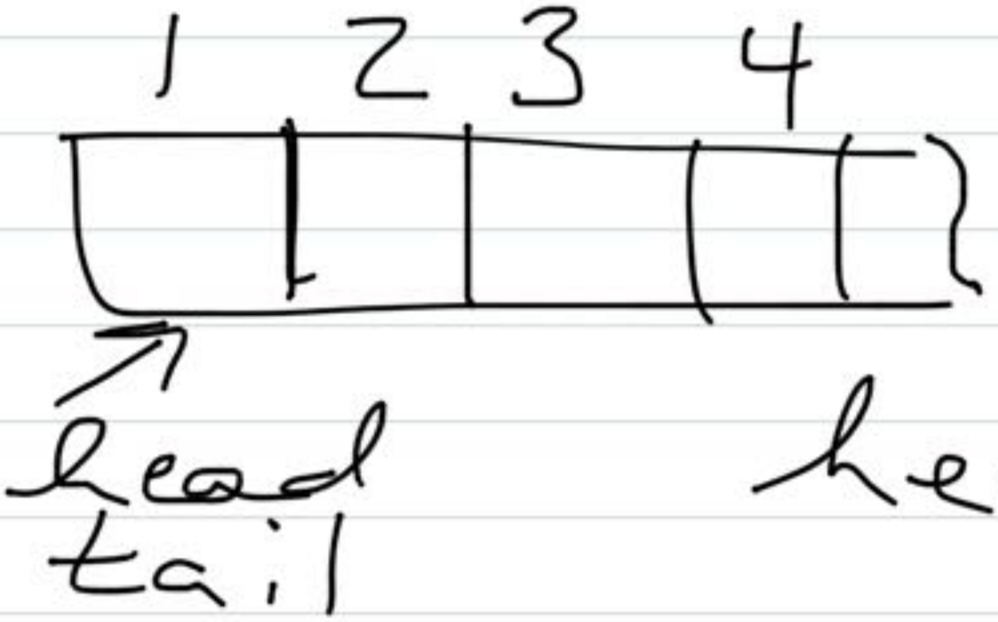
Linked list - no size limit

Array implementation (circular)
pseudcode:

Start with empty queue



head = tail = 1



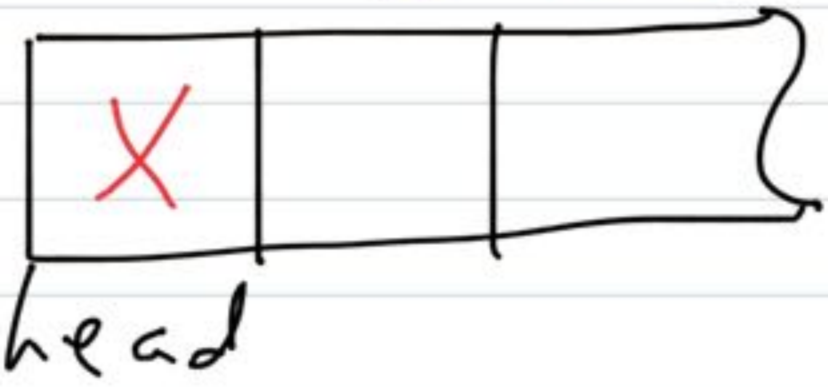
Pg 234, 235

head = tail = 1

Add an item (Enqueue)

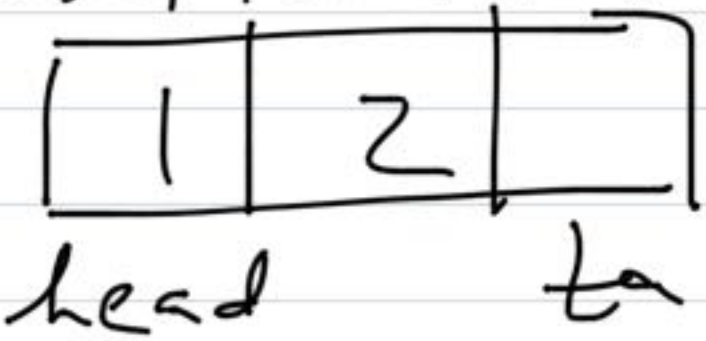
```
Enqueue(Q, x)
1 Q[Q.tail] = x
  if Q.tail == Q.length
      Q.tail = 1 //circular
  else
2     Q.tail = Q.tail + 1
```

