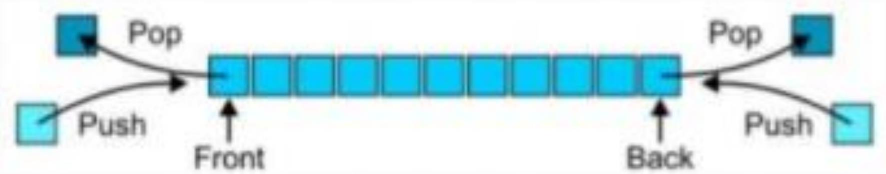
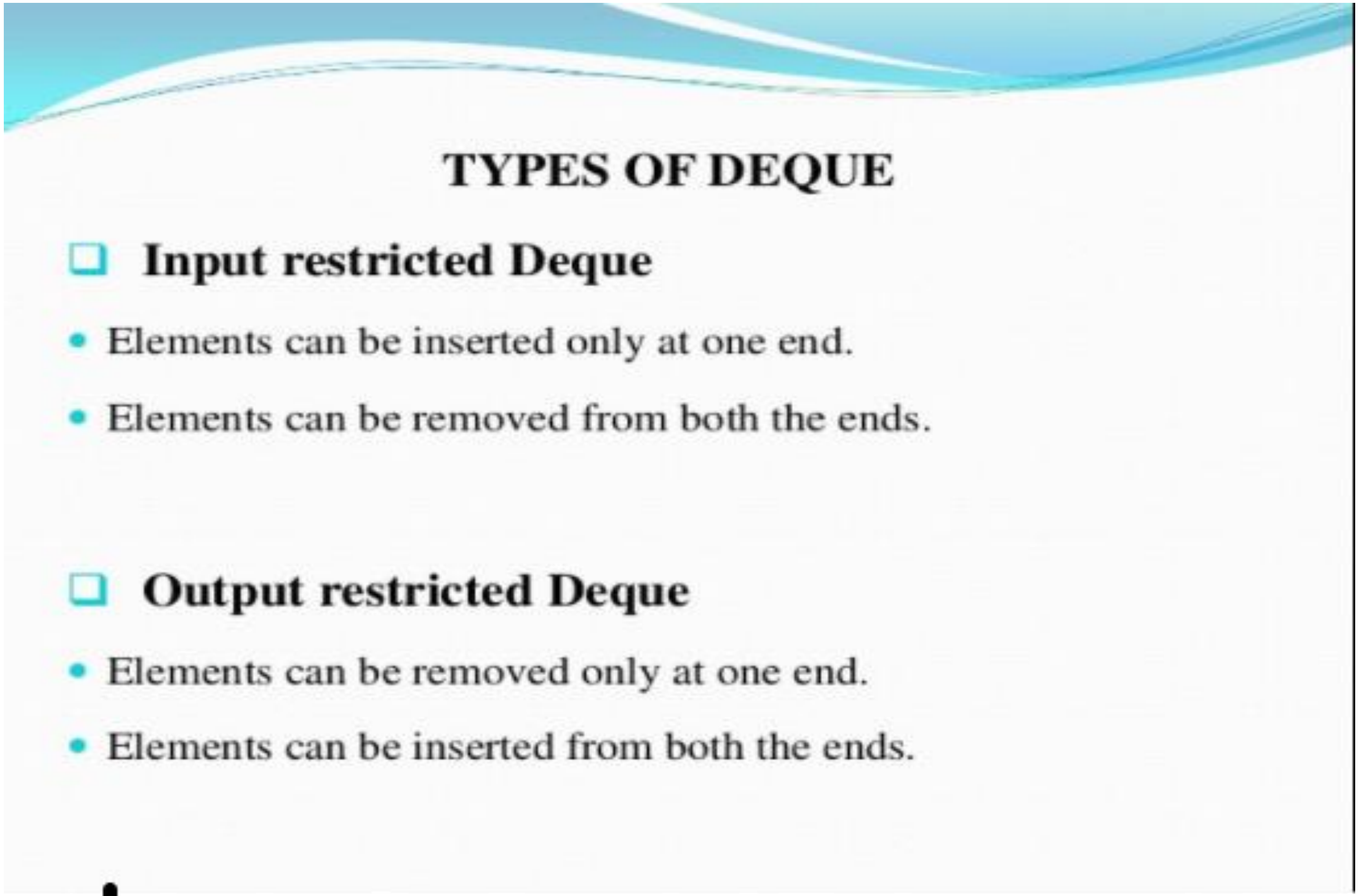
Double Ended Queue

Double-Ended Queue

- A Deque or deck is a double-ended queue.
- Allows elements to be added or removed on either the ends.





