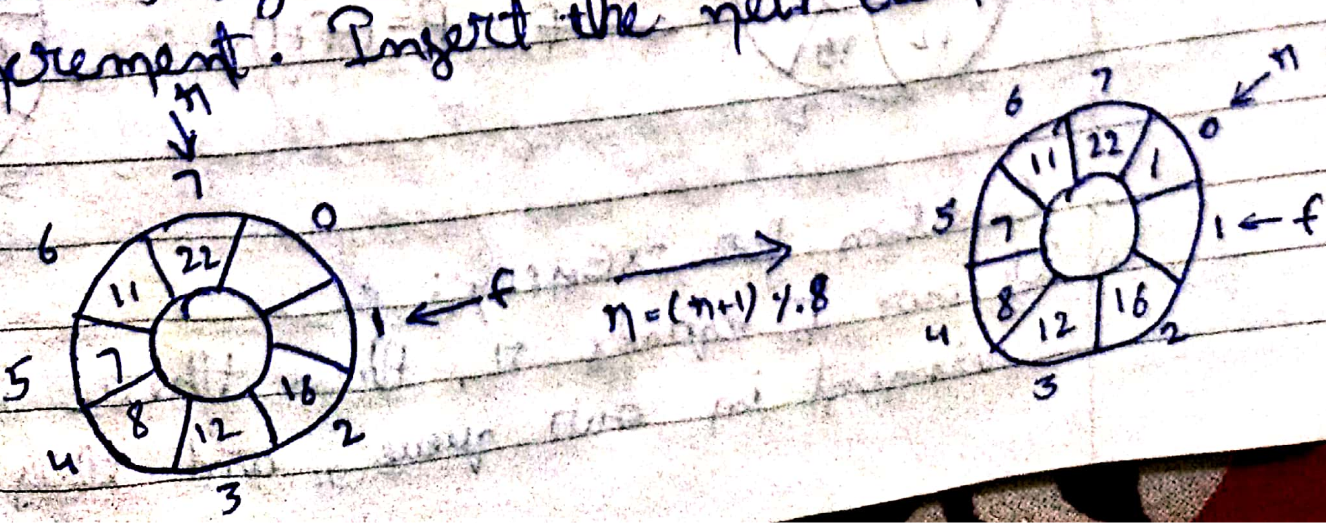
ENQUEUE & DEQUEUE in Circular queue: 43

ENQUEUE:

1) First, check if the queue is already not full. Here, the usual method to check the full ~~codition~~ condition wouldn't work. We will ~~m~~ now check if the next index to the rear is whether the front or not.

2) If it ~~means~~ ~~&~~ is, it means the queue is full. Because ~~from~~ front f represents the starting of the queue, and rear r represents the end. And the front coming next to the rear indicates that the queue is full. Therefore this is the case of queue overflow. Else just increment the rear by 1 and take its modulus by the circular queue's size. This is called the increment. Insert the new element there.

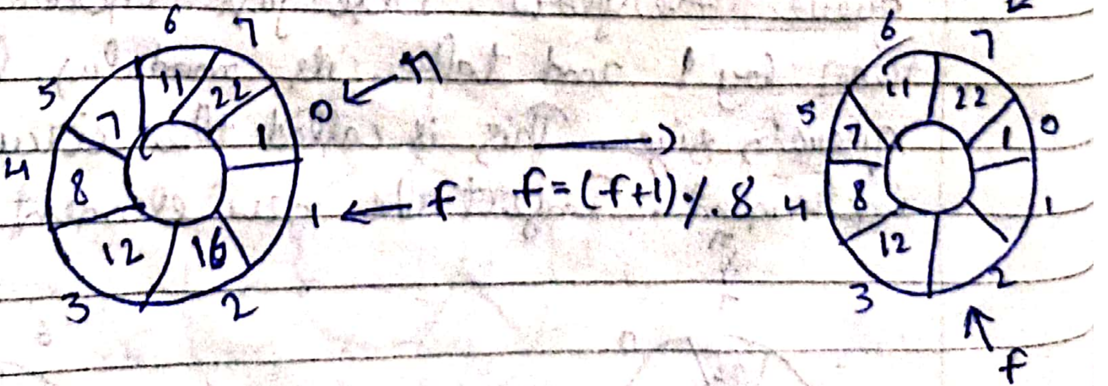


→ Now, since the f is just next to the r, the queue is full, and no more elements can be pushed.

DEQUEUE:

1) First, check if the queue is already not empty. Previously, we would just check if our front equals the rear, and if it did, we declared the queue empty. And you'll be amazed to know ~~the~~ that it works here as well. There are zero modifications here.

2) So, if the front ~~f~~ f equals the rear r, it is the case of queue underflow, else just increment f by 1 and take its modulus by the queue's size. While dequeuing, we store the element being removed and return it at the end.



→ Condition for isEmpty:

1) If our f equals r, then there is no element in our queue, and this is

the case of an empty queue.

→ Condition for isFULL :

1) If our (r+1)%SIZE equals f, then there is no space left in our queue, and this is the case of a full queue.