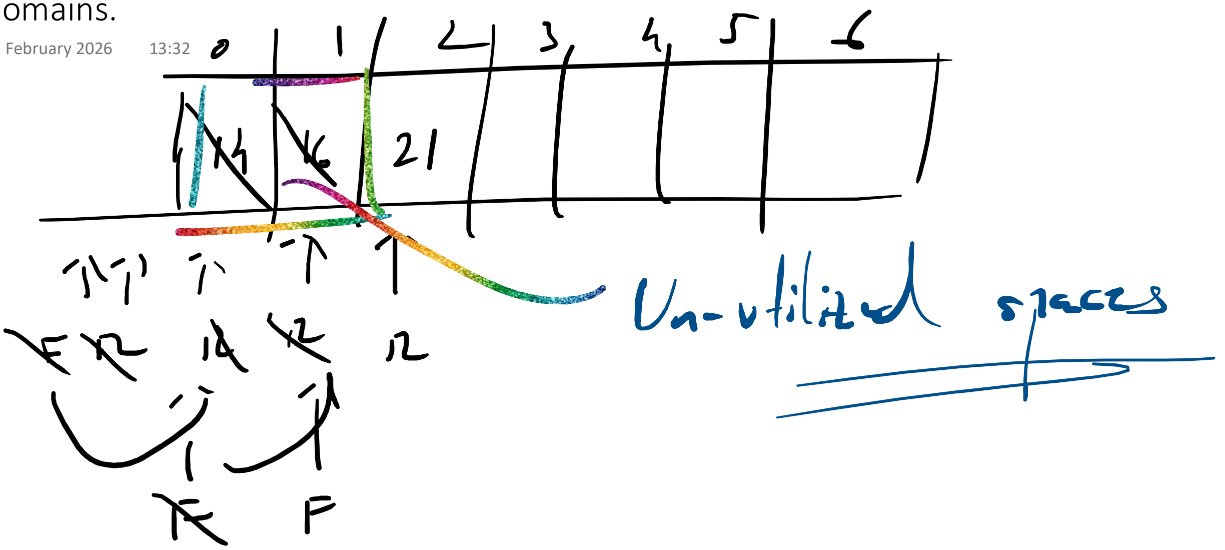
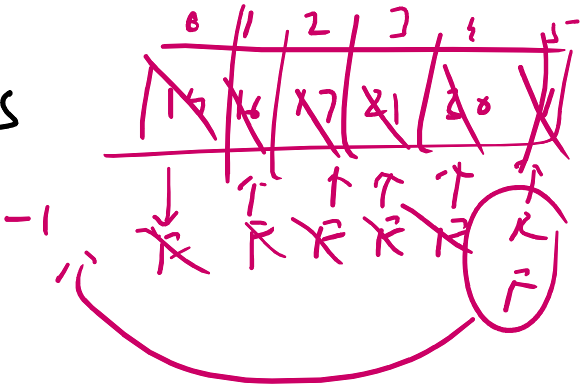
02-02-2026 - Apply queue operations for problems in different domains.

02 February 2026 13:32

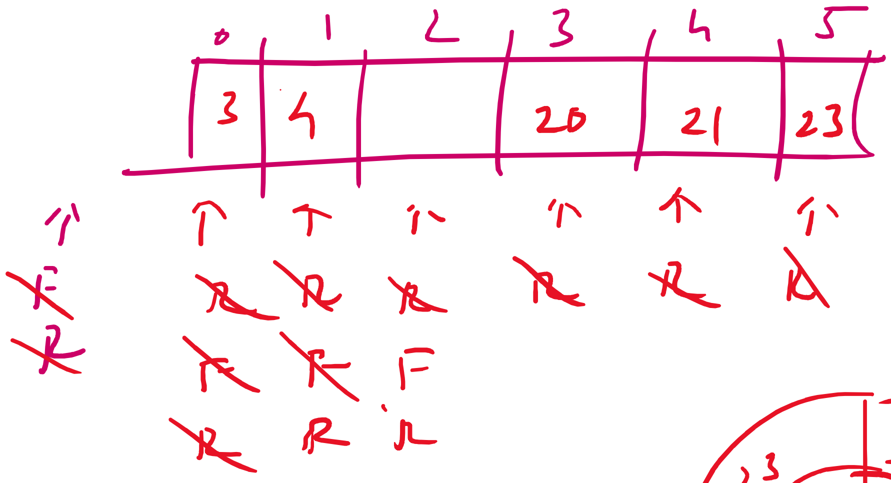


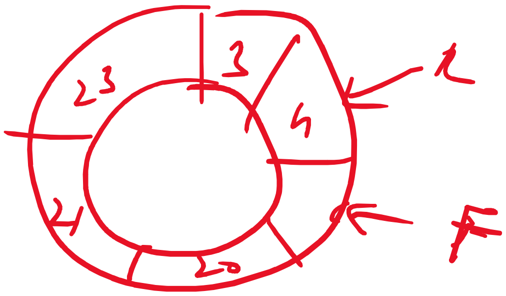
=> 1) Resetting Pointers

2) Circular Queue



=> Circular Queue





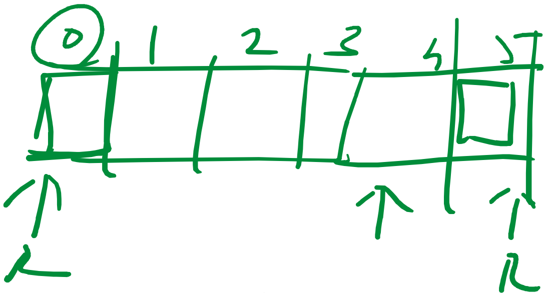
$Rear = (Rear + 1) \% size$

$$Rear = (Rear + 1) \% \; size$$

0	$(0+1) \% \; 6$	1
1	$(1+1) \% \; 6$	2
2	$(2+1) \% \; 6$	3
3	$(3+1) \% \; 6$	4
$\boxed{4}$	$(4+1) \% \; 6$	$\boxed{5}$
$\boxed{5}$	$(5+1) \% \; 6$	0

Remainder

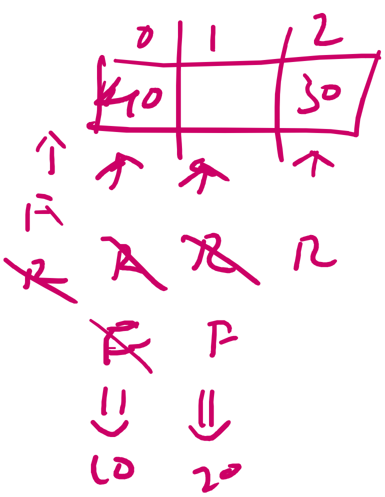
$$6 \overline{) \; 2}$$
$$- \; 0$$
$$\overline{2}$$



```
void enqueue(struct q, int x) {
    if ((q->rear+1)% q->size)==q->front){
        print ( Queue full)
    else
        q->rear = [(q->rear+1) % q->size;]
        q->S [q->rear] = x;

    }
```

```
=>  int dequeue (struct q ) {
        int x = -1;
        if (q-> front == q-> rear) {
            print (queue empty)
        else {

            q-> front = (q->front+1) % q->size;
            x = q-> Q [q->front];

        }
        return x;
```

=>



size = 3

$R = (R+1) \% N$

0	$(0+1) \% 3 = 1$
1	$(1+1) \% 3 = 2$
2	$(2+1) \% 3 = 0$

enqueue(10)

enqueue(20)

enqueue(30)

dequeue( )

dequeue( )

enqueue(40)