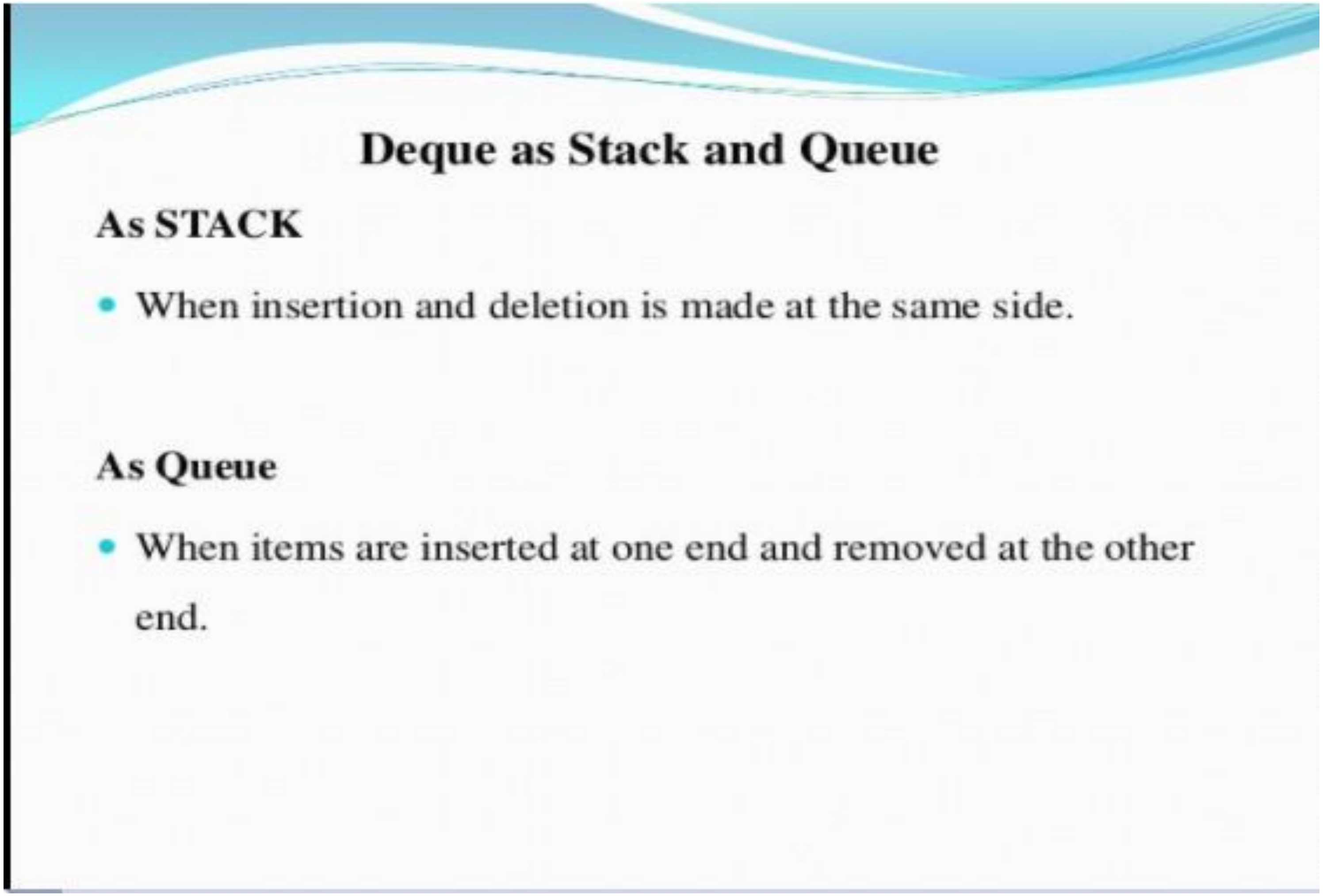
TYPES OF DEQUE

❑ Input restricted Deque

- Elements can be inserted only at one end.
- Elements can be removed from both the ends.

