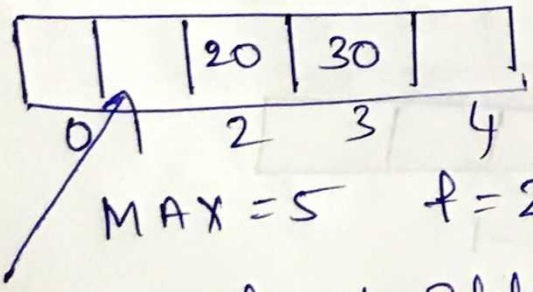
Enqueue — Double Ended Queue

* Enqueue means inserting an element in the queue.
* There are two ways to insert an element in the double ended queue.
  1. Insert an element at the front end.
  2. Insert an element at the rear end.
* We will be Implementing
  - EnqueueFront()
  - EnqueueRear()

Enqueue Front

Case 1 - Random Insertion

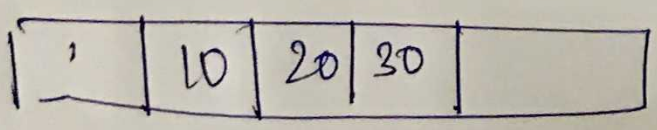


MAX = 5 f = 2 , r = 3

Enqueue front shld be done here.

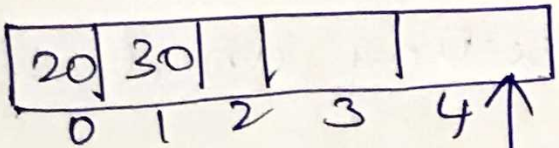
* In order to insert an element in the front, we have decrement the front by 1.
* Here is the code.

```
front = front - 1
dequeue[front] = data;
data = 10
```



MAX = 5
front = 1 rear 3

Case 2:- front is zero.

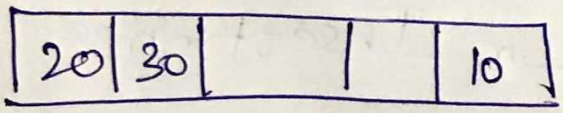
deque. 

MAX=5 f=0 , r=1

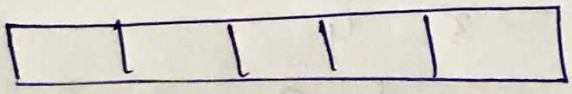
here the element to be inserted, it is a double ended queue.

* Here is the code for this case

```
if(front==0)
    front=MAX-1;
deque[front]=data.
```



Case 3:- front is -1

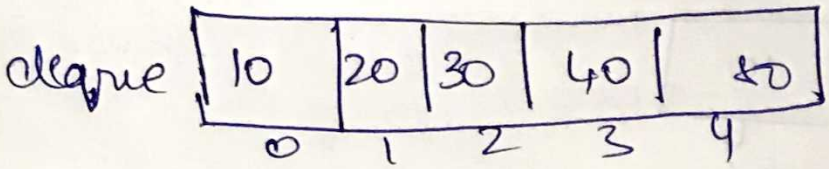
deque 

f=-1 , r=-1

Here is the code for this case

```
if(front==-1)
{
    front=0;
    rear=0;
}
deque[front]=data;
```

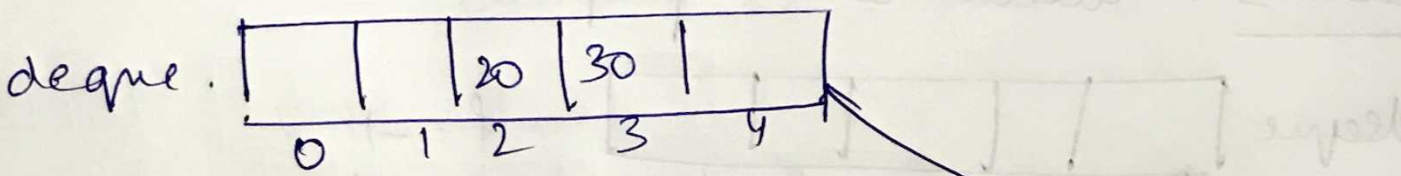
Case 4:- Queue is full.




```
if(isfull())
{
    printf("Queue overflow\n");
    exit(1);
}
```

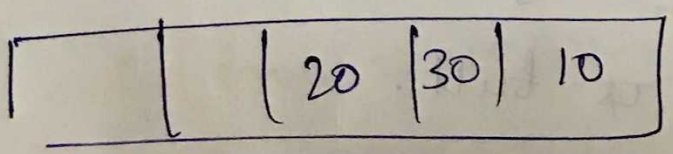
Enqueue ~~front~~ rear()