← Q.tail = 2

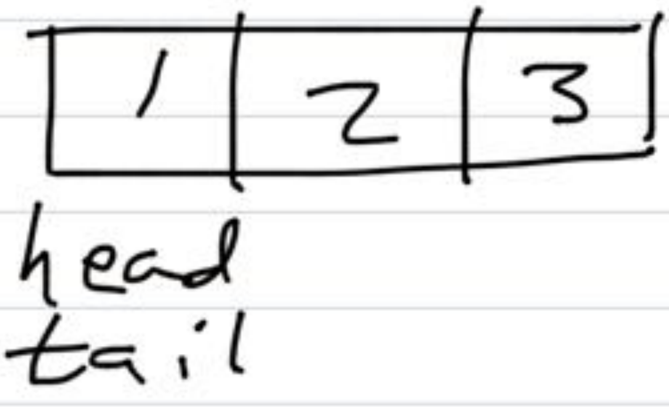


Missing from Enqueue Psendcode is not checking if queue is full before writing to queue. Circular is efficient, but requires some checking of head tail positions

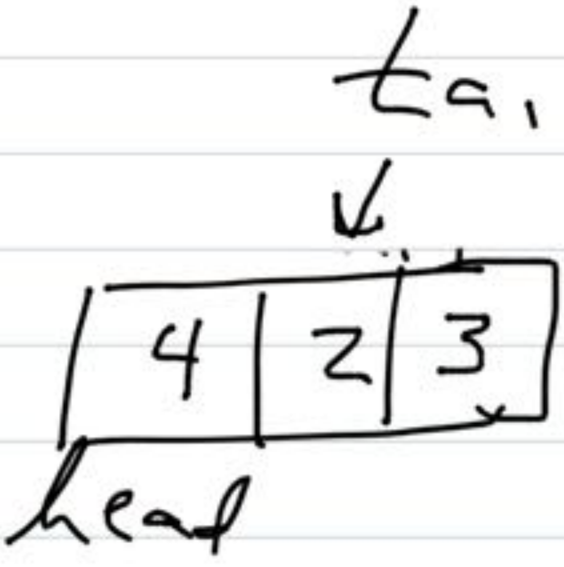
eg.



Enqueue(3)



Enqueue(4)

