QUEUE

FIFO - First in First out
LILO - Last in Last out

0	1	2	3	4	5

Enque → Insertion

Deque → Deletion

Display → F ->R

Common Terms

Front to rear

Here front = -1 and rear = -1 since front and rear are at same position and queue is empty.If rear is at (size-1)th position, queue is full and rear is the end point.

Enque Operation: In this in general , we insert elements from front to rear, we always insert the elements at the place of rear

0	1	2	3	4	5
F R					

Front = -1 and rear = -1

At this case, when queue is empty:-

enque(10)

rear++ i.e; **rear = 0**

front++ i.e; **front=0**

que[rear]=val

10					
0	1	2	3	4	5
F	R				

At this case, when queue is not empty:-

Enque operation:-

rear++

que[rear]=val

Enque operation when rear = size-1

Prints **queue is full** as rear reached the end of the queue if we try insert another element into the queue.

Enque(10)		Enque(20)		Enque(30)		Enque(40)		Enque(50)		Enque(60)		F=0 R=SIZE-1
5									R	60	R	
4							R	50		50		
3					R	40		40		40		
2				R	30	30		30		30		
1		R	20		20	20		20		20		
0	10	F	10	F	10	F	10	F	10	F	10	F

Deque Operation:- In this in general , we delete elements from front to rear, we always delete the elements at the place of front.

0	1	2	3	4	5

Front = -1 and rear = -1

At this case, when queue is empty:-

It prints **queue is empty** since there are no elements to delete

In all these cases,
Val = queue[front] ; front++;
and Finally return val to main function

In this case,
When front = rear
val = queue[front];F=-1;R=-1 ; Finally return val to

	Initial		deque()		deque()		deque()		deque()		deque()		F= -1 R=-1
5	60	R	60	R	60	R	60	R	60	R	60	F=R	
4	50		50		50		50		50	F			
3	40		40		40		40	F					
2	30		30		30	F							
1	20		20	F									
0	10	F											

Output :-

10 20 30 40 50 60

Display operation :- In this in general , we display elements from front to rear, we always display the elements using the loop from running from front to rear.

0	1	2	3	4	5

Front = -1 and rear = -1

At this case, when queue is empty:-

It prints **queue is empty** since there are no elements to display

	display()		display()		display()		display()		display()		display()	
R=5	60		60		60		60		60		60	i
4	50		50		50		50		50	i	50	
3	40		40		40		40	i	40		40	
2	30		30		30	i	30		30		30	
1	20		20	i	20		20		20		20	
F=0	10	i	10		10		10		10		10	

Output:- 10 20 30 40 50 60

Circular Queue:- Circular Queue is a linear data structure which follows the First In First Out principle and the last position of the queue is connected back to the first position of the queue to make it a circle.

Enque Operation: In this in general , we insert elements from front to rear, we always insert the elements at the place of rear

0	1	2	3	4	5
F R					

Front = -1 and rear = -1

At this case, when queue is empty[F=-1&R=-1] :-
enque(10)

rear++ i.e; **rear = 0**

front++ i.e; **front=0**

que[rear]=val

10					
0	1	2	3	4	5
F	R				

At this case, when queue is not empty:-

Enque operation:-

rear = (rear+1)%size

que[rear]=val

Enque operation when rear = front -1 or (rear = size-1 when front = 0) :-

Prints **queue is full** as rear reached the end of the queue, if we try insert another element into the queue.

Here I am considering Front at 2nd position and Rear at 3rd position

enque(14)			enque(15)		enque(16)		enque(17)		enque(18)	
5				R	15		15		15	
4		R	14		14		14		14	
3	13		13		13		13		13	
2	12	F	12	F	12	F	12	F	12	F
1								R	17	R
0						R	16		16	

enque(18) is not possible since rear reached the (front – 1)th position

Deque Operation:- In this in general , we delete elements from front to rear, we always delete the elements at the place of front.

0	1	2	3	4	5

Front = -1 and Rear = -1

At this case, when queue is empty:-

It prints **queue is empty** since there are no elements to delete

In all these cases when queue is not empty :-
val = queue[front] ; front = (front+1) %size ;
and Finally return **val** to main function

In this case,
When front = rear
val = queue[front];
F=-1;R=-1 ; Finally return **val** to main

	Initial		deque()		deque()		deque()		deque()		deque()		F= -1 R=-1
5	15		15		15		15	F					
4	14		14		14	F							
3	13		13	F									
2	12	F											
1	17	R	17	R	17	R	17	R	17	R	17	F R	
0	16		16		16		16		16	F			

Output :- 12 13 14 15 16 17

Display operation :- In this in general , we display elements from front to rear, we always display the elements using the loop running from front to rear. But instead of incrementing the i we take i = (i + 1) % size as updation. Since it is the circular queue 0 will become the next position after the size-1 the position of a queue.