❑ Output restricted Deque

- Elements can be removed only at one end.
- Elements can be inserted from both the ends.



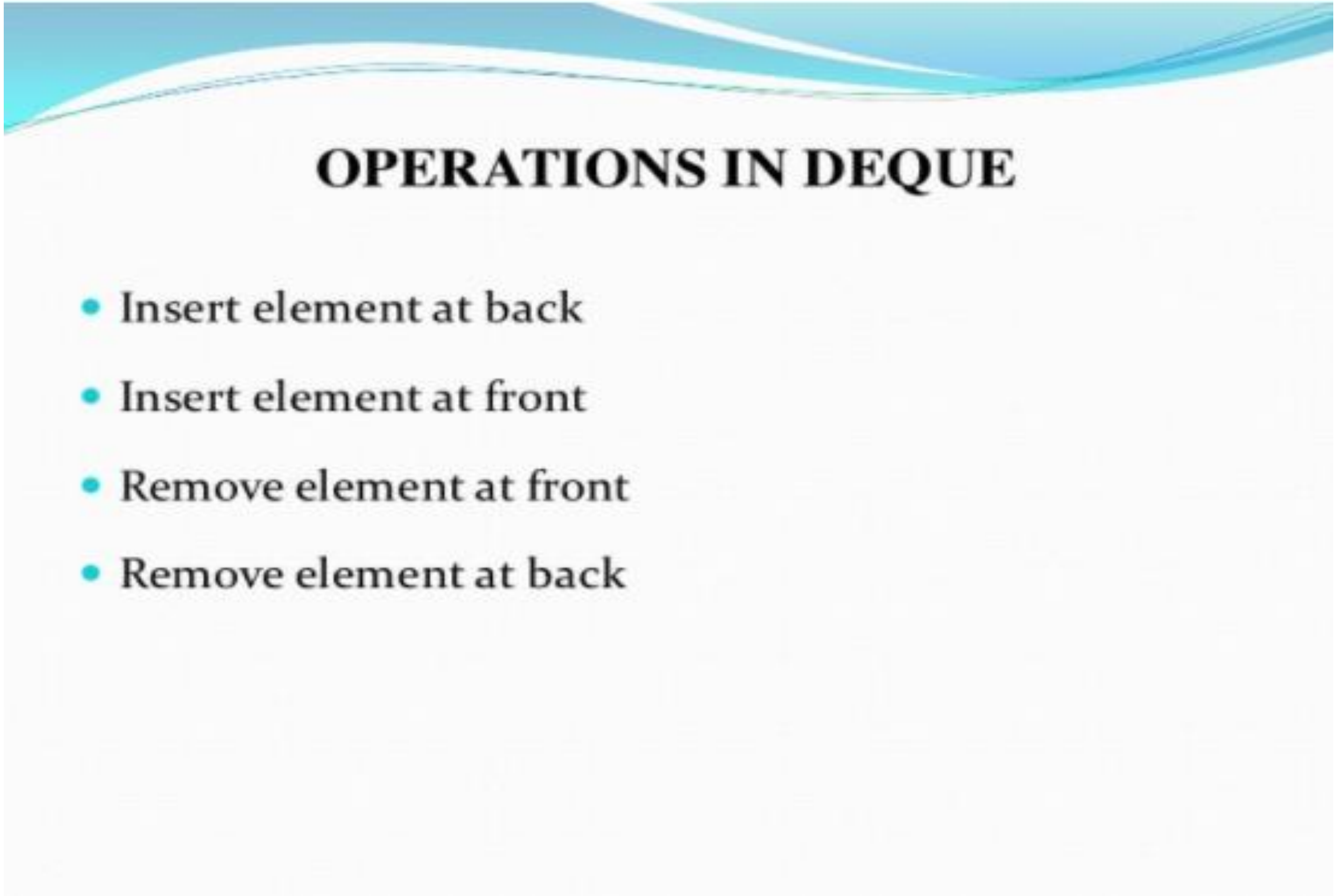
Deque as Stack and Queue

As STACK

- When insertion and deletion is made at the same side.