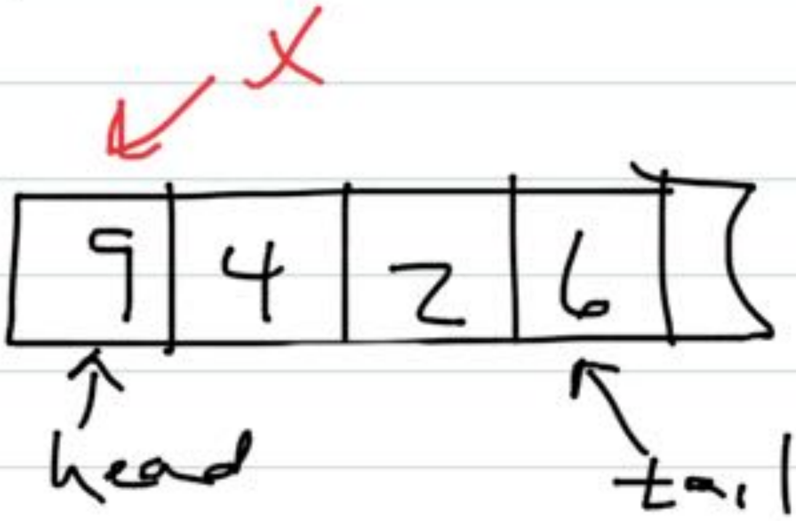


Data lost when 1 overwritten with 4

Removing an item

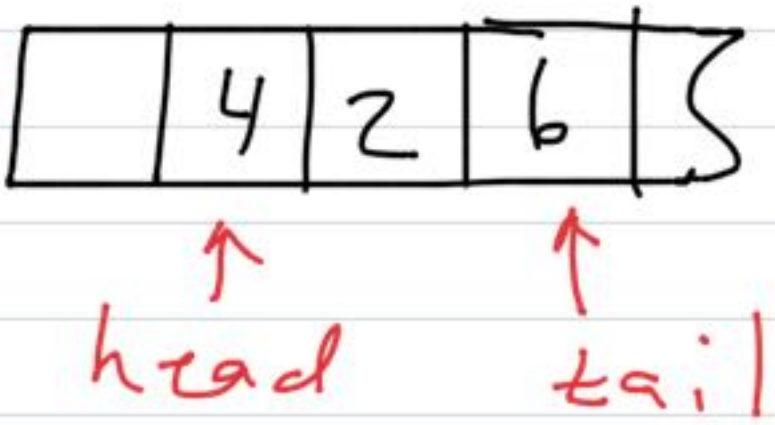
pseudocode

```
Dequeue(Q)
x = Q[Q.head]
if Q.head == Q.length
        Q.head = 1
else
        Q.head = Q.head + 1
```



x = 9

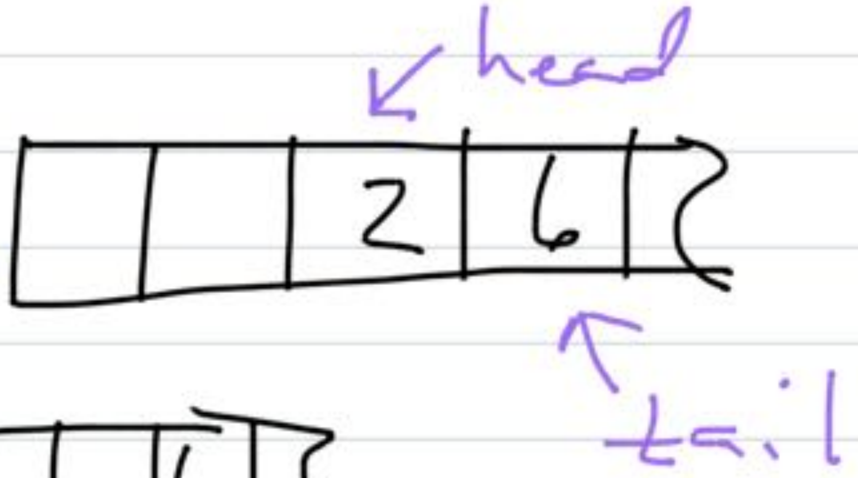
After dequeue:



Dequeue()