Case 1:- Random Insertion

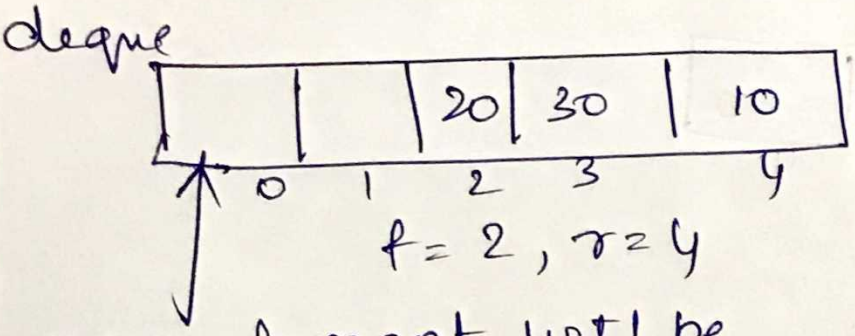


* Here is the code

```
rear = rear + 1;
deque[rear] = data;
```



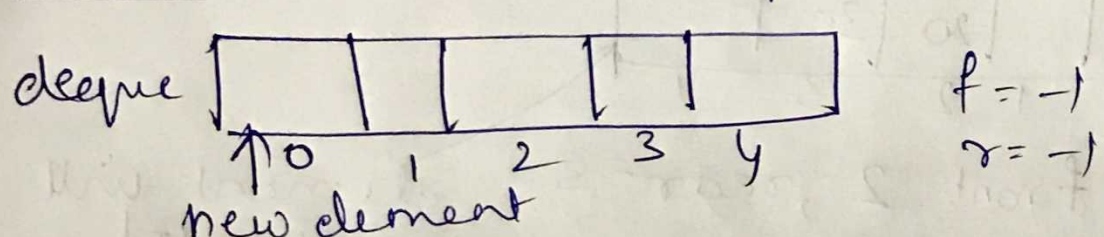
Case 2:- Rear is holding the last index.



* Here is the code

```
if (rear == MAX-1)
    rear = 0;
deque[rear] = data;
```

Case 3 :- Queue is Empty.




```
if (front == -1)
{
    rear = 0;
    front = 0;
}
deque[rear] = data;
```

Case 4 :- Queue is full.

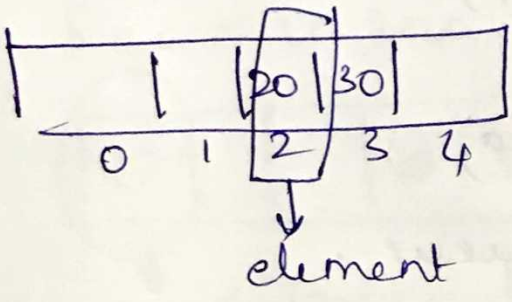
isfull()

Dequeue

1. Delete an element from the front end
   dequeuefront( )
2. Delete an element from the rear end
   dequeuerear( ).

dequeueFront( )

Case 1 :- Random Deletion

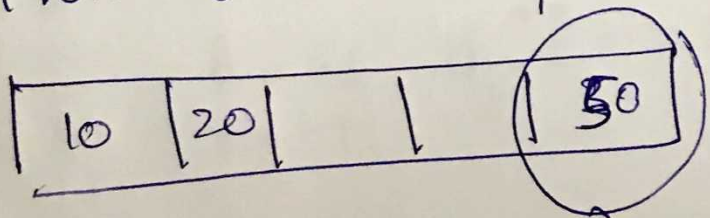


f = 2, r = 3

Code :-

```
int data;
data = deque[front];
front = front+1;
return data;
```

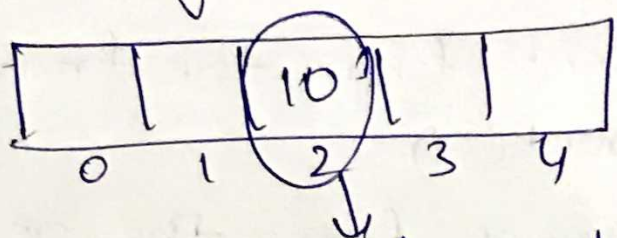
Case 2 :- front is holding the last index



f = 4
r = 1

```
if(front == MAX-1)
    front = 0;
return data;
```

Case 3:- Single element in the Queue

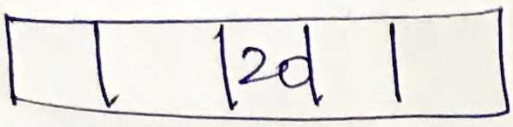



```
data= deque[front];
if(front==rear)
{
    front=-1;
    rear=-1;
}
return data;
```

Case 4:- Empty Queue.

isEmpty()

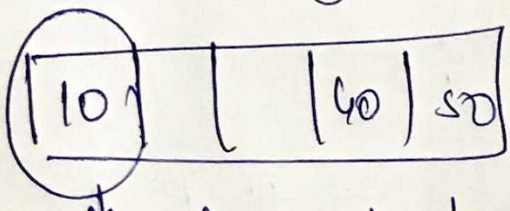
dequeue Rear ()



Case 1:- Random Deletion

```
int data;
data = deque[rear];
rear = rear - 1;
return data;
```

Case 2:- Rear is zero.



element to be deleted

```
data = deque[rear]
if (rear == 0)
    rear = MAX - 1;
return data;
```

Case 3:- Single element

```
f = -1
r = -1
if (f == r)
```