As Queue

- When items are inserted at one end and removed at the other end.



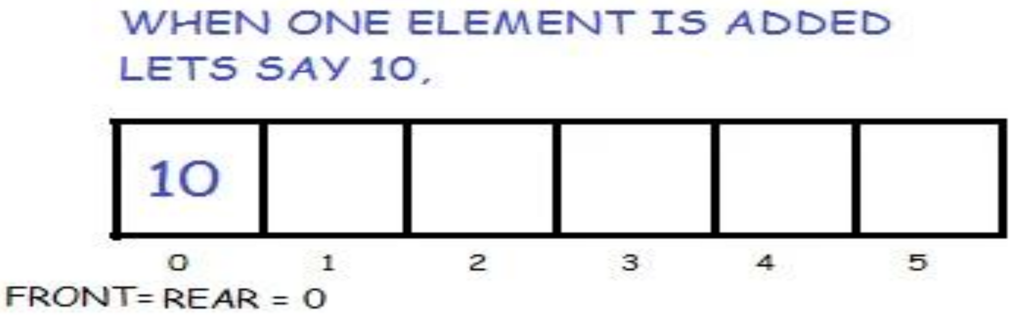
OPERATIONS IN DEQUE

- Insert element at back
- Insert element at front
- Remove element at front
- Remove element at back

Insert element at front

First we check if the queue is full. If its not full we insert an element at front end by following the given conditions :

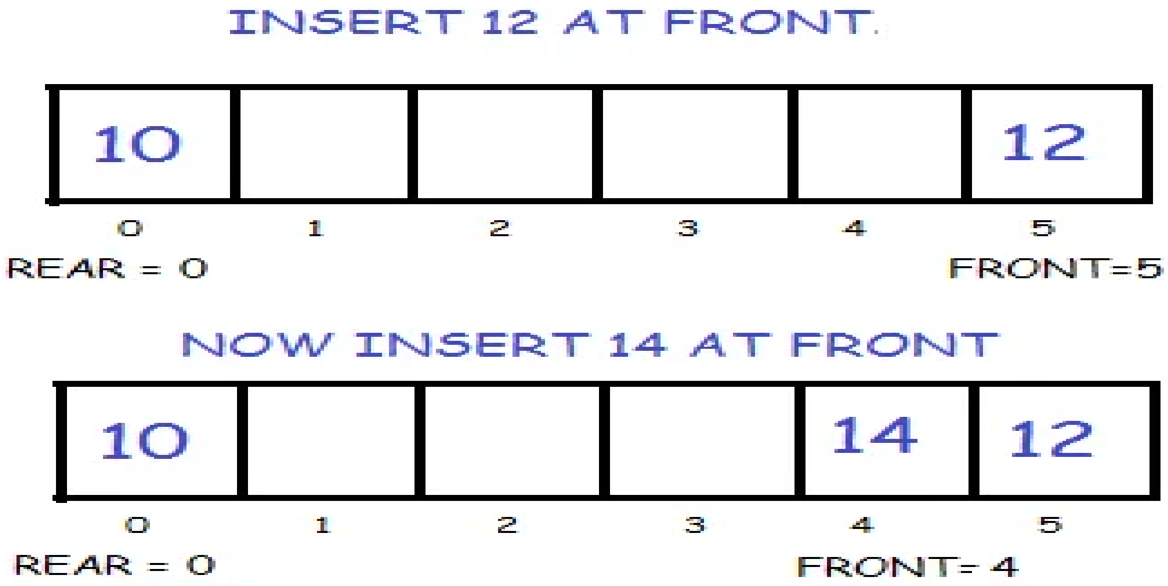
- If the queue is empty then intialize front and rear to 0. Both will point to the first element.