x = 4

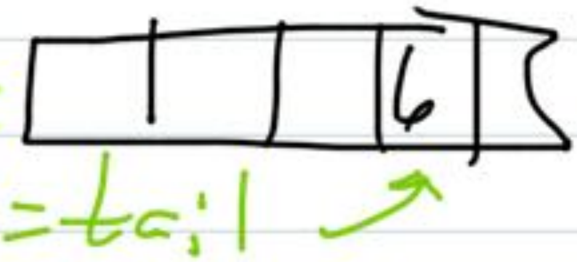
After dequeue



Dequeue

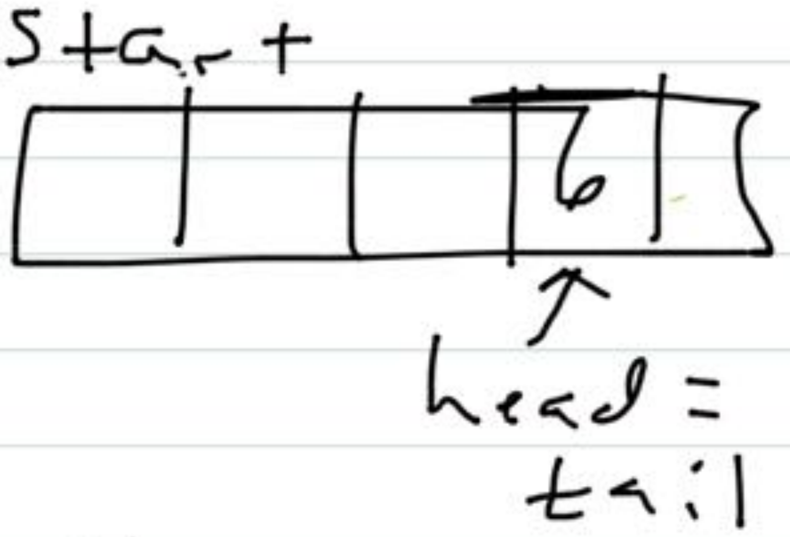
x = 2

After dequeue



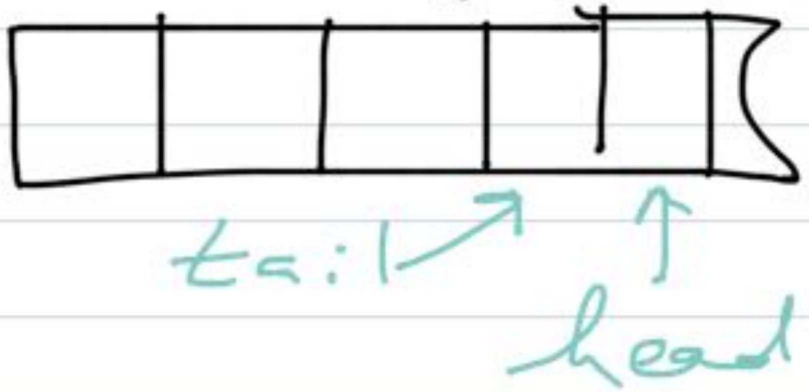
head = tail

Dequeue()
x = 6



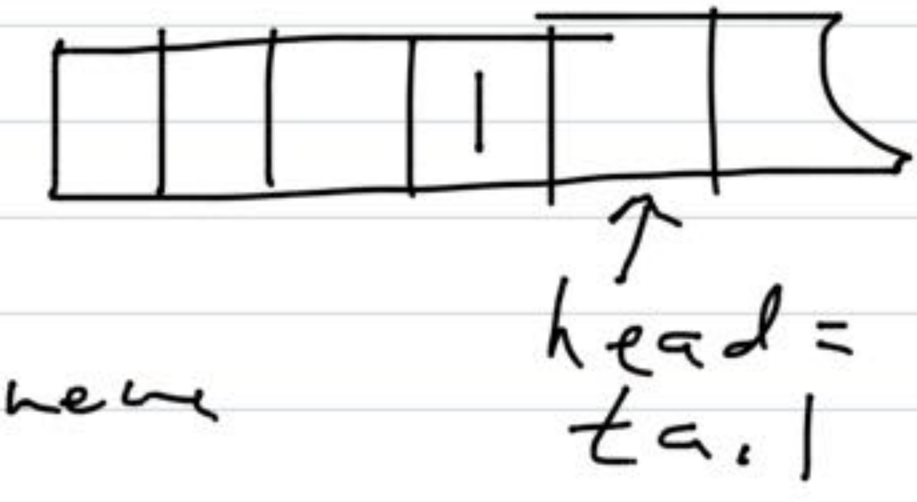
Without error checking:

Is this a problem?