0	1	2	3	4	5

Front = -1 and rear = -1

At this case, when queue is empty :-

It prints **queue is empty** since there are no elements to display

Here I am considering Front at 4th position and rear at 3rd position

	display()		display()		display()		display()		display()		display()	
5	60		60	i	60		60		60		60	
F=4	50	i	50		50		50		50		50	
R=3	40		40		40		40		40		40	i
2	30		30		30		30		30	i	30	
1	20		20		20		20	i	20		20	
0	10		10		10	i	10		10		10	

Output:- 50 60 10 20 30 40

Double Ended Queue:-

Size =6

0	1	2	3	4	5

Rear = -1

Front = -1

Different cases:

1.Enque _ rear
2.Deque _ front
3.Enque _ front
4. Deque _ rear
5.Display
1,2 &5 are same as enque and deque operations in circular queue

Here _ = at

Enque at rear Operation: In this in general , we insert elements from front to rear, we always insert the elements at the place of rear

0	1	2	3	4	5
F R					

Front = -1 and rear = -1

At this case, when queue is empty[F=-1&R=-1] :-
enque_at_rear(10)

rear++ i.e; **rear = 0**

front++ i.e; **front=0**

que[rear]=val

10					
0	1	2	3	4	5
F	R				

At this case, when queue is not empty:-

Enque at rear operation:-

rear = (rear+1)%size

que[rear]=val

Enque at rear operation when rear = front -1 or (rear = size-1 when front = 0):-

Prints **queue is full** as rear reached the end of the queue if we try insert another element into the queue.

Here I am considering Front at 2nd position and Rear at 3rd position

	enque(14)		enque(15)		enque(16)		enque(17)		enque(18)	
5				R	15		15		15	
4		R	14		14		14		14	
3	13		13		13		13		13	
2	12	F	12	F	12	F	12	F	12	F
1								R	17	R
0						R	16		16	

enque(18) is not possible since rear reached the (front – 1)th position

Deque at front Operation:- In this in general , we delete elements from front to rear, we always delete the elements at the place of front.

0	1	2	3	4	5

Front = -1 and Rear = -1

At this case, when queue is empty:-

It prints **queue is empty** since there are no elements to delete

In all these cases,
Val = queue[front] ; front = (front+1) %size ;
and Finally return val to main function

In this case,
When front = rear
val = queue[front];
F=-1;R=-1 ; Finally return val to main

	Initial		deque()		deque()		deque()		deque()		deque()		F= -1 R=-1
5	15		15		15		15	F					
4	14		14		14	F							
3	13		13	F									
2	12	F											
1	17	R	17	R	17	R	17	R	17	R	17	F R	
0	16		16		16		16		16	F			
Output :-			12		13		14		15		16		17

Display operation :- In this in general , we display elements from front to rear, we always display the elements using the loop from running from front to rear. But instead of incrementing the i we take i=(i+1)%size as updation . Since it is the circular queue 0 will become the next position after the size-1 the position of a queue.

0	1	2	3	4	5

Front = -1 and rear = -1

At this case, when queue is empty:-

It prints **queue is empty** since there are no elements to display

Here I am considering Front at 4th position and rear at 3rd position

	display()		display()		display()		display()		display()		display()	
5	60		60	i	60		60		60		60	
F=4	50	i	50		50		50		50		50	
R=3	40		40		40		40		40		40	i
2	30		30		30		30		30	i	30	
1	20		20		20		20	i	20		20	
F=0	10		10		10	i	10		10		10	
Output:-	50		60		10		20		30		40	

Enque at front Operation: In this, we always insert the elements at the place of front

0	1	2	3	4	5
F R					

Front = -1 and rear = -1

At this case, when queue is empty[F=-1&R=-1] :-
enque_at_front(12)

rear++ i.e; **rear = 0**

front++ i.e; **front=0**

que[front]=val

12					
0	1	2	3	4	5
R					F

When Front = 0
front = size – 1
que[front] = val;

At this case, when queue is not empty:-

Enque at rear operation:-

decrement front by 1 i.e; **front- -**

que[front]=val

Enque at rear operation when rear = front -1 or (rear = size-1 when front = 0):-

Prints **queue is full** as rear reached the end of the queue if we try insert another element into the queue.

Here I am considering Front at 2nd position and Rear at 3rd position

	enque(13)		enque(14)		enque(15)		enque(16)		enque(17)		enque(18)	
5		F	13		13		13		13		13	
4				F	14		14		14		14	
3						F	15		15		15	
2								F	16		16	
1										F	17	F
0	12	R	12	R	12	R	12	R	12	R	12	R

enque(18) is not possible since front reached the (rear + 1)th position.

Deque at rear Operation:- In this, we always delete the elements at the place of rear.

0	1	2	3	4	5

Front = -1 and Rear = -1

At this case, when queue is empty:-

It prints **queue is empty** since there are no elements to delete

In all these cases,
Val = queue[rear] ; decrement rear by 1 i.e; **rear- - ;**
and Finally return val to main function

When rear = 0
val = queue[rear];
rear = size-1
Finally return val to main

In this case,
When front = rear
val = queue[rear];
F=-1;R=-1 ; Finally return val to main

	Initial		deque()		deque()		deque()		deque()		deque()		F= -1 R=-1
5	13		13	R									
4	14		14		14	R							
3	15		15		15		15	R					
2	16		16		16		16		16	R			
1	17	F	17	F	17	F	17	F	17	F	17	F R	
0	12	R											

Output :- 12 13 14 15 16 17