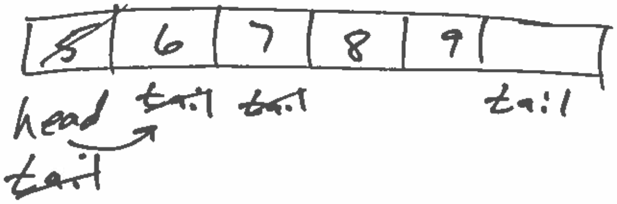
Circular array queue



Remove item and shift   o(n)

Move head   o(1)

Queue

private:
- head
- tail
- data
- currentSize - # of elements in the queue
- maxSize - size of queue

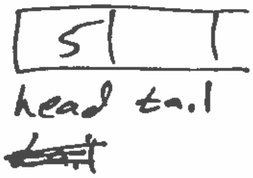
public:
- Queue() - constructor
- enqueue(value) - add to queue
- dequeue() - remove data from queue

dequeue()

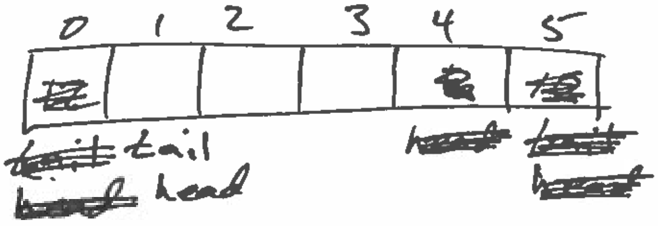
pre-cond: isEmpty() that returns true if currentSize = 0.

post-cond: returns data[head] value
head moves one position

```
    if(!isEmpty())
        value=data[head]
        currentSize--
        if(head == maxSize-1)
              head=0
        else
            head++
    return value
```



```
Enqueue(value)
    Pre-cond: value is valid, isFull exists
    Post-cond: value added to queue at tail
           position, tail position increases by 1
    if(!isFull())
        data[tail]=value
        currentSize++
        if(tail == maxSize-1)
            tail=0
        else
            tail ++
```



Enq(10)
maxSize=6

Enq(12) currentSize=3

```
deq() return 6
deq() return 10
deq() return 12
```

~~12~~	~~10~~	~~5~~	15	7	
[illegible]	~~t~~	~~t~~	~~t~~	~~t~~	t
[illegible]	~~h~~	~~h~~	h		

12, 10, 5

			15	7	

t

~~7~~	~~2~~	~~9~~	16	11	21
h	~~t~~	~~t~~	~~t~~	~~t~~	~~t~~
~~t~~	~~h~~	h			

7, 2, 9

			16	11	21

Linked List Queue

LL without a fixed size - singly LL

Enqueue(value)

Post-cond: new node add to queue at tail
position

```
node *n = new node(value, NULL)
if (tail != NULL)
    tail -> next = n
    tail = n
else    // true when the queue is empty
  tail = n
  head = tail
```

Dequeu( )

Post-cond: head of the queue returned
head points to next node in queue

```
node *n = NULL
if (head != NULL)
      n = head
      head = head -> next
else    // queue is empty
   tail = head
return n
```