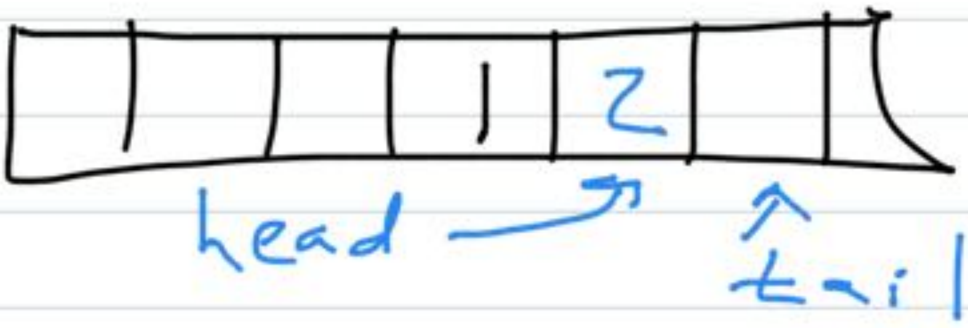
Dequeue()
- nothing there, may need additional code to track number of items in queue

Enqueue(1)



Dequeue()
- still won't dequeue the 1

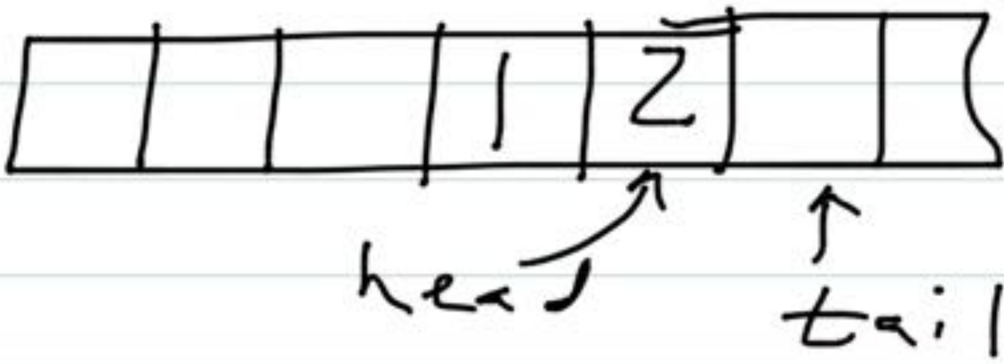
Enqueue(2)

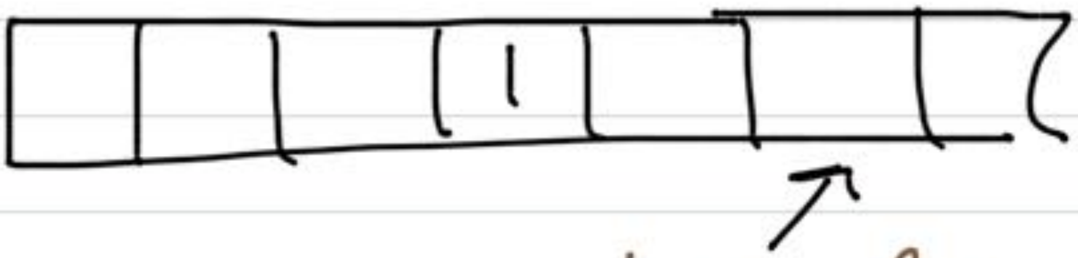


Dequeue() x=2

Dequeue()