WHEN ONE ELEMENT IS ADDED LETS SAY 10,

Insert element at front

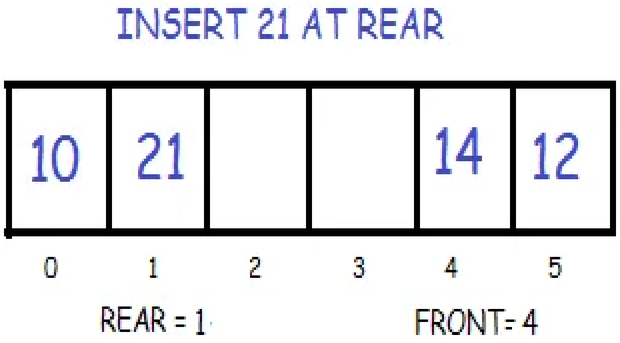
- **Else we decrement front and insert the element. Since we are using circular array, we have to keep in mind that if front is equal to 0 then instead of decreasing it by 1 we make it equal to SIZE-1.**



Insert element at Back

Again we check if the queue is full. If its not full we insert an element at back by following the given conditions:

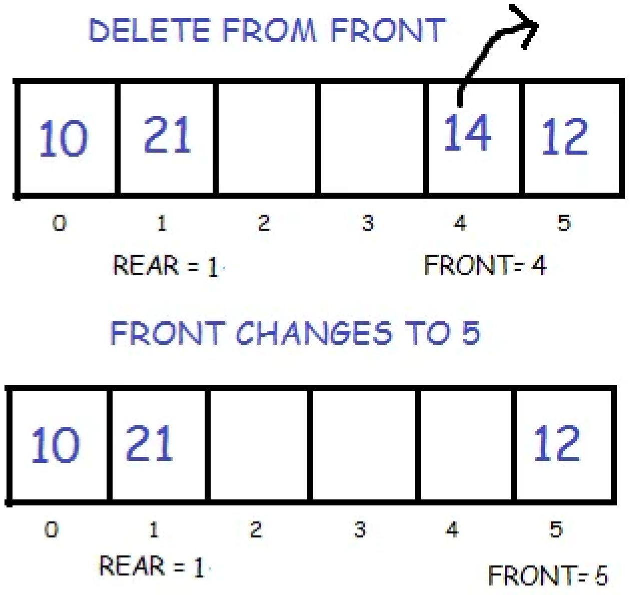
- **If the queue is empty then intialize front and rear to 0. Both will point to the first element.**
- **Else we increment rear and insert the element. Since we are using circular array, we have to keep in mind that if rear is equal to SIZE-1 then instead of increasing it by 1 we make it equal to 0.**



Delete First Element

In order to do this, we first check if the queue is empty. If its not then delete the front element by following the given conditions :

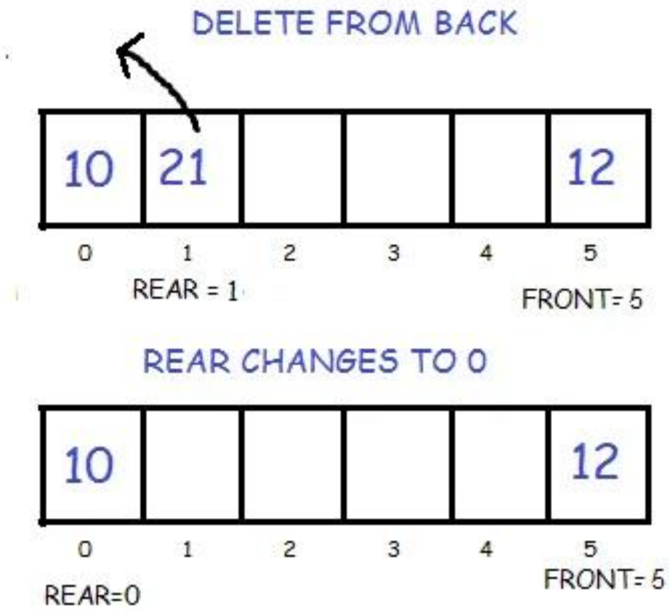
- If only one element is present we once again make front and rear equal to -1.
- Else we increment front. But we have to keep in mind that if front is equal to SIZE-1 then instead of increasing it by 1 we make it equal to 0.



