Queues

Stacks are a last-in, first-out LIFO

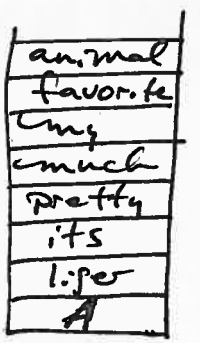
Queues are a first-in, first-out FIFO

For stacks, we use push, pop to add, remove

For queues, we use enqueue, dequeue to add, remove respectively

Queue has a head and a tail

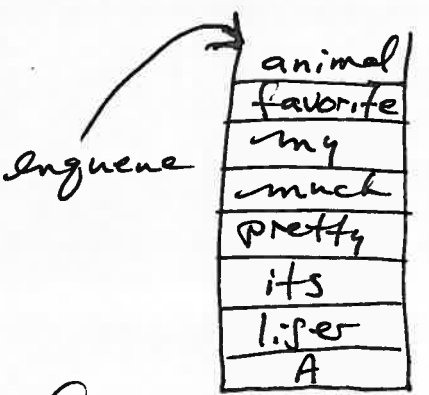
Read a word, add to position at the tail, like being at the end of the line



"A liger its pretty much my favorite animal."

Enqueue an item at the tail position.

Dequeue an item at the head position.



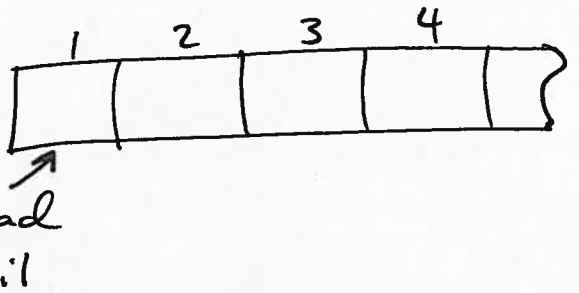
Like being at the head of the line.

dequeue

Array implementation of a queue

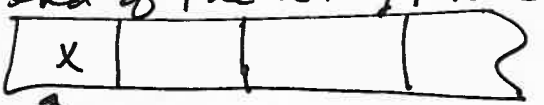
Start with an empty queue

head = tail = 1



Add an item (Enqueue)

```
Enqueue(Q, x)          //Q is array
                         x is item to add
  Q[Q.tail] = x        //add x to tail position
  if Q.tail==Q.length  //end of the array, no errors handling
    Q.tail = 1
  else
    Q.tail=Q.tail+1
```



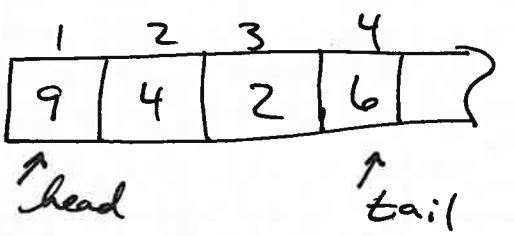
head, tail

// move tail +1, head doesn't change

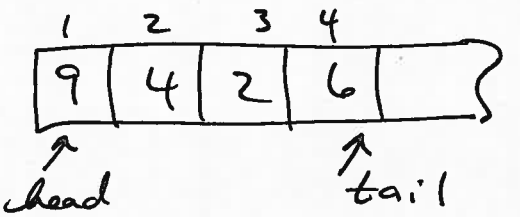
head tail

Remove an item (Dequeue)

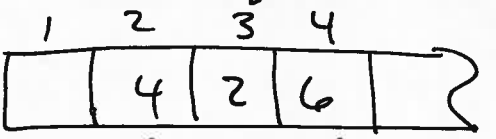
See dequeue next page



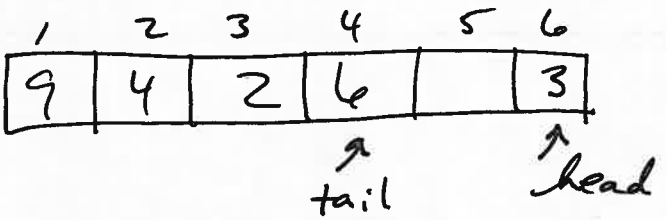
```
Dequeue (Q)
    x = Q[Q.head]          // grab item at head in array
    if Q.head == Q.length  // this is a circular queue, head and tail will wrap around
        Q.head = 1
    else
        Q.head = Q.head + 1
    return x
```



After dequeue

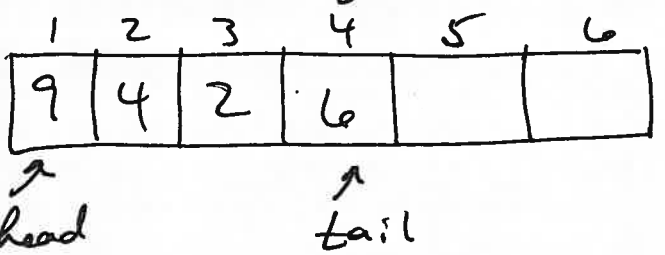


Wrap-around example

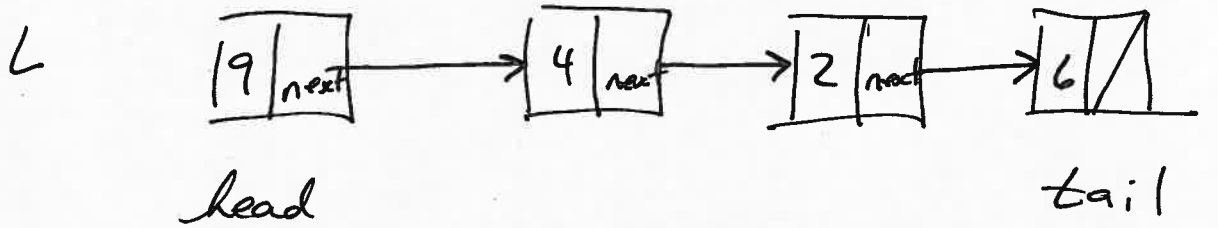


Call dequeue
head = length
Set head back to beginning with head = 1
tail unchanged

After dequeue

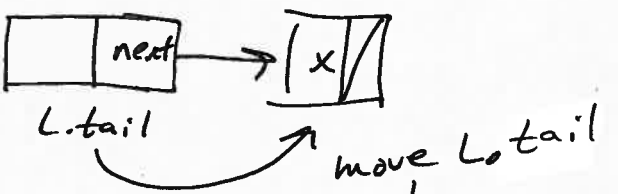


Linked list implementation of queue

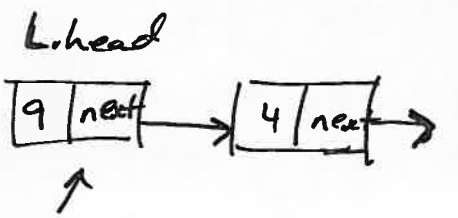



```
Enqueue (L, x)
    L.tail.next = x
    L.tail = x
```

// set next pointer of current tail to point to the new node




```
Dequeue (L)
    if L.head == NULL
        "underflow"
    else
        x = L.head
        L.head = L.head.next
    return x
```



After dequeue called,