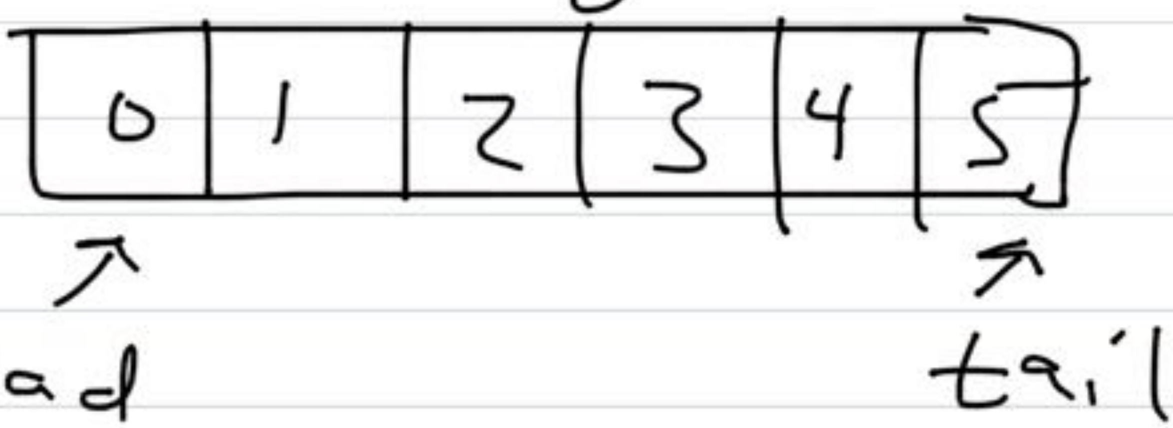
x = 2





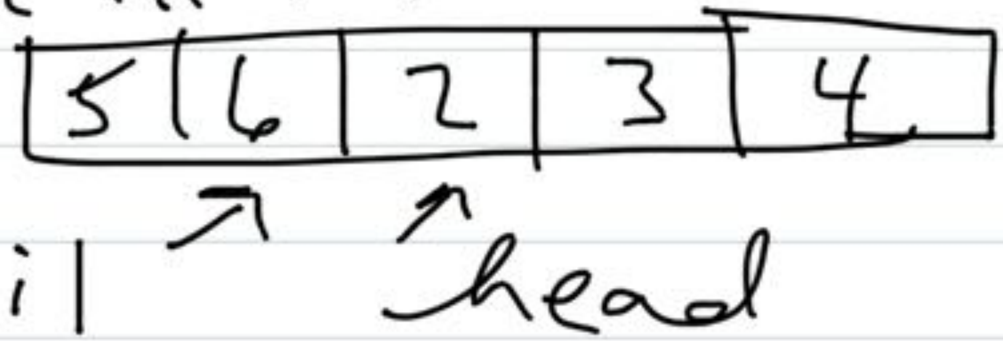
head =
tail

When is queue full?