Delete Last Element

Inorder to do this, we again first check if the queue is empty. If its not then we delete the last element by following the given conditions :

- If only one element is present we make front and rear equal to -1.
- Else we decrement rear. But we have to keep in mind that if rear is equal to 0 then instead of decreasing it by 1 we make it equal to SIZE-1.

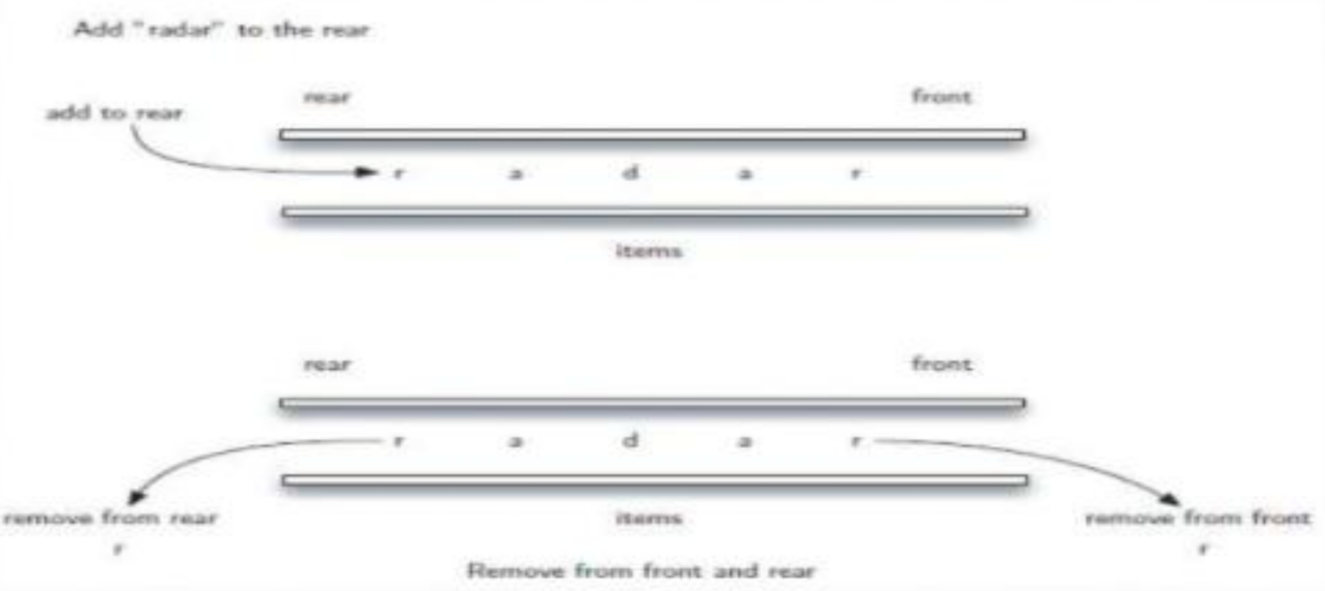


Check if Queue is empty/full

```
if(front == -1)
            return true;
  else

            return false;

if((front == 0 && rear == SIZE-1)  ||
            (front == rear + 1))
      return true;
  else

      return false;
```

APPLICATIONS OF DEQUE

Palindrome-checker



Add "radar" to the rear

Remove from front and rear

palindrome checker

We will process the string from left to right and add each character to the rear of the deque.

At this point, the deque will be acting very much like an ordinary queue.

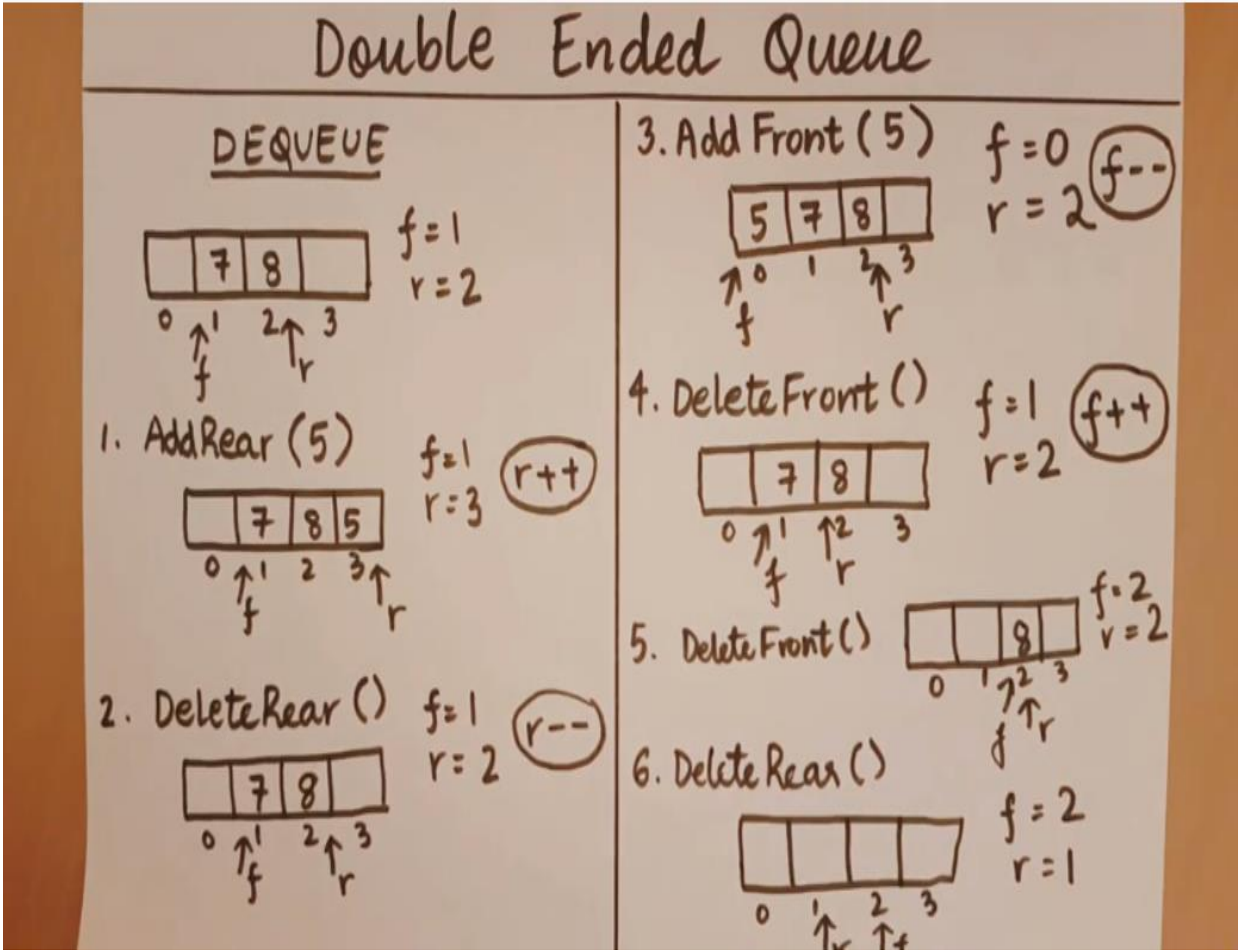
However, we can now make use of the dual functionality of the deque.

The front of the deque will hold the first character of the string and the rear of the deque will hold the last character.

Since we can remove both of them directly, we can compare them and continue only if they match.

If we can keep matching first and the last items, we will eventually either run out of characters or be left with a deque of size 1 depending on whether the length of the original string was even or odd.

In either case, the string must be a palindrome.



Double Ended Queue

DEQUEUE

0	1	2	3
	7	8	
	↑f	↑r	

f = 1
r = 2

1. AddRear (5)

0	1	2	3
	7	8	5
	↑f		↑r

f = 1
r = 3
r++

2. DeleteRear ()

0	1	2	3
	7	8	
	↑f	↑r	

f = 1
r = 2
r--

3. Add Front (5)

0	1	2	3
5	7	8	
↑f		↑r	

f = 0
r = 2
f--

4. Delete Front ()

0	1	2	3
	7	8	
	↑f	↑r	

f = 1
r = 2
f++

5. Delete Front ()

0	1	2	3
		8	
		↑f ↑r	

f = 2
r = 2

6. Delete Rear ()

0	1	2	3
	↑r	↑f	

f = 2
r = 1