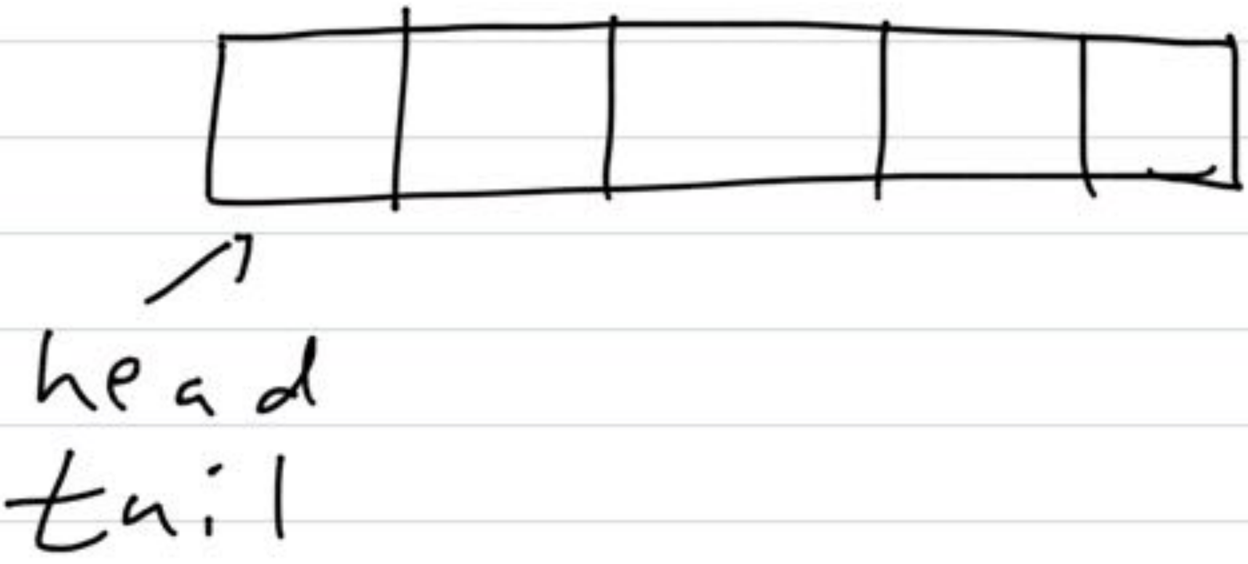


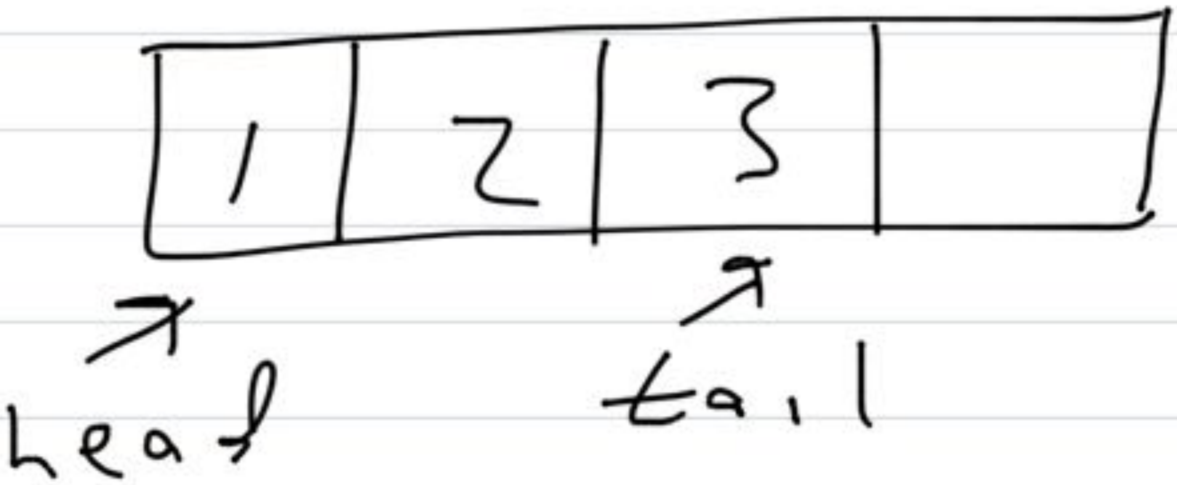
head = 0, tail = queueSize

head = tail + 1

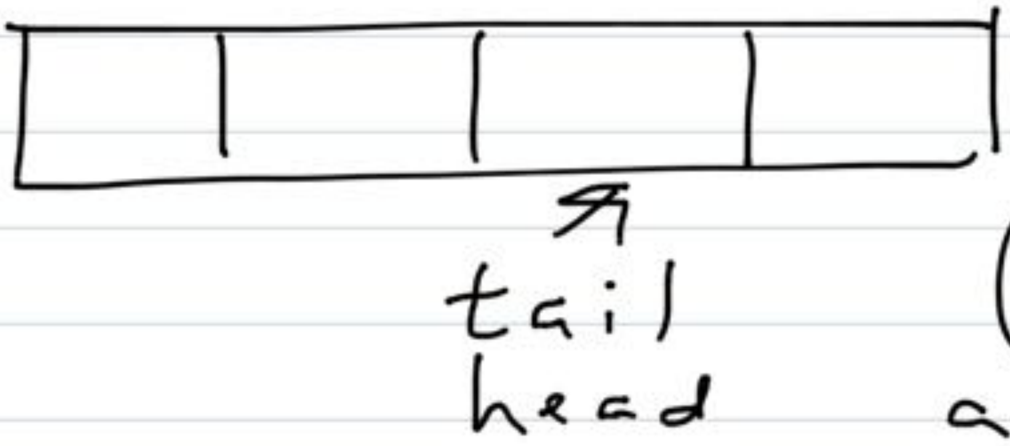


When is queue empty?





Dequeue 3 times



(Before advancing head)